MapperLib

Generated by Doxygen 1.12.0

| 1 Deprecated List | 1 |
|--|----|
| 2 Namespace Index | 3 |
| 2.1 Namespace List | |
| 3 Hierarchical Index | 5 |
| 3.1 Class Hierarchy | 5 |
| 4 Class Index | 7 |
| 4.1 Class List | |
| 5 File Index | 9 |
| 5.1 File List | 9 |
| 6 Namespace Documentation | 11 |
| 6.1 MapperLib Namespace Reference | 11 |
| 6.1.1 Typedef Documentation | 12 |
| 6.1.1.1 Cluster | 12 |
| 6.1.1.2 ClusterAssignment | 12 |
| 6.1.1.3 ClusterId | 12 |
| 6.1.1.4 Dimension | 12 |
| 6.1.1.5 IntegerCubeld | 12 |
| 6.1.1.6 Matrix | |
| 6.1.1.7 Pointld | |
| 6.1.1.8 Scalar | |
| 6.1.1.9 SimplexId | |
| 6.1.1.10 Vector | |
| 6.1.2 Function Documentation | |
| 6.1.2.1 check_data_equal_dimension() | |
| 6.1.2.2 euclididan distance() | |
| 6.1.2.3 get_data_dimension() | |
| 6.1.2.4 maximum_distance() | |
| 6.1.2.5 print() | |
| 6.1.2.6 PYBIND11_MODULE() | |
| 7 Class Documentation | 15 |
| 7.1 MapperLib::CechComplex Class Reference | 15 |
| 7.1.1 Detailed Description | |
| 7.1.2 Constructor & Destructor Documentation | |
| 7.1.2.1 CechComplex() | |
| 7.1.3 Member Function Documentation | _ |
| 7.1.3.1 generate() | |
| 7.2 MapperLib::CechComplexFactory Class Reference | |
| 7.2.1 Detailed Description | |
| The second of th | |

| 7.2.2 Constructor & Destructor Documentation | 17 |
|--|----|
| 7.2.2.1 CechComplexFactory() | 17 |
| 7.2.3 Member Function Documentation | 18 |
| 7.2.3.1 create_complex() | 18 |
| 7.2.3.2 make_shared() | 18 |
| 7.3 MapperLib::Clusterer Class Reference | 19 |
| 7.3.1 Detailed Description | 19 |
| 7.3.2 Constructor & Destructor Documentation | 19 |
| 7.3.2.1 ~Clusterer() | 19 |
| 7.3.3 Member Function Documentation | 19 |
| 7.3.3.1 predict() | 19 |
| 7.4 MapperLib::Complex Class Reference | 20 |
| 7.4.1 Detailed Description | 20 |
| 7.4.2 Constructor & Destructor Documentation | 20 |
| 7.4.2.1 ~Complex() | 20 |
| 7.4.3 Member Function Documentation | 20 |
| 7.4.3.1 generate() | 20 |
| 7.5 MapperLib::ComplexFactory Class Reference | 21 |
| 7.5.1 Detailed Description | 21 |
| 7.5.2 Constructor & Destructor Documentation | 21 |
| 7.5.2.1 ~ComplexFactory() | 21 |
| 7.5.3 Member Function Documentation | 21 |
| 7.5.3.1 create_complex() | 21 |
| 7.6 MapperLib::CoordinatePlaneProjection Class Reference | 22 |
| 7.6.1 Detailed Description | 22 |
| 7.6.2 Constructor & Destructor Documentation | 22 |
| 7.6.2.1 CoordinatePlaneProjection() | 22 |
| 7.6.3 Member Function Documentation | 23 |
| 7.6.3.1 make_shared() | 23 |
| 7.6.3.2 project() | 23 |
| 7.7 MapperLib::DataCover Class Reference | 23 |
| 7.7.1 Detailed Description | 24 |
| 7.7.2 Member Typedef Documentation | 24 |
| 7.7.2.1 Cubeld | 24 |
| 7.7.3 Constructor & Destructor Documentation | 24 |
| 7.7.3.1 DataCover() [1/2] | 24 |
| 7.7.3.2 DataCover() [2/2] | 24 |
| 7.7.4 Member Function Documentation | 25 |
| 7.7.4.1 convert_to_cube_id() | 25 |
| 7.7.4.2 convert_to_integer_cube_id() | 25 |
| 7.7.4.3 get_native_cube_id() | 25 |
| 7.7.4.4 get_neighbor_cubes() | 25 |

