Package 'MapperAlgo'

August 23, 2025

Title Topological Data Analysis: Mapper Algorithm

Version 1.0.4

Date 2025-08-23

Maintainer ChiChien Wang <kennywang2003@gmail.com>

Description The Mapper algorithm from Topological Data Analysis, the steps are as follows 1. Define a filter (lens) function on the data. 2. Perform clustering within each level set. 3. Generate a complex from the clustering results.

Depends R (>= 3.1.2)

Imports parallel, doParallel, foreach, networkD3, igraph, ggraph, tidygraph, ggplot2, htmlwidgets, rlang

Suggests fastcluster, cluster, dbscan, testthat (>= 3.0.0)

License MIT + file LICENSE

URL https://github.com/kennywang112/MapperAlgo/

BugReports https://github.com/kennywang112/MapperAlgo/issues

Encoding UTF-8

Config/testthat/edition 3

RoxygenNote 7.3.2

NeedsCompilation no

Author ChiChien Wang [aut, cre, trl],
Paul Pearson [ctb],
Daniel Muellner [ctb],
Gurjeet Singh [ctb]

Repository CRAN

Date/Publication 2025-08-23 00:10:07 UTC

Contents

`cluster_cutoff_at_first_empty_bin` *Cut the hierarchical clustering tree to define clusters*

Description

Cut the hierarchical clustering tree to define clusters

Usage

```
cluster_cutoff_at_first_empty_bin(heights, diam, num_bins_when_clustering)
```

Arguments

`heights`	Heights of the clusters.
`diam`	Diameter of the clusters.
`num_bins_when_clustering`	Number of bins when clustering.

Value

The cutoff height for the clusters.

`cover_points` *Cover points based on intervals and overlap*

Description

Cover points based on intervals and overlap

Usage

```
cover_points(
  lsfi,
  filter_min,
  interval_width,
  percent_overlap,
  filter_values,
  num_intervals,
  type = "stride"
)
```

Arguments

lsfi	Level set flat index.
filter_min	Minimum filter value.
interval_width	Width of the interval.
percent_overlap	Percentage overlap between intervals.
filter_values	The filter values to be analyzed.
num_intervals	Number of intervals.
type	Type of interval, either 'stride' or 'extension'.

Value

Indices of points in the range.

---

find_best_k_for_kmeans

Find the optimal number of clusters for k-means

Description

This function calculates the total within-cluster sum of squares (WSS) for a range of cluster numbers and identifies the best number of clusters (k) based on the elbow method.

Usage

```
find_best_k_for_kmeans(dist_object, max_clusters = 10)
```

Arguments

dist_object	A distance matrix or data frame containing the data to be clustered.
max_clusters	The maximum number of clusters to test for k-means. Default is 10.

Value

The optimal number of clusters (k) based on the elbow method.

`MapperAlgo` *Mapper Algorithm*

Description

Implements the Mapper algorithm for Topological Data Analysis (TDA). It divides data into intervals, applies clustering within each interval, and constructs a simplicial complex representing the structure of the data.

Usage

```
MapperAlgo(
  filter_values,
  percent_overlap,
  methods,
  method_params = list(),
  cover_type = "extension",
  intervals = NULL,
  interval_width = NULL,
  num_cores = 1
)
```

Arguments

`filter_values`	A data frame or matrix of the data to be analyzed.
`percent_overlap`	Percentage of overlap between consecutive intervals.
`methods`	Specify the clustering method to be used, e.g., "hclust" or "kmeans".
`method_params`	A list of parameters for the clustering method.
`cover_type`	Type of interval, either 'stride' or 'extension'.
`intervals`	An integer specifying the number of intervals.
`interval_width`	The width of each interval.
`num_cores`	Number of cores to use for parallel computing.

Value

A list containing the Mapper graph components:

adjacency The adjacency matrix of the Mapper graph.

num_vertices The number of vertices in the Mapper graph.

level_of_vertex A vector specifying the level of each vertex.

points_in_vertex A list of the indices of the points in each vertex.

points_in_level_set A list of the indices of the points in each level set.

vertices_in_level_set A list of the indices of the vertices in each level set.

mapperEdges *Create Mapper Edges*

Description

This function generates the edges of the Mapper graph by analyzing the adjacency matrix. It returns a data frame with source and target vertices that are connected by edges.

Usage

```
mapperEdges(m)
```

Arguments

m	The Mapper output object that contains the adjacency matrix and other graph components.

Value

A data frame containing the source (`Linksource`), target (`Linktarget`), and edge values (`Linkvalue`) for the graph's edges.

MapperPlotter *Plot Mapper Result*

Description

Visualizes the Mapper output using either networkD3 or ggraph.

Usage

```
MapperPlotter(Mapper, label, data, type = "forceNetwork", avg = FALSE)
```

Arguments

Mapper	Mapper object.
label	Label of the data.
data	Data.
type	Visualization type: "forceNetwork" or "ggraph".
avg	Whether coloring the nodes by average label or majority label.

Value

Plot of the Mapper.

mapperVertices *Create Mapper Vertices*

Description

This function generates the vertices of the Mapper graph, including their labels and groupings. It returns a data frame with the vertex names, the group each vertex belongs to, and the size of each vertex.

Usage

```
mapperVertices(m, pt_labels)
```

Arguments

m	The Mapper output object that contains information about the vertices and level sets.
pt_labels	A vector of point labels to be assigned to the points in each vertex.

Value

A data frame containing the vertex names (Nodename), group information (Nodegroup), and vertex sizes (Nodesize).

perform_clustering *Perform clustering within a level set*

Description

Perform clustering within a level set

Usage

```
perform_clustering(
  points_in_this_level,
  filter_values,
  methods,
  method_params = list()
)
```

Arguments

points_in_this_level	Points in the current level set.
filter_values	The filter values.
methods	Specify the clustering method to be used, e.g., "hclust" or "kmeans".
method_params	A list of parameters for the clustering method.

Value

A list containing the number of vertices, external indices, and internal indices.

`simplcial_complex` *Construct adjacency matrix of the simplicial complex*

Description

Construct adjacency matrix of the simplicial complex

Usage

```
simplcial_complex(
  filter_values,
  vertex_index,
  num_levelsets,
  num_intervals,
  vertices_in_level_set,
  points_in_vertex
)
```

Arguments

`filter_values`	A matrix of filter values.
`vertex_index`	The number of vertices.
`num_levelsets`	The total number of level sets.
`num_intervals`	A vector representing the number of intervals for each filter.
`vertices_in_level_set`	A list where each element contains the vertices corresponding to each level set.
`points_in_vertex`	A list where each element contains the points corresponding to each vertex.

Value

An adjacency matrix representing the simplicial complex.

to_lsfi *Convert level set multi-index (lsmi) to flat index (lsfi)*

Description

Convert level set multi-index (lsmi) to flat index (lsfi)

Usage

```
to_lsfi(lsmi, num_intervals)
```

Arguments

lsmi	Level set multi-index.
num_intervals	Number of intervals.

Value

A flat index corresponding to the multi-index.

to_lsmi *Convert level set flat index (lsfi) to multi-index (lsmi)*

Description

Convert level set flat index (lsfi) to multi-index (lsmi)

Usage

```
to_lsmi(lsfi, num_intervals)
```

Arguments

lsfi	Level set flat index.
num_intervals	Number of intervals.

Value

A multi-index corresponding to the flat index.

Index