MapperLib

Generated by Doxygen 1.12.0

Chapter 1

Deprecated List

Class MapperLib::SingleLinkage

use SLink_SingleLinkage instead

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 MapperLib Namespace Reference

Classes

- class CechComplex

    Class for generating Cech complexes from overlapping clusters.
- class CechComplexFactory

    Factory class for creating CechComplex objects.
- class Clusterer

    Abstract base class for clustering algorithms.
- class Complex

    Abstract base class for simplicial complexes.
- class ComplexFactory

    Abstract factory class for creating Complex objects.
- class CoordinatePlaneProjection

    projection of data to coordinate planes
- class DataCover

    Class for sectioning data into hypercubes.
- class DataCoverFactory

    Factory class creating DataCover objects.
- class Mapper

    class implementing the Mapper algorithm
- struct MapperCluster

    a cluster containign additional information
- class Projection

    Abstract base class for projection methods.
- struct Simplex

    Struct representing a simplex.
- class SingleLinkage

    primitive implementation of single linkage clustering
- class SLink_SingleLinkage

    efficient implementation of single linkage clustering

Typedefs

- using Cluster = std::vector<PointId>
- using ClusterAssignment = std::vector<Cluster>
- using Scalar = double
- using Vector = std::vector<Scalar>
- using Matrix = std::vector<std::vector<Scalar>>
- using PointId = size_t
- using Dimension = size_t
- using SimplexId = size_t
- using ClusterId = size_t
- using IntegerCubeId = size_t

Functions

- Scalar euclididan_distance (Vector const &vec1, Vector const &vec2)
- Scalar maximum_distance (Vector const &vec1, Vector const &vec2)
- bool check_data_equal_dimension (Matrix const &mat)
- Dimension get_data_dimension (Matrix const &mat)
- void print (Vector const &vec)
- PYBIND11_MODULE (MapperLib, mod)

6.1.1 Typedef Documentation

6.1.1.1 Cluster

```
using MapperLib::Cluster = std::vector<PointId>
```

6.1.1.2 ClusterAssignment

```
using MapperLib::ClusterAssignment = std::vector<Cluster>
```

6.1.1.3 ClusterId

```
using MapperLib::ClusterId = size_t
```

6.1.1.4 Dimension

```
using MapperLib::Dimension = size_t
```

6.1.1.5 IntegerCubeId

```
using MapperLib::IntegerCubeId = size_t
```

6.1.1.6 Matrix

```
using MapperLib::Matrix = std::vector<std::vector<Scalar>>
```

6.1.1.7 PointId

```
using MapperLib::PointId = size_t
```

6.1.1.8 Scalar

```
using MapperLib::Scalar = double
```

6.1.1.9 SimplexId

```
using MapperLib::SimplexId = size_t
```

6.1.1.10 Vector

```
using MapperLib::Vector = std::vector<Scalar>
```

6.1.2 Function Documentation

6.1.2.1 check_data_equal_dimension()

```
bool MapperLib::check_data_equal_dimension (
            Matrix const & mat)
```

Checks that all columns in the matrix have the same dimension, as Matrix is only an alias for `std::vector<std::vector<Scalar>>`

Parameters

mat	the matrix to check

Returns

true if matrix is valid, else false

6.1.2.2 euclididan_distance()

```
Scalar MapperLib::euclididan_distance (
            Vector const & vec1,
            Vector const & vec2)
```

Compute the euclidian distance between two data points, i.e. $\|v_1 - v_2\|_2$

Parameters

vec1	first data point
vec2	second data point

Returns

The distance between the vectors as a Scalar

6.1.2.3 get_data_dimension()

```
Dimension MapperLib::get_data_dimension (
            Matrix const & mat)
```

Get the dimension of the data points in a matrix

Parameters

mat	the data matrix

Returns

dimension of the first element in the matrix

Warning

This method does not assert that all data points are of equal length!

6.1.2.4 maximum_distance()

```
Scalar MapperLib::maximum_distance (
            Vector const & vec1,
            Vector const & vec2)
```

Compute the maximum distance between two data points, i.e. $\|v_1 - v_2\|_\infty$

Parameters

vec1	first data point
vec2	second data point

Returns

The distance between the vectors as a Scalar

6.1.2.5 print()

```
void MapperLib::print (
            Vector const & vec)
```

6.1.2.6 PYBIND11_MODULE()

```
MapperLib::PYBIND11_MODULE (
            MapperLib ,
            mod )
```

Chapter 7

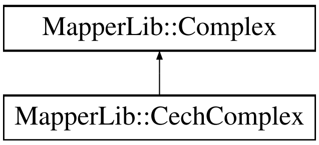
Class Documentation

7.1 MapperLib::CechComplex Class Reference

Class for generating Cech complexes from overlapping clusters.

```
#include <CechComplex.h>
```

Inheritance diagram for MapperLib::CechComplex:



Public Member Functions

- CechComplex (DataCover const &data_cover, Dimension max_dimension)

    Constructs a CechComplex object.
- std::vector< Simplex > generate (std::vector< MapperCluster > const &clusters) const override

    Generates a Cech complex on overlapping clusters.

Public Member Functions inherited from MapperLib::Complex

- virtual ∼Complex ()=default

7.1.1 Detailed Description

Class for generating Cech complexes from overlapping clusters.

This class implements the Complex interface and provides methods to generate a cech complex on the given clusters containing simplices up to a specified dimension

7.1.2 Constructor & Destructor Documentation

7.1.2.1 CechComplex()

```
MapperLib::CechComplex::CechComplex (
             DataCover const & data_cover,
             Dimension max_dimension)
```

Constructs a CechComplex object.

Initializes the CechComplex with a data cover and maximum dimension. The data cover is used to facilitate reasonably efficient enumeration of possible simplices.

Parameters

data_cover	A reference to a data_cover object.
max_dimension	The maximum dimension of simplices to generate.

7.1.3 Member Function Documentation

7.1.3.1 generate()

```
std::vector< Simplex > MapperLib::CechComplex::generate (
             std::vector< MapperCluster > const & clusters) const  [nodiscard], [override],
[virtual]
```

Generates a Cech complex on overlapping clusters.

This method iterates through the specified dimensions and generates simplices for each dimension. The process is multithreaded.

Warning

> The number of threads is currently hard coded to be <= 8. Change NUM_THREADS for more or less cores.

Parameters

clusters	A vector of clusters as generated by the mapper clusterer.

Returns

> A vector of simplices representing a simplicial complex.

Implements MapperLib::Complex.

The documentation for this class was generated from the following files:

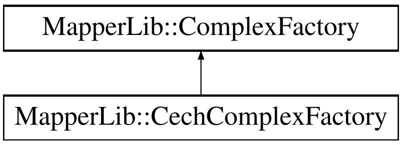
- CechComplex.h
- CechComplex.cpp

7.2 MapperLib::CechComplexFactory Class Reference

Factory class for creating CechComplex objects.

```
#include <CechComplex.h>
```

Inheritance diagram for MapperLib::CechComplexFactory:



Public Member Functions

- CechComplexFactory (Dimension max_dimension)

  Constructs a CechComplexFactory object.
- std::unique_ptr< Complex > create_complex (DataCover const &data_cover) const override

  Creates a CechComplex object.