| 7.7.4.5 get_num_cubes_in_dimension() | 26 |
|---|----|
| 7.7.4.6 get_points_in_cube() | 26 |
| 7.7.4.7 get_total_num_cubes() | 26 |
| 7.7.4.8 is_vector_in_cube() | 26 |
| 7.8 MapperLib::DataCoverFactory Class Reference | 27 |
| 7.8.1 Detailed Description | 27 |
| 7.8.2 Constructor & Destructor Documentation | 27 |
| 7.8.2.1 DataCoverFactory() [1/2] | 27 |
| 7.8.2.2 DataCoverFactory() [2/2] | 28 |
| 7.8.3 Member Function Documentation | 28 |
| 7.8.3.1 create_data_cover() | 28 |
| 7.8.3.2 make_shared() [1/2] | 28 |
| 7.8.3.3 make_shared() [2/2] | 29 |
| 7.9 MapperLib::Mapper Class Reference | 29 |
| 7.9.1 Detailed Description | 30 |
| 7.9.2 Constructor & Destructor Documentation | 30 |
| 7.9.2.1 Mapper() | 30 |
| 7.9.3 Member Function Documentation | 30 |
| 7.9.3.1 map() | 30 |
| 7.10 MapperLib::MapperCluster Struct Reference | 31 |
| 7.10.1 Detailed Description | 31 |
| 7.10.2 Member Data Documentation | 31 |
| 7.10.2.1 cluster_id | 31 |
| 7.10.2.2 integer_cube_id | 31 |
| 7.10.2.3 points | 31 |
| 7.11 MapperLib::Projection Class Reference | 31 |
| 7.11.1 Detailed Description | 32 |
| 7.11.2 Constructor & Destructor Documentation | 32 |
| 7.11.2.1 ~Projection() | 32 |
| 7.11.3 Member Function Documentation | 32 |
| 7.11.3.1 project() | 32 |
| 7.12 MapperLib::Simplex Struct Reference | 32 |
| 7.12.1 Detailed Description | 33 |
| 7.12.2 Member Function Documentation | 33 |
| 7.12.2.1 dimension() | 33 |
| 7.12.2.2 get_points() | 33 |
| 7.12.2.3 num_nodes() | 33 |
| 7.12.2.4 operator[]() | 33 |
| 7.12.3 Member Data Documentation | 33 |
| 7.12.3.1 points | 33 |
| 7.13 MapperLib::SingleLinkage Class Reference | 33 |
| 7.13.1 Detailed Description | 34 |

| 7.13.2 Constructor & Destructor Documentation | 34 |
|---|----|
| 7.13.2.1 SingleLinkage() | 34 |
| 7.13.3 Member Function Documentation | 34 |
| 7.13.3.1 make_shared() | 34 |
| 7.13.3.2 predict() | 34 |
| 7.14 MapperLib::SLink_SingleLinkage Class Reference | 35 |
| 7.14.1 Detailed Description | 35 |
| 7.14.2 Constructor & Destructor Documentation | 35 |
| 7.14.2.1 SLink_SingleLinkage() | 35 |
| 7.14.3 Member Function Documentation | 36 |
| 7.14.3.1 make_shared() | 36 |
| 7.14.3.2 predict() | 36 |
| 8 File Documentation | 37 |
| 8.1 CechComplex.cpp File Reference | |
| 8.2 CechComplex.h File Reference | |
| 8.3 CechComplex.h | |
| 8.4 Clusterer.h File Reference | |
| 8.5 Clusterer.h | |
| 8.6 DataCover.cpp File Reference | 40 |
| 8.6.1 Function Documentation | 40 |
| 8.6.1.1 operator<<() | 40 |
| 8.7 DataCover.h File Reference | 40 |
| 8.7.1 Function Documentation | 41 |
| 8.7.1.1 operator<<() | 41 |
| 8.8 DataCover.h | 41 |
| 8.9 LinalgHelpers.cpp File Reference | 43 |
| 8.9.1 Function Documentation | 43 |
| 8.9.1.1 operator<<() [1/2] | 43 |
| 8.9.1.2 operator<<() [2/2] | 43 |
| 8.10 LinalgHelpers.h File Reference | 43 |
| 8.10.1 Function Documentation | 44 |
| 8.10.1.1 operator<<() [1/2] | 44 |
| 8.10.1.2 operator<<() [2/2] | 44 |
| 8.11 LinalgHelpers.h | 44 |
| 8.12 main.cpp File Reference | 45 |
| 8.12.1 Function Documentation | 45 |
| 8.12.1.1 main() | 45 |
| 8.13 Mapper.cpp File Reference | |
| 8.14 Mapper.h File Reference | |
| 8.15 Mapper.h | |
| 8.16 Projection.cpp File Reference | 46 |

