

Package ‘MapperAlgo’

August 31, 2026

Title Topological Data Analysis: Mapper Algorithm

Version 1.2.0

Date 2026-08-30

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 inaparc, ppclust, parallel, doParallel, foreach, networkD3,
plotly, igraph, ggplot2, htmlwidgets, webshot2, jsonlite,
rlang, viridisLite, mclust, nortest

Suggests fastcluster, cluster, dbscan, mlr3, mlr3cluster, testthat (>= 3.0.0)

License MIT + file LICENSE

URL <https://github.com/TDA-R/MapperAlgo>

BugReports <https://github.com/TDA-R/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 2026-08-31 14:40:02 UTC

Contents

cluster_cutoff_at_first_empty_bin	2
cover_points	3
CPEmbedding	3
find_best_k_for_kmeans	4
FuzzyMapperAlgo	5
GMapperAlgo	5
GridSearch	6
MapperAlgo	7
MapperCorrelation	8
mapperEdges	9
MapperPlotter	9
MapperPlotter3D	10
mapperVertices	11
perform_clustering	11
perform_clustering_mlr3	12
recursive_gaussian_split	13
save_mapper_png	13
simplcial_complex	14
to_lsfi	15
to_lsmi	15
Index	16

cluster_cutoff_at_first_empty_bin
<i>Cut the hierarchical clustering tree to define clusters</i>

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	<i>Cover points based on intervals and overlap</i>
--------------	--

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.

CPEmbedding	<i>Conditional Probability Embedding for Mapper Nodes</i>
-------------	---

Description

The origin Mapper includes mean and majority label embeddings. And this function provides another way to color the Mapper nodes. The function is useful to connect original data for color labeling, especially if you're interested in characteristic attributes.

Usage

```
CPEmbedding(
  mapper,
  original_data,
  columns = list(),
  a_level = NULL,
  b_level = NULL
)
```

Arguments

mapper	A Mapper object created by the MapperAlgo function.
original_data	Original dataframe, not the filter values.
columns	Two columns in original_data to compute conditional probability.
a_level	The level (attribute) of column A to condition on. If NULL, the first level is used.
b_level	The level (attribute) of column B for which the conditional probability is computed. If NULL, the first level is used.

Value

A list of conditional probabilities value for each Mapper node.

```
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.

FuzzyMapperAlgo

*Fuzzy Mapper Algorithm (Fixed Adjacency Calculation)***Description**

Implements a variant of the Mapper algorithm using Fuzzy C-Means (FCM) clustering for the level sets.

Usage

```
FuzzyMapperAlgo(
  original_data,
  filter_values,
  cluster_n = 5,
  fcm_threshold = NULL,
  methods,
  method_params = list(),
  num_cores = 1
)
```

Arguments

<code>original_data</code>	Original dataframe, not the filter values.
<code>filter_values</code>	A data frame or matrix of the data to be analyzed.
<code>cluster_n</code>	Number of fuzzy clusters (c in FCM). Default is 5.
<code>fcm_threshold</code>	Membership threshold (τ). Points with $u > \tau$ are included in the interval.
<code>methods</code>	Specify the clustering method to be used, e.g., "hclust" or "kmeans".
<code>method_params</code>	A list of parameters for the clustering method.
<code>num_cores</code>	Number of cores to use for parallel computing.

Value

A MapperAlgo object same as MapperAlgo output

GMapperAlgo

*G-Mapper Algorithm***Description**

Implements a Mapper algorithm using Anderson-Darling tests and Gaussian Mixture Models (GMM) to automatically learn the cover.

Usage

```
MapperAlgo(
  original_data,
  filter_values,
  AD_threshold = 10,
  g_overlap = 0.1,
  methods,
  method_params = list(),
  num_cores = 1
)
```

Arguments

<code>original_data</code>	Original dataframe, not the filter values.
<code>filter_values</code>	A data frame or matrix of the data to be analysed (1-D).
<code>AD_threshold</code>	Critical value for the Anderson-Darling test
<code>g_overlap</code>	The geometric overlap percentage when splitting an interval
<code>methods</code>	Specify the clustering method to be used, e.g., "hclust" or "kmeans".
<code>method_params</code>	A list of parameters for the clustering method.
<code>num_cores</code>	Number of cores to use for parallel computing.

Value

A MapperAlgo object same as MapperAlgo output

GridSearch	<i>GridSearch searched over a list of interval width and overlap, useful for visualizing the convergence of the Mapper.</i>
------------	---

Description

GridSearch searched over a list of interval width and overlap, useful for visualizing the convergence of the Mapper.

Usage