Public Member Functions inherited from MapperLib::ComplexFactory

- virtual ~ComplexFactory ()=default

Static Public Member Functions

- static std::shared_ptr< ComplexFactory > make_shared (Dimension max_dimension)

  Creates a shared pointer to a CechComplexFactory instance.

7.2.1 Detailed Description

Factory class for creating CechComplex objects.

This class implements the ComplexFactory interface to create instances of CechComplex.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 CechComplexFactory()

```
MapperLib::CechComplexFactory::CechComplexFactory (
              Dimension max_dimension)  [explicit]
```

Constructs a CechComplexFactory object.

Initializes the factory with the specified maximum dimension.

Parameters

max_dimension	The maximum dimension for the CechComplex to be created.

7.2.3 Member Function Documentation

7.2.3.1 create_complex()

```
std::unique_ptr< Complex > MapperLib::CechComplexFactory::create_complex (
            DataCover const & data_cover) const  [nodiscard], [override], [virtual]
```

Creates a CechComplex object.

This method creates a new instance of CechComplex using the provided DataCover.

Parameters

data_cover	A reference to the DataCover object.

Returns

A unique pointer to the created CechComplex object.

Implements MapperLib::ComplexFactory.

7.2.3.2 make_shared()

```
std::shared_ptr< ComplexFactory > MapperLib::CechComplexFactory::make_shared (
            Dimension max_dimension)  [static], [nodiscard]
```

Creates a shared pointer to a CechComplexFactory instance.

This static method allows for easy memory management and object creation.

Parameters

max_dimension	The maximum dimension for the CechComplex.

Returns

A shared pointer to the newly created CechComplexFactory instance.

The documentation for this class was generated from the following files:

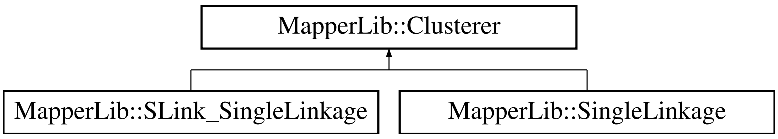
- CechComplex.h
- CechComplex.cpp

7.3 MapperLib::Clusterer Class Reference

Abstract base class for clustering algorithms.

```
#include <Clusterer.h>
```

Inheritance diagram for MapperLib::Clusterer:



Public Member Functions

- virtual ∼Clusterer ()=default
- virtual ClusterAssignment predict (Matrix const &data, std::vector< PointId > data_filter)=0

7.3.1 Detailed Description

Abstract base class for clustering algorithms.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 ∼Clusterer()

```
virtual MapperLib::Clusterer::∼Clusterer ()  [virtual], [default]
```

7.3.3 Member Function Documentation

7.3.3.1 predict()

```
virtual ClusterAssignment MapperLib::Clusterer::predict (
            Matrix const & data,
            std::vector< PointId > data_filter)  [pure virtual]
```

Implemented in MapperLib::SingleLinkage, and MapperLib::SLink_SingleLinkage.

The documentation for this class was generated from the following file:

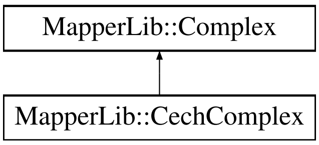
- Clusterer.h

7.4 MapperLib::Complex Class Reference

Abstract base class for simplicial complexes.

```
#include <CechComplex.h>
```

Inheritance diagram for MapperLib::Complex:



Public Member Functions

- virtual ~Complex ()=default
- virtual std::vector< Simplex > generate (std::vector< MapperCluster > const &clusters) const =0

  Generates simplices from the provided clusters.

7.4.1 Detailed Description

Abstract base class for simplicial complexes.

This class provides an interface for generating simplicial complices from clusters.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 ~Complex()

```
virtual MapperLib::Complex::~Complex ()  [virtual], [default]
```

7.4.3 Member Function Documentation

7.4.3.1 generate()

```
virtual std::vector< Simplex > MapperLib::Complex::generate (
            std::vector< MapperCluster > const & clusters) const  [nodiscard], [pure virtual]
```

Generates simplices from the provided clusters.

This method must be implemented by derived classes to generate a vector of simplices.

Parameters

clusters	A vector of clusters as generated by the mapper clusterer.

Returns

A vector of simplices representing a simplicial complex.

Implemented in MapperLib::CechComplex.

The documentation for this class was generated from the following file:

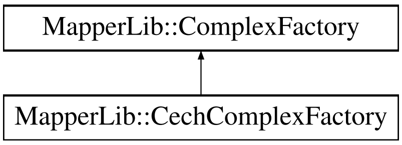
- CechComplex.h

7.5 MapperLib::ComplexFactory Class Reference

Abstract factory class for creating Complex objects.

```
#include <CechComplex.h>
```

Inheritance diagram for MapperLib::ComplexFactory:



Public Member Functions

- virtual ∼ComplexFactory ()=default
- virtual std::unique_ptr< Complex > create_complex (DataCover const &data_cover) const =0

  Creates a Complex object.

7.5.1 Detailed Description

Abstract factory class for creating Complex objects.

This class provides an interface for creating instances of Complex subclasses.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 ∼ComplexFactory()

```
virtual MapperLib::ComplexFactory::∼ComplexFactory ()  [virtual], [default]
```

7.5.3 Member Function Documentation

7.5.3.1 create_complex()

```
virtual std::unique_ptr< Complex > MapperLib::ComplexFactory::create_complex (
            DataCover const & data_cover) const  [nodiscard], [pure virtual]
```

Creates a Complex object.

This method must be implemented by derived classes to create a specific type of Complex.

Parameters

data_cover	A reference to the DataCover object.

Returns

A unique pointer to the created Complex object.

Implemented in MapperLib::CechComplexFactory.

The documentation for this class was generated from the following file:

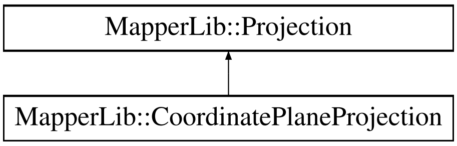
- CechComplex.h

7.6 MapperLib::CoordinatePlaneProjection Class Reference

projection of data to coordinate planes

```
#include <Projection.h>
```

Inheritance diagram for MapperLib::CoordinatePlaneProjection:



Public Member Functions

- CoordinatePlaneProjection (std::vector< Dimension > dimensions)
- Matrix project (Matrix const &data) const override

Public Member Functions inherited from MapperLib::Projection

- virtual ∼Projection ()=default

Static Public Member Functions

- static std::shared_ptr< Projection > make_shared (std::vector< Dimension > dimensions)

7.6.1 Detailed Description

projection of data to coordinate planes

This implements a very basic projection algorithm. The data is projected to a chosen coordinate hyperplane

7.6.2 Constructor & Destructor Documentation

7.6.2.1 CoordinatePlaneProjection()

```
MapperLib::CoordinatePlaneProjection::CoordinatePlaneProjection (
            std::vector< Dimension > dimensions)  [explicit]
```

Initialize a projection

Parameters

dimensions	the axis included in the coordinate hyperplane

7.6.3 Member Function Documentation

7.6.3.1 make_shared()

```
std::shared_ptr< Projection > MapperLib::CoordinatePlaneProjection::make_shared (
            std::vector< Dimension > dimensions)  [static], [nodiscard]
```

7.6.3.2 project()

```
Matrix MapperLib::CoordinatePlaneProjection::project (
            Matrix const & data) const  [nodiscard], [override], [virtual]
```

project the data to the coordinate hyperplane defined in the constructor

Parameters

data	the data to project

Returns

a Matrix of the dimension of the hyperplane containing all projected points

Implements MapperLib::Projection.