| | 8.17 Projection.h File Reference | 47 |
|-----|---|----|
| | 8.18 Projection.h | 47 |
| | 8.19 PythonModule.cpp File Reference | 47 |
| | 8.20 SingleLinkage.cpp File Reference | 48 |
| | 8.21 SingleLinkage.h File Reference | 48 |
| | 8.22 SingleLinkage.h | 49 |
| | 8.23 SLink_SingleLinkage.cpp File Reference | 49 |
| | 8.24 SLink_SingleLinkage.h File Reference | 49 |
| | 8.25 SLink_SingleLinkage.h | 50 |
| | 8.26 typedefs.h File Reference | 50 |
| | 8.26.1 Function Documentation | 51 |
| | 8.26.1.1 operator<<() [1/3] | 51 |
| | 8.26.1.2 operator<<() [2/3] | 51 |
| | 8.26.1.3 operator<<() [3/3] | 51 |
| | 8.27 typedefs.h | 51 |
| Ind | lex | 53 |

Deprecated List

Class MapperLib::SingleLinkage

use SLink_SingleLinkage instead

2 Deprecated List

Namespace Index

| 2.1 Namespace Lis | st |
|-------------------|----|
|-------------------|----|

| Here is a list of all namespaces with brief descriptions: | | | | |
|---|----|--|--|--|
| MapperLib | 11 | | | |

4 Namespace Index

Hierarchical Index

3.1 Class Hierarchy

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

| MapperLib::Clusterer | 19 |
|--------------------------------------|------|
| MapperLib::SLink_SingleLinkage | . 35 |
| MapperLib::SingleLinkage | . 33 |
| MapperLib::Complex | 20 |
| MapperLib::CechComplex | . 15 |
| MapperLib::ComplexFactory | 21 |
| MapperLib::CechComplexFactory | . 17 |
| MapperLib::DataCover | 23 |
| MapperLib::DataCoverFactory | 27 |
| MapperLib::Mapper | 29 |
| MapperLib::MapperCluster | 31 |
| MapperLib::Projection | 31 |
| MapperLib::CoordinatePlaneProjection | . 22 |
| MapperLib::Simplex | 32 |

6 Hierarchical Index

Class Index

4.1 Class List

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

| MapperLib::CechComplex | |
|---|------------|
| Class for generating Cech complexes from overlapping clusters | 5 |
| MapperLib::CechComplexFactory | |
| Factory class for creating CechComplex objects | 7 |
| MapperLib::Clusterer | |
| Abstract base class for clustering algorithms | 9 |
| MapperLib::Complex | |
| Abstract base class for simplicial complexes | 02 |
| MapperLib::ComplexFactory | |
| Abstract factory class for creating Complex objects | 11 |
| MapperLib::CoordinatePlaneProjection | |
| Projection of data to coordinate planes | 2 |
| MapperLib::DataCover | |
| Class for sectioning data into hypercubes | 23 |
| MapperLib::DataCoverFactory | |
| Factory class creating DataCover objects | <u>'</u> 7 |
| MapperLib::Mapper | |
| Class implementing the Mapper algorithm | 29 |
| MapperLib::MapperCluster | |
| Cluster containign additional information | 11 |
| MapperLib::Projection | |
| Abstract base class for projection methods | 11 |
| MapperLib::Simplex | |
| Struct representing a simplex | 2 |
| MapperLib::SingleLinkage | |
| Primitive implementation of single linkage clustering | 3 |
| MapperLib::SLink_SingleLinkage | |
| Efficient implementation of single linkage clustering | 5 |

