Package ‘HGC’

July 13, 2023

Type Package

Title A fast hierarchical graph-based clustering method

Version 1.8.0

Description HGC (short for Hierarchical Graph-based Clustering) is an R package for conducting hierarchical clustering on large-scale single-cell RNA-seq (scRNA-seq) data. The key idea is to construct a dendrogram of cells on their shared nearest neighbor (SNN) graph. HGC provides functions for building graphs and for conducting hierarchical clustering on the graph. The users with old R version could visit https://github.com/XuegongLab/HGC/tree/HGC4oldRVersion to get HGC package built for R 3.6.

License GPL-3

Encoding UTF-8

SystemRequirements C++11

Depends R (>= 4.1.0)

Imports Rcpp (>= 1.0.0), RcppEigen(>= 0.3.2.0), Matrix, RANN, ape, dendextend, ggplot2, mclust, patchwork, dplyr, grDevices, methods, stats

LinkingTo Rcpp, RcppEigen

Suggests BiocStyle, rmarkdown, knitr, testthat (>= 3.0.0)

VignetteBuilder knitr

biocViews SingleCell, Software, Clustering, RNASeq, GraphAndNetwork, DNASeq

Config/testthat/edition 3

NeedsCompilation yes

git_url https://git.bioconductor.org/packages/HGC

git_branch RELEASE_3_17

git_last_commit cd4628d

git_last_commit_date 2023-04-25

Date/Publication 2023-07-13

Author Zou Ziheng [aut],
Hua Kui [aut],
XGlab [cre, cph]

Maintainer XGlab <xglab@mail.tsinghua.edu.cn>

R topics documented:

`CKNN.Construction` *Building Unweighted Continuous K Nearest Neighbor Graph*

Description

This function builds a Continuous K Nearest Neighbor (CKNN) graph in the input feature space using Euclidean distance metric.

Usage

```
CKNN.Construction(mat, k, delta)
```

Arguments

`mat`	the input data saved as a numerical matrix. The columns are the features and the rows are the samples.
`k`	the number of nearest neighbors for building the CKNN graph.
`delta`	the parameter related with the distance threshold.

Details

This function fist built a KNN graph from the input data. Then the CKNN graph is built from the KNN graph. For node i and node j in the KNN graph, CKNN will link them if the distance d(i,j) between node i and node j is less than δ times of the geometric mean of d_k(i) and d_k(j). Here δ is the parameter, d_k(i) and d_k(j) are the distances from node i or node j to their k nearest neighbor.

Value

An n by n binary dgCMatrix object C, where n is the number of input samples. The matrix C is the adjacency matrix of the built CKNN graph. C[i,j] = 1 means that there is an edge between sample i and sample j.

Examples

```
data(Pollen)
Pollen.PCs <- Pollen[["PCs"]]
G <- CKNN.Construction(Pollen.PCs)
```

`FindClusteringTree` *The HGC algorithm embedded in Seurat pipeline*

Description

The function runs hierarchical clustering with `HGC.dendrogram` on the SNN or KNN calculated by the `Seurat` pipeline. The output clustering tree is also packaged in the `Seurat` object.

Usage

```
FindClusteringTree(object, graph.type)
```

Arguments

`object`	The Seurat object containing the graphs built with scRNA-seq data.
`graph.type`	The type of graphs used for the hierarchical clustering, could be "SNN" or "KNN". The default value is "SNN".

Details

For the KNN graph, we symmetrize it by adding its transposition on the graph. And for the details of data preprocessing and graph construction by `Seurat`, please check the `Seurat` vignettes.

Value

An `Seurat` object. The clustering tree is saved under the item `graphs`, i.e. `object@graphs$ClusteringTree`.

Note

The function needs the R package `Seurat`. We recommend that the version of `Seurat` is higher than version 3.0.