The documentation for this class was generated from the following files:

- Projection.h
- Projection.cpp

7.7 MapperLib::DataCover Class Reference

Class for sectioning data into hypercubes.

```
#include <DataCover.h>
```

Public Types

- using CubeId = std::vector<int>

Public Member Functions

- DataCover (size_t resolution, double perc_overlap, Matrix const &data, std::optional< Vector > minima=std::nullopt, std::optional< Vector > maxima=std::nullopt)
- DataCover (std::vector< size_t > resolution, double perc_overlap, Matrix const &data, std::optional< Vector > minima=std::nullopt, std::optional< Vector > maxima=std::nullopt)
- CubeId get_native_cube_id (Vector const &vec) const
- std::vector< PointId > get_points_in_cube (IntegerCubeId cube_id) const
- std::vector< IntegerCubeId > get_neighbor_cubes (IntegerCubeId integer_cube_id) const
- IntegerCubeId convert_to_integer_cube_id (CubeId const &cube_id) const
- CubeId convert_to_cube_id (IntegerCubeId integer_cube_id) const
- size_t get_total_num_cubes () const
- size_t get_num_cubes_in_dimension (Dimension dim) const
- bool is_vector_in_cube (Vector const &vec, CubeId const &cube_id) const

7.7.1 Detailed Description

Class for sectioning data into hypercubes.

This class provides the framework for splitting data into overlapping hypercubes for the Mapper algorithm.

7.7.2 Member Typedef Documentation

7.7.2.1 CubeId

```
using MapperLib::DataCover::CubeId = std::vector<int>
```

7.7.3 Constructor & Destructor Documentation

7.7.3.1 DataCover() [1/2]

```
MapperLib::DataCover::DataCover (
            size_t resolution,
            double perc_overlap,
            Matrix const & data,
            std::optional< Vector > minima = std::nullopt,
            std::optional< Vector > maxima = std::nullopt)
```

Create a sectioning of the data space into hypercubes.

Parameters

resolution	Integer variable declaring how many intervals there should be along each axis
perc_overlap	Floating point variable declaring how much the hypercubes should overlap, has to be nonnegative and <= 0.5.
data	A reference to the data points, to interpolate the minima and maxima
minima	A vector defining the minima that should be used in each dimension
maxima	A vector defining the maxima that should be used in each dimension

7.7.3.2 DataCover() [2/2]

```
MapperLib::DataCover::DataCover (
            std::vector< size_t > resolution,
            double perc_overlap,
            Matrix const & data,
            std::optional< Vector > minima = std::nullopt,
            std::optional< Vector > maxima = std::nullopt)
```

Create a sectioning of the data space into hypercubes.

Parameters

resolution	vector defining the number of intervals along each axis
perc_overlap	Floating point variable declaring how much the hypercubes should overlap, has to be nonnegative and <= 0.5.
data	A reference to the data points, to interpolate the minima and maxima
minima	A vector defining the minima that should be used in each dimension
maxima	A vector defining the maxima that should be used in each dimension

7.7.4 Member Function Documentation

7.7.4.1 convert_to_cube_id()

```
DataCover::CubeId MapperLib::DataCover::convert_to_cube_id (
            IntegerCubeId integer_cube_id) const
```

Convert the numerical integer cube id of a cube to its coordinate in the cube grid

Parameters

integer_cube↩ _id	the id to convert

Returns

a cube id, i.e. the coordinates of the cube in the grid.

7.7.4.2 convert_to_integer_cube_id()

```
IntegerCubeId MapperLib::DataCover::convert_to_integer_cube_id (
            CubeId const & cube_id) const
```

Convert the coordinates of a cube (i.e. the cube id) to the integer id

Parameters

cube↩ _id	the id to convert

Returns

an integer uniquely representing the cube

7.7.4.3 get_native_cube_id()

```
DataCover::CubeId MapperLib::DataCover::get_native_cube_id (
            Vector const & vec) const [nodiscard]
```

Compute the cube a vector lies in disregarding the overlaps

Parameters

vec	data point

Returns

the id of the native cube of vec

7.7.4.4 get_neighbor_cubes()

```
std::vector< IntegerCubeId > MapperLib::DataCover::get_neighbor_cubes (
            IntegerCubeId integer_cube_id) const [nodiscard]
```

Get all cubes neighboring the given cube

Parameters

integer_cube↩ _id	

Returns

vector of cube ids

7.7.4.5 get_num_cubes_in_dimension()

```
size_t MapperLib::DataCover::get_num_cubes_in_dimension (
            Dimension dim) const
```

7.7.4.6 get_points_in_cube()

```
std::vector< PointId > MapperLib::DataCover::get_points_in_cube (
            IntegerCubeId cube_id) const  [nodiscard]
```

Get the indices of the points in a given cube.

Parameters

cube↩ _id	

Returns

Vector of points in the cube

7.7.4.7 get_total_num_cubes()

```
size_t MapperLib::DataCover::get_total_num_cubes () const
```

get the number of all cubes in the cover

Returns

number of cubes

7.7.4.8 is_vector_in_cube()

```
bool MapperLib::DataCover::is_vector_in_cube (
            Vector const & vec,
            CubeId const & cube_id) const
```

Determine whether a data point is inside a cube, taking overlaps into account

Parameters

vec	the data point
cube↩ _id	the cube to test against

Returns

true if the cube contains the point, else false

The documentation for this class was generated from the following files:

- DataCover.h
- DataCover.cpp

7.8 MapperLib::DataCoverFactory Class Reference

Factory class creating DataCover objects.

```
#include <DataCover.h>
```

Public Member Functions

- DataCoverFactory (size_t resolution, double perc_overlap, std::optional< Vector > minima=std::nullopt, std↩ ::optional< Vector > maxima=std::nullopt)
- DataCoverFactory (std::vector< size_t > resolution, double perc_overlap, std::optional< Vector > minima=std::nullopt, std::optional< Vector > maxima=std::nullopt)
- std::unique_ptr< DataCover > create_data_cover (Matrix const &data) const