8 Class Index

File Index

5.1 File List

Here is a list of all files with brief descriptions:

| CechComplex.cpp | 37 |
|-------------------------|----|
| CechComplex.h | 37 |
| Clusterer.h | 39 |
| DataCover.cpp | 40 |
| DataCover.h | 40 |
| LinalgHelpers.cpp | 43 |
| LinalgHelpers.h | 43 |
| | 45 |
| Mapper.cpp | 45 |
| | 45 |
| Projection.cpp | 46 |
| Projection.h | 47 |
| PythonModule.cpp | 47 |
| SingleLinkage.cpp | 48 |
| | 48 |
| SLink_SingleLinkage.cpp | 49 |
| SLink_SingleLinkage.h | 49 |
| typedefs.h | 50 |

10 File Index

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 IntegerCubeld

```
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()

Checks that all columns in the matrix have the same dimension, as Matrix is only an alias for $std \leftarrow ::vector < std ::vector < scalar >>$

Parameters

```
mat the matrix to check
```

Returns

true if matrix is valid, else false

6.1.2.2 euclididan_distance()

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()

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()

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()

6.1.2.6 PYBIND11_MODULE()

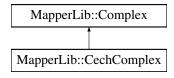
Class Documentation

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()

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()

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:

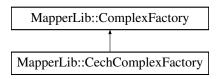
- · CechComplex.h
- CechComplex.cpp

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()

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()

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()

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:

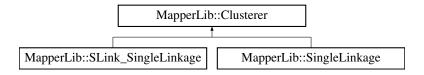
- CechComplex.h
- CechComplex.cpp

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()

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

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

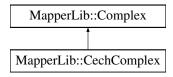
· Clusterer.h

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 \sim Complex()

```
\label{local_complex} \mbox{virtual MapperLib::Complex::} \sim \mbox{Complex ()} \quad \mbox{[virtual], [default]}
```

7.4.3 Member Function Documentation

7.4.3.1 generate()

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:

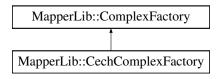
CechComplex.h

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()

```
\verb|virtual MapperLib::ComplexFactory:: \sim \verb|ComplexFactory () [virtual], [default]| \\
```

7.5.3 Member Function Documentation

7.5.3.1 create_complex()

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:

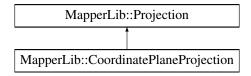
· CechComplex.h

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()

Initialize a projection

Parameters

| ane |
|-----|
| |

7.6.3 Member Function Documentation

7.6.3.1 make_shared()

7.6.3.2 project()

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 Cubeld = 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)
- Cubeld 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
- IntegerCubeld convert to integer cube id (Cubeld const &cube id) const
- Cubeld convert_to_cube_id (IntegerCubeld 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, Cubeld 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 Cubeld

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

7.7.3 Constructor & Destructor Documentation

7.7.3.1 DataCover() [1/2]

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 \leq 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()

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

Parameters

| integer_cube <i>⊷</i> | the id to convert |
|-----------------------|-------------------|
| _id | |

Returns

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

7.7.4.2 convert_to_integer_cube_id()

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

Parameters

| cube← | the id to convert |
|-------|-------------------|
| _id | |

Returns

an integer uniquely representing the cube

7.7.4.3 get_native_cube_id()

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()

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()

```
\label{local_const_size} size\_t \ \ \mbox{MapperLib::DataCover::get\_num\_cubes\_in\_dimension (} \\ \mbox{Dimension $dim$)} \ \mbox{const}
```

7.7.4.6 get_points_in_cube()

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()

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]

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()

Create a DataCover object

Parameters

| data | A reference to the data points |
|------|--------------------------------|
| uaia | A reference to the data points |

Returns

Unique pointer to a new DataCover instance

7.8.3.2 make_shared() [1/2]

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]

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)

30 Class Documentation

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()

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()

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:

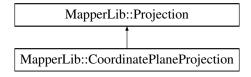
· typedefs.h

7.11 MapperLib::Projection Class Reference

Abstract base class for projection methods.

#include <Projection.h>

Inheritance diagram for MapperLib::Projection:



32 Class Documentation

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()

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[]()

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:

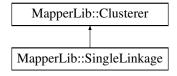
· typedefs.h

7.13 MapperLib::SingleLinkage Class Reference

primitive implementation of single linkage clustering

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

Inheritance diagram for MapperLib::SingleLinkage:



34 Class Documentation

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()

7.13.3 Member Function Documentation

7.13.3.1 make_shared()

7.13.3.2 predict()

Implements MapperLib::Clusterer.

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

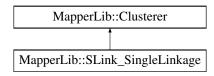
- · SingleLinkage.h
- SingleLinkage.cpp

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()

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.

36 Class Documentation

Parameters

| num | n_clusters | Optional integer specifying the desired number of clusters. |
|-------|----------------|---|
| dista | ance_threshold | Optional Scalar value for the maximum distance threshold. |

7.14.3 Member Function Documentation

7.14.3.1 make_shared()

7.14.3.2 predict()

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.