Examples

```
## Do not run
# require(Seurat)

## DemoData is a input gene expression matrix.
# DemoData.seuratobj <- CreateSeuratObject(counts = DemoData,
#                                          min.cells = 20)
# DemoData.seuratobj <- NormalizeData(object = DemoData.seuratobj,
#                                     verbose = F)
# DemoData.seuratobj <- ScaleData(object = DemoData.seuratobj,
#                                 features = row.names(DemoData.seuratobj),
#                                 verbose = F)
# DemoData.seuratobj <- FindVariableFeatures(object = DemoData.seuratobj,
#                                            nfeatures = 2000, verbose = F)
# DemoData.seuratobj <- RunPCA(object = DemoData.seuratobj,
#                              npcs = 100, verbose = F)
# DemoData.seuratobj <- FindNeighbors(object = DemoData.seuratobj,
#                                     nn.eps = 0.5, k.param = 30,
#                                     dims = 1:25, verbose = F)
# DemoData.seuratobj <- FindClusteringTree(object = DemoData.seuratobj,
#                                          graph.type = "SNN")
```

`HGC.dendrogram` *Hierarchical Graph-based Clustering*

Description

Hierarchical clustering on a given undirected graph.

Usage

```
HGC.dendrogram(G)
```

Arguments

`G`	an object which represents the adjacency matrix of the graph, where G[i,j] is the weight of the edge between node i and node j, and zero means no link. The supported data structures include `matrix`, `dgCMatrix`, `graph`, and `igraph`.

Details

The function runs a hierarchical clustering on the given graph. It is a recursive procedure of two steps, first, the node pair sampling ratio is used as the distance metric to search the nearest neighbor pairs. Then the neighbor pair are merged and the graph is updated. The whole procedure is accelerated using the nearest neighbor chain algorithm. The algorithm stops when there's only one node left in the updated graph.

Value

An object of class `hclust` defined by the `hclust` function in the `stats` package. It is a list containing the clustering tree information with the components:

`merge`	an *n-1* by 2 matrix. It records the two nodes in each merging step.
`height`	a set of *n-1* real values. It is the height of the non-leaf nodes in the tree.
`order`	a vector giving the permutation of the original observations suitable for plotting.
`labels`	labels for the objects being clustered. Same as the rownames of G in default.
`call`	the call which produced the result.
`method`	the cluster method that has been used.
`dist.method`	the distance used here.

More details about the components are in the hclust.

Examples

```
data(Pollen)
Pollen.PCs <- Pollen[["PCs"]]
G <- SNN.Construction(Pollen.PCs, 25, 0.15)
tree = HGC.dendrogram(G)
```

`HGC.parameter` *Recording the Parameters of the Graph-based Hierarchical Clustering*

Description

This function records and outputs the length of the nearest neighbor chain and the average neighbor number in each iteration of hierarchical clustering. These values can be used for the time complexity analysis of `HGC.dendrogram`.

Usage

```
HGC.parameter(G)
```

Arguments

`G`	an undirected graph saved as a dgCMatrix. The matrix G is the adjacency matrix of the graph, and element G[i,j] is the weight of the edge between node i and node j. Zeros in the matrix mean no link between nodes here.

Details

This function contains the whole function of `HGC.dendrogram`, but will record the key parameters during the whole process. The function is provided for advanced users to conduct time complexity analysis on their own data. The construction of the dendrogram is a recursive procedure of two steps: 1.finding the nearest neighbour pair, 2. merge the node pair and update the graph. For different data structures of graph, there's a trade-off between the time consumptions of the two steps. Generally speaking, storing more information about the graph makes it faster to find the nearest neighbour pair (step 1) but slower to update the graph (step 2). We have experimented serval datasets and chosen the best data structure in `HGC.dendrogram` for the overall efficiency.

Value

A 2 by m matrix. The two rows of the matrix are the nearest neighbor chain length and the average neighbor number. m is equal to n-s, where s is the number of unconnected parts in the graph.

Examples

```
data(Pollen)
Pollen.PCs <- Pollen[["PCs"]]
G <- SNN.Construction(Pollen.PCs, 25, 0.15)
record = HGC.parameter(G)
```

`HGC.PlotARIs` *Calculating and Visualizing ARIs of the clustering results with given labels*

Description

The function cut the dendrogram into specific clusters at different levels and compared the clusterings with given labels using Adjusted Rand Index (ARI)

Usage

```
HGC.PlotARIs(tree, k.min, k.max, labels, return.ARI)
```

Arguments

`tree`	the input clustering tree saved as `hclust` data structure.
`k.min`	the minimum number to cut the tree.
`k.max`	the maximum number to cut the tree.
`labels`	a data frame or a matrix to store the label information. Different labels should be in different columns and the users should name the columns correspondingly.
`return.ARI`	a bool variable to choose whether output the ARI matrix.