Static Public Member Functions

- static std::shared_ptr< DataCoverFactory > make_shared (size_t resolution, double perc_overlap, std↩ ::optional< Vector > minima=std::nullopt, std::optional< Vector > maxima=std::nullopt)
- static std::shared_ptr< DataCoverFactory > make_shared (std::vector< size_t > resolution, double perc_↩ overlap, std::optional< Vector > minima=std::nullopt, std::optional< Vector > maxima=std::nullopt)

7.8.1 Detailed Description

Factory class creating DataCover objects.

used to create the object within the map method of Mapper

7.8.2 Constructor & Destructor Documentation

7.8.2.1 DataCoverFactory() [1/2]

```
MapperLib::DataCoverFactory::DataCoverFactory (
              size_t resolution,
              double perc_overlap,
              std::optional< Vector > minima = std::nullopt,
              std::optional< Vector > maxima = std::nullopt)
```

Constructor for DataCoverFactory with single resolution

Parameters

resolution	Single resolution value for all dimensions
perc_overlap	Percentage of overlap between hypercubes
minima	Optional minima for each dimension
maxima	Optional maxima for each dimension

7.8.2.2 DataCoverFactory() [2/2]

```
MapperLib::DataCoverFactory::DataCoverFactory (
            std::vector< size_t > resolution,
            double perc_overlap,
            std::optional< Vector > minima = std::nullopt,
            std::optional< Vector > maxima = std::nullopt)
```

Constructor for DataCoverFactory with multiple resolutions

Parameters

resolution	Vector of resolution values for each dimension
perc_overlap	Percentage of overlap between hypercubes
minima	Optional minima for each dimension
maxima	Optional maxima for each dimension

7.8.3 Member Function Documentation

7.8.3.1 create_data_cover()

```
std::unique_ptr< MapperLib::DataCover > MapperLib::DataCoverFactory::create_data_cover (
            Matrix const & data) const [nodiscard]
```

Create a DataCover object

Parameters

data	A reference to the data points

Returns

Unique pointer to a new DataCover instance

7.8.3.2 make_shared() [1/2]

```
std::shared_ptr< MapperLib::DataCoverFactory > MapperLib::DataCoverFactory::make_shared (
            size_t resolution,
            double perc_overlap,
            std::optional< Vector > minima = std::nullopt,
            std::optional< Vector > maxima = std::nullopt) [static], [nodiscard]
```

Static factory method to create a shared pointer to DataCoverFactory

Parameters

resolution	Single resolution value for all dimensions
perc_overlap	Percentage of overlap between hypercubes
minima	Optional minima for each dimension
maxima	Optional maxima for each dimension

Returns

Shared pointer to a new DataCoverFactory instance

7.8.3.3 make_shared() [2/2]

```
std::shared_ptr< MapperLib::DataCoverFactory > MapperLib::DataCoverFactory::make_shared (
             std::vector< size_t > resolution,
             double perc_overlap,
             std::optional< Vector > minima = std::nullopt,
             std::optional< Vector > maxima = std::nullopt)  [static], [nodiscard]
```

Static factory method to create a shared pointer to DataCoverFactory

Parameters

resolution	Vector of resolution values for each dimension
perc_overlap	Percentage of overlap between hypercubes
minima	Optional minima for each dimension
maxima	Optional maxima for each dimension

Returns

Shared pointer to a new DataCoverFactory instance

The documentation for this class was generated from the following files:

- DataCover.h
- DataCover.cpp

7.9 MapperLib::Mapper Class Reference

class implementing the Mapper algorithm

```
#include <Mapper.h>
```

Public Member Functions

- Mapper (std::shared_ptr< DataCoverFactory > data_cover_factory, std::shared_ptr< ComplexFactory > complex_factory, std::shared_ptr< Clusterer > clusterer, std::shared_ptr< Projection > projection)
- std::vector< Simplex > map (Matrix const &data)

7.9.1 Detailed Description

class implementing the Mapper algorithm

Mapper is a data visualization algorithm. It takes a point cloud, projects it to a lower dimension, covers the remaining space in overlapping hypercubes and clusters the original data in each of these hypercubes. Overlapping clusters are now linked by simplices in accordance to the chosen complex. The resulting simplicial complex is the output of the algorithm.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 Mapper()

```
MapperLib::Mapper::Mapper (
            std::shared_ptr< DataCoverFactory > data_cover_factory,
            std::shared_ptr< ComplexFactory > complex_factory,
            std::shared_ptr< Clusterer > clusterer,
            std::shared_ptr< Projection > projection)
```

Create a Mapper algorithm object

Parameters

data_cover_factory	factory creating the data cover to use
complex_factory	factory creating the complex to use
clusterer	the clustering algorithm to use
projection	the projection to use.

7.9.3 Member Function Documentation

7.9.3.1 map()

```
std::vector< Simplex > MapperLib::Mapper::map (
            Matrix const & data)  [nodiscard]
```

The main method of the Mapper algorithm

Parameters

data	data to apply the mapper algorithm to

Returns

the simplicial complex generated by the Mapper algorithm

The documentation for this class was generated from the following files:

- Mapper.h
- Mapper.cpp

7.10 MapperLib::MapperCluster Struct Reference

a cluster containign additional information

```
#include <typedefs.h>
```

Public Attributes

- std::vector< PointId > points
- ClusterId cluster_id
- IntegerCubeId integer_cube_id

7.10.1 Detailed Description

a cluster containign additional information

This cluster is used in the mapper algorithm to decrease lookup times.

7.10.2 Member Data Documentation

7.10.2.1 cluster_id

```
ClusterId MapperLib::MapperCluster::cluster_id
```

7.10.2.2 integer_cube_id

```
IntegerCubeId MapperLib::MapperCluster::integer_cube_id
```

7.10.2.3 points

```
std::vector<PointId> MapperLib::MapperCluster::points
```

The documentation for this struct was generated from the following file:

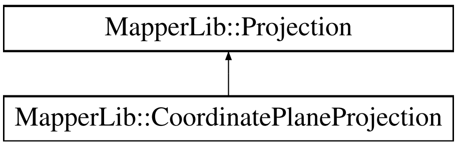
- typedefs.h

7.11 MapperLib::Projection Class Reference

Abstract base class for projection methods.

```
#include <Projection.h>
```

Inheritance diagram for MapperLib::Projection:



Public Member Functions

- virtual ∼Projection ()=default
- virtual Matrix project (Matrix const &data) const =0

7.11.1 Detailed Description

Abstract base class for projection methods.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 ∼Projection()

```
virtual MapperLib::Projection::∼Projection ()  [virtual], [default]
```

7.11.3 Member Function Documentation

7.11.3.1 project()

```
virtual Matrix MapperLib::Projection::project (
            Matrix const & data) const  [nodiscard], [pure virtual]
```

Implemented in MapperLib::CoordinatePlaneProjection.

The documentation for this class was generated from the following file:

- Projection.h

7.12 MapperLib::Simplex Struct Reference

Struct representing a simplex.

```
#include <typedefs.h>
```

Public Member Functions

- std::vector< PointId > get_points ()
- Dimension dimension () const
- size_t num_nodes () const
- PointId operator[ ] (size_t index) const

Public Attributes

- std::vector< PointId > points

  The points making up the simplex.