```
00002 // Created by jgier on 29.06.2024.
00003 //
00004
00005 #ifndef MAPPER_CECHCOMPLEX_H
00006 #define MAPPER CECHCOMPLEX H
00007
00008 #include <vector>
00009 #include <generator>
00010 #include <memory>
00011
00012 #include "typedefs.h"
00013
00014 namespace MapperLib {
00015
00016 class DataCover;
00017
00024 class Complex {
00025 public:
00026
          virtual ~Complex() = default;
          [[nodiscard]] virtual std::vector<Simplex> generate(std::vector<MapperCluster> const &clusters)
     const = 0;
00037 };
00038
00047 class CechComplex final : public Complex {
          static constexpr size_t NUM_THREADS = 8;
00059
          CechComplex(DataCover const& data_cover, Dimension max_dimension);
00060
00071
          [[nodiscard]] std::vector<Simplex> generate(std::vector<MapperCluster> const& clusters) const
      override;
00072
00073 private:
00074
          using Iterator = std::vector<MapperCluster>::const_iterator;
00075
          using ClustersByCube = std::vector<std::vector<MapperCluster const*»;</pre>
00076
00083
          class SimplexComputer {
00084
             Iterator _begin;
00085
              Iterator _end;
              ClustersByCube& _clusters_by_cube;
Dimension const _dim;
00086
00087
00088
              DataCover const& _data_cover;
00089
              std::vector<Simplex> _own_result;
00090
              std::vector<Simplex> &_result;
00091
              std::mutex &_mutex;
00092
              int _id;
00093
00094
          public:
              static inline int id_counter = 0;
00095
00096
00110
              SimplexComputer(Iterator begin, Iterator end, ClustersByCube &clusters_by_cube, Dimension dim,
00111
                               DataCover const &data_cover, std::vector<Simplex> &result, std::mutex &mutex);
00112
00118
              void compute();
00119
          };
00120
          struct ComputerWrapper {
00128
              SimplexComputer & computer;
00129
00133
              void operator()() const { computer.compute(); }
00134
          };
00135
00145
          [[nodiscard]] std::vector<Simplex> generate_k_simplices(std::vector<MapperCluster> const&
     clusters, Dimension k) const;
00146
00156
          static std::generator<std::vector<size_t» generate_k_subsets_of_range(size_t index_max, size_t k);</pre>
00157
          static bool check_cluster_intersection(std::vector<MapperCluster const*> const& all_clusters,
00167
      std::vector<size t> const& relevant indices);
00168
00178
          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
```

```
00179
00180
          DataCover const &_data_cover;
00181
          Dimension _max_dimension;
00182 };
00183
00190 class ComplexFactory {
00191 public:
00192
          virtual ~ComplexFactory() = default;
00193
00202
          [[nodiscard]] virtual std::unique_ptr<Complex> create_complex(DataCover const& data_cover) const =
     0;
00203 };
00204
00211 class CechComplexFactory final : public ComplexFactory {
00212 public:
00220
          explicit CechComplexFactory(Dimension max_dimension);
00221
00230
          [[nodiscard]] static std::shared_ptr<ComplexFactory> make_shared(Dimension max_dimension);
00231
00240
          [[nodiscard]] std::unique_ptr<Complex> create_complex(DataCover const &data_cover) const override;
00241
00242 private:
00243
         Dimension _max_dimension;
00244 };
00245
00246 } // namespace MapperLib
00247
00248 #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.

```
00001 //
00002 // Created by jakob on 23.06.24.
00003 //
00004
00005 #ifndef CLUSTERER_H
00006 #define CLUSTERER_H
00007
00008 #include "typedefs.h"
00009
00010
00011 namespace MapperLib {
```

```
00013 using Cluster = std::vector<PointId>;
00014 using ClusterAssignment = std::vector<Cluster>;
00015
00020 class Clusterer {
00021 public:
         virtual ~Clusterer() = default;
00023
         virtual ClusterAssignment predict(Matrix const &data, std::vector<PointId> data_filter) = 0;
00024 private:
00025 };
00026
00027 } // cluster
00028
00029
00030
00031 #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::Cubeld const &vec)

8.6.1 Function Documentation

8.6.1.1 operator<<()

8.7 DataCover.h File Reference

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

8.8 DataCover.h 41

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::Cubeld const &vec)

8.7.1 Function Documentation

8.7.1.1 operator<<()

8.8 DataCover.h

Go to the documentation of this file.