Details

ARI is a widely used index to evaluate the consistence between two partitions of the same samples. This function will first cut a given tree into specific number of clusters using the function `cutree`. Then it calculates the ARIs between the clustering result and the given labels with the help of R package `mclust`. The function does such cutting and calculation for different ks between k.min and k.max. Finally it visualize these results using a line chart. ARIs with different labels are shown as different lines with different colors in the figure.

Value

A line chart will be drawn and a matrix of the ARIs will be returned.

Examples

```
data(Pollen)
Pollen.PCs <- Pollen[["PCs"]]
Pollen.Label.Tissue <- Pollen[["Tissue"]]
Pollen.Label.CellLine <- Pollen[["CellLine"]]

Pollen.SNN <- SNN.Construction(Pollen.PCs)
Pollen.ClusteringTree <- HGC.dendrogram(G = Pollen.SNN)
Pollen.labels <- data.frame(Tissue = Pollen.Label.Tissue,
                            CellLine = Pollen.Label.CellLine)
HGC.PlotARIs(tree = Pollen.ClusteringTree,
                            k.min = 2, k.max = 15,
                            labels = Pollen.labels)
```

`HGC.PlotDendrogram` *Visualizing the dendrogram*

Description

The function will plot the dendrogram, with different colors for different clusters.

Usage

```
HGC.PlotDendrogram(tree, k, plot.label, labels)
```

Arguments

`tree`	the input clustering tree saved as `hclust` data structure.
`k`	the number of clusters to cut the tree into.
`plot.label`	a bool variable. It decides whether the function will add color bars.
`labels`	a data frame or a matrix to store the label information. Different labels should be in different columns and the users should name the columns correspondingly. The label information will show in the figure as color bars below the dendrogram, and each label takes one color bar.

Details

The function plots the clustering tree, with alternative colors showing the clustering results and the label information. It is based on the R package dendextend which contains many parameters for the visualization. For users' convenience, most of the parameters are set to be the default value. Advanced users could visit the vignette of dendextend for more flexible visualization.

Value

The function will return 1 if the dendrogram is successfully drawn.

Examples

```
data(Pollen)
Pollen.PCs <- Pollen[["PCs"]]
Pollen.Label.Tissue <- Pollen[["Tissue"]]
Pollen.Label.CellLine <- Pollen[["CellLine"]]

Pollen.SNN <- SNN.Construction(Pollen.PCs)
Pollen.ClusteringTree <- HGC.dendrogram(G = Pollen.SNN)
Pollen.labels <- data.frame(Tissue = Pollen.Label.Tissue,
                            CellLine = Pollen.Label.CellLine)
HGC.PlotDendrogram(tree = Pollen.ClusteringTree,
                   k = 5, plot.label = TRUE,
                   labels = Pollen.labels)
```

HGC.PlotParameter *Visualizing the Parameter Records during Clustering*

Description

The function visualizes the parameter output from HGC.parameter.

Usage

```
HGC.PlotParameter(record, parameter)
```

Arguments

record	the input record matrix of parameters from HGC.parameter.
parameter	a string with alternatives "CL" or "ANN". Choose "CL" to plot the chain lengths and "ANN" to plot the average neighbor number.

Details

The chain length(CL) and average neighbor number(ANN) are key factors related with the time complexity of clustering by HGC. The function provides the visualization of them.

Value

The function will return 1 if the dendrogram is successfully drawn.

Examples

```
data(Pollen)
Pollen.PCs <- Pollen[["PCs"]]
Pollen.SNN <- SNN.Construction(Pollen.PCs)
Pollen.ParameterRecord <- HGC.parameter(G = Pollen.SNN)
HGC.PlotParameter(Pollen.ParameterRecord, parameter = "CL")
HGC.PlotParameter(Pollen.ParameterRecord, parameter = "ANN")
```

KNN.Construction *Building Unweighted K Nearest Neighbor Graph*

Description

This function builds an Unweighted K Nearest Neighbor (KNN) graph in the input feature space using Euclidean distance metric.

Usage

```
KNN.Construction(mat, k)
```

Arguments

`mat`	the input data saved as a numerical matrix. The columns are the features and the rows are the samples.
`k`	the number of nearest neighbors for building the KNN graph.

Details

This function builds a KNN graph of the input data. The main function comes from the R package `RANN`.

Value

An n by n binary dgCMatrix object C, where n is the number of input samples. The matrix C is the adjacency matrix of the built KNN graph. C[i,j] = 1 means that there is an edge between sample i and sample j.