7.12.1 Detailed Description

Struct representing a simplex.

A simplex consisting of node ids instead of actual points to save space.

7.12.2 Member Function Documentation

7.12.2.1 dimension()

```
Dimension MapperLib::Simplex::dimension () const  [inline], [nodiscard]
```

7.12.2.2 get_points()

```
std::vector< PointId > MapperLib::Simplex::get_points ()  [inline], [nodiscard]
```

7.12.2.3 num_nodes()

```
size_t MapperLib::Simplex::num_nodes () const  [inline], [nodiscard]
```

7.12.2.4 operator[]()

```
PointId MapperLib::Simplex::operator[] (
             size_t index) const  [inline], [nodiscard]
```

7.12.3 Member Data Documentation

7.12.3.1 points

```
std::vector<PointId> MapperLib::Simplex::points
```

The points making up the simplex.

The documentation for this struct was generated from the following file:

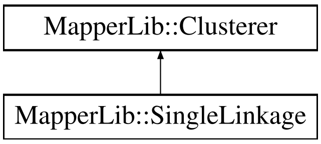
- typedefs.h

7.13 MapperLib::SingleLinkage Class Reference

primitive implementation of single linkage clustering

```
#include <SingleLinkage.h>
```

Inheritance diagram for MapperLib::SingleLinkage:



Public Member Functions

- SingleLinkage (std::optional< int > num_clusters, std::optional< Scalar > distance_threshold)
- ClusterAssignment predict (Matrix const &data, std::vector< PointId > data_filter) override

Public Member Functions inherited from MapperLib::Clusterer

- virtual ∼Clusterer ()=default

Static Public Member Functions

- static std::shared_ptr< Clusterer > make_shared (std::optional< int > num_clusters, std::optional< Scalar > distance_threshold)

7.13.1 Detailed Description

primitive implementation of single linkage clustering

Deprecated use SLink_SingleLinkage instead

7.13.2 Constructor & Destructor Documentation

7.13.2.1 SingleLinkage()

```
MapperLib::SingleLinkage::SingleLinkage (
              std::optional< int > num_clusters,
              std::optional< Scalar > distance_threshold)
```

7.13.3 Member Function Documentation

7.13.3.1 make_shared()

```
std::shared_ptr< Clusterer > MapperLib::SingleLinkage::make_shared (
              std::optional< int > num_clusters,
              std::optional< Scalar > distance_threshold)  [static], [nodiscard]
```

7.13.3.2 predict()

```
ClusterAssignment MapperLib::SingleLinkage::predict (
              Matrix const & data,
              std::vector< PointId > data_filter)  [override], [virtual]
```

Implements MapperLib::Clusterer.

The documentation for this class was generated from the following files:

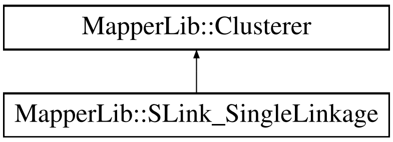
- SingleLinkage.h
- SingleLinkage.cpp

7.14 MapperLib::SLink_SingleLinkage Class Reference

efficient implementation of single linkage clustering

```
#include <SLink_SingleLinkage.h>
```

Inheritance diagram for MapperLib::SLink_SingleLinkage:



Public Member Functions

- SLink_SingleLinkage (std::optional< int > num_clusters, std::optional< Scalar > distance_threshold)

  Constructs an SLink_SingleLinkage object.
- ClusterAssignment predict (Matrix const &data, std::vector< PointId > data_filter) override

  Predicts cluster assignments for the given dataset.

Public Member Functions inherited from MapperLib::Clusterer

- virtual ∼Clusterer ()=default

Static Public Member Functions

- static std::shared_ptr< Clusterer > make_shared (std::optional< int > num_clusters, std::optional< Scalar > distance_threshold)

7.14.1 Detailed Description

efficient implementation of single linkage clustering

A class implementing the SLINK algorithm for single linkage agglomorative clustering. The main algorithm is due to R. Sibson: "SLINK: An optimally efficient algorithm for the single-link cluster method".

7.14.2 Constructor & Destructor Documentation

7.14.2.1 SLink_SingleLinkage()

```
MapperLib::SLink_SingleLinkage::SLink_SingleLinkage (
            std::optional< int > num_clusters,
            std::optional< Scalar > distance_threshold)
```

Constructs an SLink_SingleLinkage object.

Initializes the clustering parameters with the provided optional values. At least one of num_clusters or distance_threshold must be specified.

Parameters

num_clusters	Optional integer specifying the desired number of clusters.
distance_threshold	Optional Scalar value for the maximum distance threshold.

7.14.3 Member Function Documentation

7.14.3.1 make_shared()

```
std::shared_ptr< Clusterer > MapperLib::SLink_SingleLinkage::make_shared (
            std::optional< int > num_clusters,
            std::optional< Scalar > distance_threshold)  [static], [nodiscard]
```

7.14.3.2 predict()

```
ClusterAssignment MapperLib::SLink_SingleLinkage::predict (
            Matrix const & data,
            std::vector< PointId > data_filter)  [override], [virtual]
```

Predicts cluster assignments for the given dataset.

This method performs the single linkage clustering algorithm on the provided data and returns the cluster assignments.

Parameters

data	A constant reference to a Matrix containing the data points to be clustered.
data_filter	A vector of ids that filtering the data points for prediction.

Returns

ClusterAssignment The resulting cluster assignments for the input data.

Implements MapperLib::Clusterer.

The documentation for this class was generated from the following files:

- SLink_SingleLinkage.h
- SLink_SingleLinkage.cpp

Chapter 8

File Documentation

8.1 CechComplex.cpp File Reference

```
#include "CechComplex.h"
#include <cassert>
#include <ranges>
#include <generator>
#include <algorithm>
#include <mutex>
#include <thread>
#include "DataCover.h"
```

Namespaces

- namespace MapperLib

8.2 CechComplex.h File Reference

```
#include <vector>
#include <generator>
#include <memory>
#include "typedefs.h"
```

Classes

- class MapperLib::Complex

  Abstract base class for simplicial complexes.
- class MapperLib::CechComplex

  Class for generating Cech complexes from overlapping clusters.
- class MapperLib::ComplexFactory

  Abstract factory class for creating Complex objects.
- class MapperLib::CechComplexFactory

  Factory class for creating CechComplex objects.