```
GridSearch(
  original_data,
  filter_values,
  label,
  column = "label",
  cover_type = "stride",
  width_vec = c(0.5, 1, 1.5),
  overlap_vec = c(10, 20, 30, 40),
  num_cores = 12,
```

```

    out_dir = "mapper_grid_outputs",
    avg = FALSE,
    use_embedding = NULL
  )

```

Arguments

<code>original_data</code>	Original dataframe, not the filter values.
<code>filter_values</code>	A numeric matrix or data frame of filter values (rows are samples, columns are filter dimensions).
<code>label</code>	A vector of labels for coloring the Mapper nodes.
<code>column</code>	The original column name (use when <code>use_embedding=TRUE</code>).
<code>cover_type</code>	The type of cover to use "stride" or "extension".
<code>width_vec</code>	A vector of interval widths.
<code>overlap_vec</code>	A vector of percent overlaps.
<code>num_cores</code>	Number of cores to use for parallel computing.
<code>out_dir</code>	Directory to save the output.
<code>avg</code>	Whether coloring the nodes by average label or majority label.
<code>use_embedding</code>	Whether to use embedding for coloring (NULL or embedding vector).

Value

A folder containing the PNG files of the Mapper visualizations.

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

```

Arguments

<code>original_data</code>	Original dataframe, not the filter values.
<code>filter_values</code>	A data frame or matrix of the data to be analysed.
<code>percent_overlap</code>	Percentage of overlap between consecutive intervals.
<code>methods</code>	Specify the clustering method to be used, e.g., "hclust" or "kmeans". Mutually exclusive with <code>method</code> .
<code>method_params</code>	A list of parameters for the clustering method.
<code>method</code>	An <code>mlr3cluster</code> Learner, e.g. <code>mlr3cluster::lrn("clust.kmeans", centers = 3)</code> . Mutually exclusive with <code>methods</code> .
<code>cover_type</code>	Type of interval, either 'stride' or 'extension'.
<code>intervals</code>	An integer specifying the number of intervals.
<code>interval_width</code>	The width of each interval.
<code>num_cores</code>	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.

MapperCorrelation	<i>Visualizes the correlation between two Mapper colorings.</i>
-------------------	---

Description

Visualizes the correlation between two Mapper colorings.

Usage

```
MapperCorrelation(mapper, labels = list(), use_embedding = list(FALSE, FALSE))
```

Arguments

<code>mapper</code>	A Mapper object created by the <code>MapperAlgo</code> function.
<code>labels</code>	List of two Mapper color.
<code>use_embedding</code>	List of two booleans indicating whether to use original data or embedding data.

Value

Plot of the correlation between two Mapper.

mapperEdges	<i>Create Mapper Edges</i>
-------------	----------------------------

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	<i>Plot Mapper Result</i>
---------------	---------------------------

Description

Visualizes the Mapper output using networkD3.

Usage

```
MapperPlotter(  
  Mapper,  
  label,  
  avg = FALSE,  
  use_embedding = FALSE,  
  legend_name = NULL  
)
```

Arguments

Mapper	Mapper object.
label	Label of the data.
avg	Whether coloring the nodes by average label or majority label.
use_embedding	Whether to use original data for coloring (TRUE or FALSE).
legend_name	Optional custom title shown on the color legend / colorbar, describing what label represents (e.g. "Eigen centrality"). Defaults to "Label", "Avg(label)", or "Majority label" depending on the coloring mode when left NULL.

Value

Plot of the Mapper.