Examples

```
data(Pollen)
Pollen.PCs <- Pollen[["PCs"]]
G <- KNN.Construction(Pollen.PCs)
```

MST.Construction *Building Unweighted Minimum Spanning Tree Graph*

Description

This function builds an Unweighted Minimum Spanning Tree (MST) graph in the input feature space using Euclidean distance metric.

Usage

```
MST.Construction(mat)
```

Arguments

`mat`	the input data saved as a numerical matrix. The columns are the features and the rows are the samples.

Details

This function builds a MST graph of the input data. The main function come from the R package `ape`.

Value

An n by n binary dgCMatrix object C, where n is the number of input samples. The matrix C is the adjacency matrix of the built MST graph. C[i,j] = 1 means that there is an edge between sample i and sample j.

Examples

```
data(Pollen)
Pollen.PCs <- Pollen[["PCs"]]
G <- MST.Construction(Pollen.PCs)
```

PMST.Construction *Building Unweighted Perturbed Minimum Spanning Tree Graph*

Description

This function builds an Unweighted Perturbed Minimum Spanning Tree (PMST) graph in the input feature space using Euclidean distance metric.

Usage

```
PMST.Construction(mat, iter, r)
```

Arguments

`mat`	the input data saved as a numerical matrix. The columns are the features and the rows are the samples.
`iter`	the number of perturbation.
`r`	the parameter about the strength of the perturbation.

Details

The function builds a PMST graph of the input data. PMST is the combination of a number of MSTs, which are built in the perturbed data spaces.

Value

An n by n binary dgCMatrix object C, where n is the number of input samples. The matrix C is the adjacency matrix of the built PMST graph. C[i,j] = 1 means that there is an edge between sample i and sample j.

Examples

```
data(Pollen)
Pollen.PCs <- Pollen[["PCs"]]
G <- PMST.Construction(Pollen.PCs)
```

`Pollen` *Embeddings of the Pollen datasets in the principal component space.*

Description

The dataset is the low dimensional principal components and labels of cells from the Pollen dataset. The 301 cells are from 11 different cell lines and are classified into 4 tissues.

Format

An object of class `list`.

The `list` contains three elements: First, a matrix with 301 rows and 25 columns, saved as `Pollen[["PCs"]]`. The rows are cells and columns are principal components. Second, the label in tissue level, saved as a vector in `Pollen[["Tissue"]]`. Third, the label in cell line level, saved as a vector in `Pollen[["CellLine"]]`.

Source

https://www.nature.com/articles/nbt.2967

RNN.Construction — *Building Unweighted ϵ Nearest Neighbor Graph*

Description

This function builds an ϵ Nearest Neighbor graph in the input feature space using Euclidean distance metric.

Usage

```
RNN.Construction(mat, max_dist)
```

Arguments

`mat`	the input data saved as a numerical matrix. The columns are the features and the rows are the samples.
`max_dist`	the threshold distance. The edges whose lengths are less than `max_dist` will be kept in the graph.

Details

The function builds an ϵ Nearest Neighbor graph which saved as a sparse matrix.

Value

An n by n binary dgCMatrix object C, where n is the number of input samples. The matrix C is the adjacency matrix of the built RNN graph. C[i,j] = 1 means that there is an edge between sample i and sample j.

Examples

```
data(Pollen)
Pollen.PCs <- Pollen[["PCs"]]
G <- RNN.Construction(Pollen.PCs, 20)
```

SNN.Construction — *Building Unweighted Shared Nearest Neighbor Graph*

Description

This function builds a Shared Nearest Neighbor (SNN) graph in the input feature space using Euclidean distance metric.

Usage

```
SNN.Construction(mat, k, threshold)
```

Arguments

`mat`	the input data saved as a numerical matrix. The columns are the features and the rows are the samples.
`k`	the number of nearest neighbor number to build the original KNN.
`threshold`	the threshold parameter for the Jaccard index. The edges in KNN whose Jaccard indices are lower than it will be removed in building the SNN.

Details

The function builds an SNN which saved as a sparse matrix.

Value

An n by n binary dgCMatrix object C, where n is the number of input samples. The matrix C is the adjacency matrix of the built SNN graph. C[i,j] = 1 means that there is an edge between sample i and sample j.

Examples

```
data(Pollen)
Pollen.PCs <- Pollen[["PCs"]]
G <- SNN.Construction(Pollen.PCs)
```

Index