Namespaces

- namespace MapperLib

8.3 CechComplex.h

Go to the documentation of this file.

```
//
// Created by jgier on 29.06.2024.
//

#ifndef MAPPER_CECHCOMPLEX_H
#define MAPPER_CECHCOMPLEX_H

#include <vector>
#include <generator>
#include <memory>

#include "typedefs.h"

namespace MapperLib {

class DataCover;

class Complex {
public:
    virtual ~Complex() = default;

    [[nodiscard]] virtual std::vector<Simplex> generate(std::vector<MapperCluster> const &clusters) const = 0;
};

class CechComplex final : public Complex {
    static constexpr size_t NUM_THREADS = 8;
public:
    CechComplex(DataCover const& data_cover, Dimension max_dimension);

    [[nodiscard]] std::vector<Simplex> generate(std::vector<MapperCluster> const& clusters) const override;

private:
    using Iterator = std::vector<MapperCluster>::const_iterator;
    using ClustersByCube = std::vector<std::vector<MapperCluster const*»;

    class SimplexComputer {
        Iterator _begin;
        Iterator _end;
        ClustersByCube& _clusters_by_cube;
        Dimension const _dim;
        DataCover const& _data_cover;
        std::vector<Simplex> _own_result;
        std::vector<Simplex> &_result;
        std::mutex &_mutex;
        int _id;

    public:
        static inline int id_counter = 0;

        SimplexComputer(Iterator begin, Iterator end, ClustersByCube &clusters_by_cube, Dimension dim,
                        DataCover const &data_cover, std::vector<Simplex> &result, std::mutex &mutex);

        void compute();
    };

    struct ComputerWrapper {
        SimplexComputer & computer;

        void operator()() const { computer.compute(); }
    };

    [[nodiscard]] std::vector<Simplex> generate_k_simplices(std::vector<MapperCluster> const& clusters, Dimension k) const;

    static std::generator<std::vector<size_t» generate_k_subsets_of_range(size_t index_max, size_t k);

    static bool check_cluster_intersection(std::vector<MapperCluster const*> const& all_clusters, std::vector<size_t> const& relevant_indices);

    static std::vector<size_t> get_vector_intersection(std::vector<size_t> vec_1, std::vector<size_t> vec_2); //ToDo: This should probably be in a helper file

    DataCover const &_data_cover;
    Dimension _max_dimension;
};

class ComplexFactory {
public:
    virtual ~ComplexFactory() = default;

    [[nodiscard]] virtual std::unique_ptr<Complex> create_complex(DataCover const& data_cover) const = 0;
};

class CechComplexFactory final : public ComplexFactory {
public:
    explicit CechComplexFactory(Dimension max_dimension);

    [[nodiscard]] static std::shared_ptr<ComplexFactory> make_shared(Dimension max_dimension);

    [[nodiscard]] std::unique_ptr<Complex> create_complex(DataCover const &data_cover) const override;

private:
    Dimension _max_dimension;
};

} // namespace MapperLib

#endif // MAPPER_CECHCOMPLEX_H
```

8.4 Clusterer.h File Reference

```
#include "typedefs.h"
```

Classes

- class MapperLib::Clusterer

  Abstract base class for clustering algorithms.

Namespaces

- namespace MapperLib

Typedefs

- using MapperLib::Cluster = std::vector< PointId >
- using MapperLib::ClusterAssignment = std::vector< Cluster >

8.5 Clusterer.h

Go to the documentation of this file.

```
//
// Created by jakob on 23.06.24.
//

#ifndef CLUSTERER_H
#define CLUSTERER_H

#include "typedefs.h"


namespace MapperLib {
```

```

using Cluster = std::vector<PointId>;
using ClusterAssignment = std::vector<Cluster>;

class Clusterer {
public:
    virtual ~Clusterer() = default;
    virtual ClusterAssignment predict(Matrix const &data, std::vector<PointId> data_filter) = 0;
private:
};

} // cluster



#endif //CLUSTERER_H
```

8.6 DataCover.cpp File Reference

```
#include "DataCover.h"
#include <cassert>
#include <algorithm>
#include <cmath>
#include <utility>
```

Namespaces

- namespace MapperLib

Functions

- std::ostream & operator<< (std::ostream &stream, MapperLib::DataCover::CubeId const &vec)

8.6.1 Function Documentation

8.6.1.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & stream,
            MapperLib::DataCover::CubeId const & vec)
```

8.7 DataCover.h File Reference

```
#include <memory>
#include <ostream>
#include "SingleLinkage.h"
#include "typedefs.h"
```

Classes

- class MapperLib::DataCover

  Class for sectioning data into hypercubes.
- class MapperLib::DataCoverFactory

  Factory class creating DataCover objects.

Namespaces

- namespace MapperLib

Functions

- std::ostream & operator<< (std::ostream &stream, MapperLib::DataCover::CubeId const &vec)

8.7.1 Function Documentation

8.7.1.1 operator<<()

```
std::ostream & operator<< (
              std::ostream & stream,
              MapperLib::DataCover::CubeId const & vec)
```

8.8 DataCover.h

Go to the documentation of this file.

```
//
// Created by jakob on 24.06.24.
//

#ifndef DATACOVER_H
#define DATACOVER_H

#include <memory>
#include <ostream>

#include "SingleLinkage.h"
#include "typedefs.h"

namespace MapperLib {
class DataCover {
public:
    using CubeId = std::vector<int>;

    DataCover(
        size_t resolution,
        double perc_overlap,
        Matrix const& data,
        std::optional<Vector> minima = std::nullopt,
        std::optional<Vector> maxima = std::nullopt
    );

    DataCover(
        std::vector<size_t> resolution,
        double perc_overlap,
        Matrix const& data,
        std::optional<Vector> minima = std::nullopt,
        std::optional<Vector> maxima = std::nullopt
    );

    [[nodiscard]] CubeId get_native_cube_id(Vector const& vec) const;

    [[nodiscard]] std::vector<PointId> get_points_in_cube(IntegerCubeId cube_id) const;

    [[nodiscard]] std::vector<IntegerCubeId> get_neighbor_cubes(IntegerCubeId integer_cube_id) const;

    IntegerCubeId convert_to_integer_cube_id(CubeId const& cube_id) const;

    CubeId convert_to_cube_id(IntegerCubeId integer_cube_id) const;

    size_t get_total_num_cubes() const;

    size_t get_num_cubes_in_dimension(Dimension dim) const;

    bool is_vector_in_cube(Vector const& vec, CubeId const& cube_id) const;

private:
    void initialize_cube_cache() const;

    std::vector<CubeId> get_parent_cubes(Vector const& v) const;

    [[nodiscard]] Scalar get_data_min_in_dimension(Dimension dimension) const;
    [[nodiscard]] Scalar get_data_max_in_dimension(Dimension dimension) const;

    void initialize_minima_from_data();
    void initialize_maxima_from_data();

    [[nodiscard]] Vector get_cube_center(CubeId const& cube_id) const;
    [[nodiscard]] std::vector<CubeId> get_neighbor_cubes(CubeId const& cube_id) const;


    std::vector<size_t> _resolution;
    double const _perc_overlap;
    Matrix const& _data;
    size_t const _data_dimension;
    Vector _minima;
    Vector _maxima;

    mutable std::optional<std::vector<std::vector<PointId»> _cube_cache;

};

class DataCoverFactory
{
public:
    DataCoverFactory(
        size_t resolution,
        double perc_overlap,
        std::optional<Vector> minima = std::nullopt,
        std::optional<Vector> maxima = std::nullopt
    );