MapperPlotter3D	<i>Plot Mapper Result in Interactive 3D</i>
-----------------	---

Description

Plot Mapper Result in Interactive 3D

Usage

```
MapperPlotter3D(
  Mapper,
  label,
  avg = FALSE,
  use_embedding = FALSE,
  seed = 42,
  legend_name = NULL
)
```

Arguments

Mapper	Mapper object.
label	Label of the data.
avg	Whether coloring the nodes by average label or majority label.
use_embedding	Whether to use original data for coloring (TRUE or FALSE).
seed	Random seed for the stochastic force-directed layout, so the plot is reproducible across calls.
legend_name	Optional custom title shown on the color legend / colorbar, describing what label represents (e.g. "Eigen centrality"). Defaults to "Label", "Avg(label)", or "Majority label" depending on the coloring mode when left NULL.

Value

An interactive plotly 3D plot of the Mapper graph.

mapperVertices	<i>Create Mapper Vertices</i>
----------------	-------------------------------

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	<i>Perform clustering within a level set</i>
--------------------	--

Description

Perform clustering within a level set

Usage

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

Arguments

original_data Original dataframe, not the filter values.
 filter_values The filter values.
 points_in_this_level
 Points in the current level set.
 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.

```
perform_clustering_mlr3
```

Perform clustering within a level set using an mlr3cluster Learner

Description

Perform clustering within a level set using an mlr3cluster Learner

Usage

```
perform_clustering_mlr3(  
  original_data,  
  filter_values,  
  points_in_this_level,  
  method  
)
```

Arguments

original_data Original dataframe, not the filter values.
 filter_values The filter values.
 points_in_this_level
 Points in the current level set.
 method An mlr3cluster Learner, e.g. mlr3cluster::lrn("clust.kmeans", centers = 3).

Value

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

recursive_gaussian_split

Helper function to recursively split data until it is Gaussian The function now takes geometric boundaries (a, b) instead of indices

Description

Helper function to recursively split data until it is Gaussian The function now takes geometric boundaries (a, b) instead of indices

Usage

```
recursive_gaussian_split(a, b, vals, AD_threshold, g_overlap, depth = 1)
```

Arguments

a	Left boundary of the interval
b	Right boundary of the interval
vals	The original filter values (1D vector)
AD_threshold	The threshold for the Anderson-Darling test to determine Gaussian
g_overlap	The geometric overlap percentage when splitting an interval
depth	Current depth of recursion to prevent infinite loops

Value

A list of geometric intervals that are Gaussian

save_mapper_png	<i>GridSearch searched over a list of interval width and overlap, useful for visualizing the convergence of the Mapper.</i>
-----------------	---

Description

GridSearch searched over a list of interval width and overlap, useful for visualizing the convergence of the Mapper.

Usage

```
save_mapper_png(
  widget,
  png_path,
  vwidth = 1200,
  vheight = 900,
  zoom = 2,
  delay = 0.5
)
```

Arguments

widget	The htmlwidget object to be saved as PNG.
png_path	The file path to save the PNG image.
vwidth	The viewport width for the webshot.
vheight	The viewport height for the webshot.
zoom	The zoom factor for the webshot.
delay	The delay in seconds before taking the snapshot. Useful for allowing time for the widget to fully render.

Value

The snapshot is saved to the specified path.

simplicial_complex	<i>Construct adjacency matrix of the simplicial complex</i>
--------------------	---

Description

Construct adjacency matrix of the simplicial complex

Usage

```
simplicial_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	<i>Convert level set multi-index (lsmi) to flat index (lsfi)</i>
---------	--

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	<i>Convert level set flat index (lsfi) to multi-index (lsmi)</i>
---------	--

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

[cluster_cutoff_at_first_empty_bin, 2](#)
[cover_points, 3](#)
[CPEmbedding, 3](#)

[find_best_k_for_kmeans, 4](#)
[FuzzyMapperAlgo, 5](#)

[GMapperAlgo, 5](#)
[GridSearch, 6](#)

[MapperAlgo, 7](#)
[MapperCorrelation, 8](#)
[mapperEdges, 9](#)
[MapperPlotter, 9](#)
[MapperPlotter3D, 10](#)
[mapperVertices, 11](#)

[perform_clustering, 11](#)
[perform_clustering_mlr3, 12](#)

[recursive_gaussian_split, 13](#)

[save_mapper_png, 13](#)
[simplicial_complex, 14](#)

[to_lsfi, 15](#)
[to_lsmi, 15](#)