    DataCoverFactory(
        std::vector<size_t> resolution,
        double perc_overlap,
        std::optional<Vector> minima = std::nullopt,
        std::optional<Vector> maxima = std::nullopt
    );

    [[nodiscard]] static std::shared_ptr<DataCoverFactory> make_shared(
        size_t resolution,
        double perc_overlap,
        std::optional<Vector> minima = std::nullopt,
        std::optional<Vector> maxima = std::nullopt
    );

    [[nodiscard]] static std::shared_ptr<DataCoverFactory> make_shared(
        std::vector<size_t> resolution,
        double perc_overlap,
        std::optional<Vector> minima = std::nullopt,
        std::optional<Vector> maxima = std::nullopt
    );

    [[nodiscard]] std::unique_ptr<DataCover> create_data_cover(Matrix const& data) const;

private:
    std::vector<size_t> _resolution;
    std::optional<size_t> _single_resolution;
    double _perc_overlap;
    std::optional<Vector> _minima;
    std::optional<Vector> _maxima;
};



} // Mapper
std::ostream& operator«(std::ostream& stream, MapperLib::DataCover::CubeId const& vec);



```

```
00223 #endif //DATACOVER_H
```

8.9 LinalgHelpers.cpp File Reference

```
#include <cassert>
#include <cmath>
#include <algorithm>
#include "LinalgHelpers.h"
#include "DataCover.h"
#include <ostream>
```

Namespaces

- namespace MapperLib

Functions

- Scalar MapperLib::euclididan_distance (Vector const &vec1, Vector const &vec2)
- Scalar MapperLib::maximum_distance (Vector const &vec1, Vector const &vec2)
- bool MapperLib::check_data_equal_dimension (Matrix const &mat)
- Dimension MapperLib::get_data_dimension (Matrix const &mat)
- std::ostream & operator<< (std::ostream &os, MapperLib::Vector const &vec)
- std::ostream & operator<< (std::ostream &os, MapperLib::Matrix const &mat)

8.9.1 Function Documentation

8.9.1.1 operator<<() [1/2]

```
std::ostream & operator<< (
            std::ostream & os,
            MapperLib::Matrix const & mat)
```

8.9.1.2 operator<<() [2/2]

```
std::ostream & operator<< (
            std::ostream & os,
            MapperLib::Vector const & vec)
```

8.10 LinalgHelpers.h File Reference

```
#include <ostream>
#include "typedefs.h"
```

Namespaces

- namespace MapperLib

Functions

- Scalar MapperLib::euclididan_distance (Vector const &vec1, Vector const &vec2)
- Scalar MapperLib::maximum_distance (Vector const &vec1, Vector const &vec2)
- bool MapperLib::check_data_equal_dimension (Matrix const &mat)
- Dimension MapperLib::get_data_dimension (Matrix const &mat)
- void MapperLib::print (Vector const &vec)
- std::ostream & operator<< (std::ostream &os, MapperLib::Vector const &vec)
- std::ostream & operator<< (std::ostream &os, MapperLib::Matrix const &mat)

8.10.1 Function Documentation

8.10.1.1 operator<<() [1/2]

```
std::ostream & operator<< (
              std::ostream & os,
              MapperLib::Matrix const & mat)
```

8.10.1.2 operator<<() [2/2]

```
std::ostream & operator<< (
              std::ostream & os,
              MapperLib::Vector const & vec)
```

8.11 LinalgHelpers.h

Go to the documentation of this file.

```
//
// Created by jakob on 23.06.24.
//

#ifndef LINALGHELPERS_H
#define LINALGHELPERS_H
#include <ostream>

#include "typedefs.h"

namespace MapperLib {
Scalar euclididan_distance(Vector const& vec1, Vector const& vec2);

Scalar maximum_distance(Vector const& vec1, Vector const& vec2);

bool check_data_equal_dimension(Matrix const& mat);

Dimension get_data_dimension(Matrix const& mat);

void print(Vector const& vec);
} // Helper

std::ostream& operator«(std::ostream& os, MapperLib::Vector const& vec);
std::ostream& operator«(std::ostream& os, MapperLib::Matrix const& mat);

#endif //LINALGHELPERS_H
```

8.12 main.cpp File Reference

```
#include <iostream>
#include <memory>
#include "CechComplex.h"
#include "SingleLinkage.h"
#include "DataCover.h"
#include "Mapper.h"
#include "typedefs.h"
#include "Projection.h"
#include "SLink_SingleLinkage.h"
```

Functions

- int main ()

8.12.1 Function Documentation

8.12.1.1 main()

```
int main ()
```

8.13 Mapper.cpp File Reference

```
#include "Mapper.h"
#include <cassert>
#include <utility>
#include "DataCover.h"
#include "CechComplex.h"
#include "Clusterer.h"
#include "Projection.h"
```

Namespaces

- namespace MapperLib

8.14 Mapper.h File Reference

```
#include <vector>
#include <memory>
#include "typedefs.h"
#include "DataCover.h"
```

Classes

- class MapperLib::Mapper

  class implementing the Mapper algorithm

Namespaces

- namespace MapperLib

8.15 Mapper.h

Go to the documentation of this file.

```
//
// Created by jakob on 26.06.24.
//

#ifndef MAPPER_H
#define MAPPER_H
#include <vector>
#include <memory>
#include "typedefs.h"
#include "DataCover.h"



namespace MapperLib {
class ComplexFactory;
class Complex;
class Clusterer;
class Projection;

class Mapper {
public:
    Mapper(
        std::shared_ptr<DataCoverFactory> data_cover_factory,
        std::shared_ptr<ComplexFactory> complex_factory,
        std::shared_ptr<Clusterer> clusterer,
        std::shared_ptr<Projection> projection
    );

    [[nodiscard]] std::vector<Simplex> map(Matrix const& data) ;

private:
    std::shared_ptr<DataCoverFactory> _data_cover_factory;
    std::shared_ptr<ComplexFactory> _complex_factory;
    std::shared_ptr<Clusterer> _clusterer;
    std::shared_ptr<Projection> _projection;
    std::unique_ptr<DataCover> _data_cover;
    std::unique_ptr<Complex> _complex;
};

} // Mapper

#endif //MAPPER_H
```

8.16 Projection.cpp File Reference

```
#include "Projection.h"
```

Namespaces

- namespace MapperLib

8.17 Projection.h File Reference

```
#include <memory>
#include "typedefs.h"
```

Classes

- class MapperLib::Projection

  Abstract base class for projection methods.
- class MapperLib::CoordinatePlaneProjection

  projection of data to coordinate planes

Namespaces

- namespace MapperLib

8.18 Projection.h

Go to the documentation of this file.

```
//
// Created by jgier on 27.06.2024.
//

#ifndef MAPPER_PROJECTION_H
#define MAPPER_PROJECTION_H
#include <memory>

#include "typedefs.h"

namespace MapperLib{
class Projection {
public:
    virtual ~Projection() = default;
    [[nodiscard]] virtual Matrix project(Matrix const& data) const = 0;
};

class CoordinatePlaneProjection final : public  Projection{
public:
    explicit CoordinatePlaneProjection(std::vector<Dimension> dimensions);
    [[nodiscard]] static std::shared_ptr<Projection> make_shared(std::vector<Dimension> dimensions);

    [[nodiscard]] Matrix project (Matrix const& data) const override;
private:
    std::vector<Dimension> _dimensions;
};

} // Mapper

#endif //MAPPER_PROJECTION_H
```

8.19 PythonModule.cpp File Reference

```
#include <pybind11/pybind11.h>
#include <pybind11/stl.h>
#include "CechComplex.h"
#include "SingleLinkage.h"
#include "DataCover.h"
#include "Mapper.h"
#include "typedefs.h"
#include "Projection.h"
#include "SLink_SingleLinkage.h"
```

Namespaces

- namespace MapperLib

Functions

- MapperLib::PYBIND11_MODULE (MapperLib, mod)

8.20 SingleLinkage.cpp File Reference

```
#include "SingleLinkage.h"
#include <cassert>
#include <algorithm>
#include <iostream>
#include <iomanip>
```

Namespaces

- namespace MapperLib

8.21 SingleLinkage.h File Reference

```
#include <limits>
#include <memory>
#include <optional>
#include "Clusterer.h"
#include "LinalgHelpers.h"
```

Classes

- class MapperLib::SingleLinkage

  primitive implementation of single linkage clustering

Namespaces

- namespace MapperLib

8.22 SingleLinkage.h

Go to the documentation of this file.

```
//
// Created by jakob on 14.06.24.
//

#ifndef SINGLELINKAGE_H
#define SINGLELINKAGE_H
#include <limits>
#include <memory>
#include <optional>

#include "Clusterer.h"
#include "LinalgHelpers.h"

namespace MapperLib {
class SingleLinkage : public Clusterer {
public:

    SingleLinkage(std::optional<int> num_clusters, std::optional<Scalar> distance_threshold);
    [[nodiscard]] static std::shared_ptr<Clusterer> make_shared(std::optional<int> num_clusters, std::optional<Scalar> distance_threshold);

    ClusterAssignment predict(Matrix const &data, std::vector<PointId> data_filter) override;

private:
    [[nodiscard]] static Scalar min_distance(
        Matrix const& distances,
        std::vector<PointId> const& cluster1,
        std::vector<PointId> const& cluster2
    );

    std::optional<size_t> _num_clusters;
    std::optional<Scalar> _distance_threshold;

};

} // Cluster

#endif //SINGLELINKAGE_H
```

8.23 SLink_SingleLinkage.cpp File Reference

```
#include "SLink_SingleLinkage.h"
#include <cassert>
#include <bits/ranges_algo.h>
#include "LinalgHelpers.h"
```

Namespaces

- namespace MapperLib

8.24 SLink_SingleLinkage.h File Reference

```
#include <memory>
#include "Clusterer.h"
```

Classes

- class MapperLib::SLink_SingleLinkage

  efficient implementation of single linkage clustering

Namespaces

- namespace MapperLib

8.25 SLink_SingleLinkage.h

Go to the documentation of this file.

```
//
// Created by jakob on 10.07.24.
//

#ifndef SLINK_SINGLELINKAGE_H
#define SLINK_SINGLELINKAGE_H

#include <memory>

#include "Clusterer.h"

namespace MapperLib {
class SLink_SingleLinkage : public Clusterer{
public:

    SLink_SingleLinkage(std::optional<int> num_clusters, std::optional<Scalar> distance_threshold);
    [[nodiscard]] static std::shared_ptr<Clusterer> make_shared(std::optional<int> num_clusters, std::optional<Scalar> distance_threshold);

    ClusterAssignment predict(Matrix const &data, std::vector<PointId> data_filter) override;
private:
    std::optional<size_t> _num_clusters;
    std::optional<Scalar> _distance_threshold;
};

} // Mapper

#endif //SLINK_SINGLELINKAGE_H
```

8.26 typedefs.h File Reference

```
#include <vector>
#include <iostream>
```

Classes

- struct MapperLib::Simplex

  Struct representing a simplex.
- struct MapperLib::MapperCluster

  a cluster containign additional information

Namespaces

- namespace MapperLib

Typedefs

- using MapperLib::Scalar = double
- using MapperLib::Vector = std::vector< Scalar >
- using MapperLib::Matrix = std::vector< std::vector< Scalar > >
- using MapperLib::PointId = size_t
- using MapperLib::Dimension = size_t
- using MapperLib::SimplexId = size_t
- using MapperLib::ClusterId = size_t
- using MapperLib::IntegerCubeId = size_t

Functions

- std::ostream & operator<< (std::ostream &os, std::vector< size_t > const &vec)
- std::ostream & operator<< (std::ostream &os, MapperLib::Simplex const &simplex)
- std::ostream & operator<< (std::ostream &os, std::vector< MapperLib::Simplex > const &vec)

8.26.1 Function Documentation

8.26.1.1 operator<<() [1/3]

```
std::ostream & operator<< (
            std::ostream & os,
            MapperLib::Simplex const & simplex)  [inline]
```

8.26.1.2 operator<<() [2/3]

```
std::ostream & operator<< (
            std::ostream & os,
            std::vector< MapperLib::Simplex > const & vec)  [inline]
```

8.26.1.3 operator<<() [3/3]

```
std::ostream & operator<< (
            std::ostream & os,
            std::vector< size_t > const & vec)  [inline]
```

8.27 typedefs.h

Go to the documentation of this file.

```
//
// Created by jakob on 23.06.24.
//

#ifndef TYPEDEFS_H
#define TYPEDEFS_H
#include <vector>
#include <iostream>

namespace MapperLib{

using Scalar = double;
using Vector = std::vector<Scalar>;
using Matrix = std::vector<std::vector<Scalar»;
using PointId = size_t;
using Dimension = size_t;
using SimplexId = size_t;
using ClusterId = size_t;
using IntegerCubeId = size_t;

struct Simplex{
    std::vector<PointId> points;
    [[nodiscard]] std::vector<PointId> get_points(){ return points; }
    [[nodiscard]] Dimension dimension() const { return points.size() - 1;}
    [[nodiscard]] size_t num_nodes() const { return points.size(); }
    [[nodiscard]] PointId operator[](size_t index) const {return points[index]; }
};

struct MapperCluster{
    std::vector<PointId> points;
    ClusterId cluster_id;
    IntegerCubeId integer_cube_id;
};
}



inline std::ostream& operator«(std::ostream& os, std::vector<size_t> const& vec)
{
    os « "[";
    for(auto const elt: vec) {
        os « elt « ", ";
    }
    os « "\b\b]";
    return os;
}

inline std::ostream& operator«(std::ostream& os, MapperLib::Simplex const& simplex)
{
    return os « simplex.points;
}

inline std::ostream& operator«(std::ostream& os, std::vector<MapperLib::Simplex> const& vec)
{
    os « "[";
    for(auto const& elt: vec) {
        os « elt « ", ";
    }
    os « "\b\b]";
    return os;
}


#endif //TYPEDEFS_H
```

Index