```
00001 //
00002 // Created by jakob on 24.06.24. 00003 //
00004
00005 #ifndef DATACOVER_H
00006 #define DATACOVER_H
00007
00008 #include <memory>
00009 #include <ostream>
00010
00011 #include "SingleLinkage.h"
00012 #include "typedefs.h"
00013
00014 namespace MapperLib {
00021 class DataCover {
00022 public:
00023
          using CubeId = std::vector<int>;
00024
00033
           DataCover(
00034
              size_t resolution,
00035
               double perc_overlap,
00036
               Matrix const& data,
               std::optional<Vector> minima = std::nullopt,
std::optional<Vector> maxima = std::nullopt
00037
00038
00039
          );
00040
00049
          DataCover(
00050
               std::vector<size t> resolution.
00051
               double perc_overlap,
00052
               Matrix const& data,
00053
               std::optional<Vector> minima = std::nullopt,
00054
               std::optional<Vector> maxima = std::nullopt
00055
           );
00056
00062
           [[nodiscard]] CubeId get_native_cube_id(Vector const& vec) const;
00063
00069
           [[nodiscard]] std::vector<PointId> get_points_in_cube(IntegerCubeId cube_id) const;
```

```
00070
00076
          [[nodiscard]] std::vector<IntegerCubeId> get_neighbor_cubes(IntegerCubeId integer_cube_id) const;
00077
00083
          IntegerCubeId convert_to_integer_cube_id(CubeId const& cube_id) const;
00084
00090
          CubeId convert to cube id(IntegerCubeId integer cube id) const:
00096
          size_t get_total_num_cubes() const;
00097
00098
          size_t get_num_cubes_in_dimension(Dimension dim) const;
00099
00106
          bool is vector in cube (Vector const& vec. CubeId const& cube id) const:
00107
00108 private:
00109
          void initialize_cube_cache() const;
00110
00111
          std::vector<CubeId> get_parent_cubes(Vector const& v) const;
00112
00113
          [[nodiscard]] Scalar get_data_min_in_dimension(Dimension dimension) const;
00114
          [[nodiscard]] Scalar get_data_max_in_dimension(Dimension dimension) const;
00115
00116
          void initialize_minima_from_data();
00117
          void initialize_maxima_from_data();
00118
00119
          [[nodiscard]] Vector get_cube_center(CubeId const& cube_id) const;
00120
          [[nodiscard]] std::vector<CubeId> get_neighbor_cubes(CubeId const& cube_id) const;
00121
00122
00123
          std::vector<size_t> _resolution;
          double const _perc_overlap;
Matrix const& _data;
size_t const _data_dimension;
00124
00125
00126
00127
          Vector _minima;
00128
          Vector _maxima;
00129
00130
          mutable std::optional<std::vector<std::vector<PointId»> _cube_cache;
00131
00132 };
00133
00140 class DataCoverFactory
00141 {
00142 public:
          DataCoverFactory(
00150
00151
              size_t resolution,
00152
              double perc_overlap,
00153
              std::optional<Vector> minima = std::nullopt,
00154
              std::optional<Vector> maxima = std::nullopt
00155
          );
00156
00164
          DataCoverFactory(
00165
              std::vector<size_t> resolution,
00166
              double perc_overlap,
00167
              std::optional<Vector> minima = std::nullopt,
00168
              std::optional<Vector> maxima = std::nullopt
00169
          );
00170
00179
          [[nodiscard]] static std::shared_ptr<DataCoverFactory> make_shared(
00180
              size_t resolution,
00181
              double perc_overlap,
00182
              std::optional<Vector> minima = std::nullopt,
00183
              std::optional<Vector> maxima = std::nullopt
00184
          );
00185
00194
          [[nodiscard]] static std::shared_ptr<DataCoverFactory> make_shared(
00195
              std::vector<size_t> resolution,
00196
              double perc_overlap,
00197
              std::optional<Vector> minima = std::nullopt,
              std::optional<Vector> maxima = std::nullopt
00198
00199
00200
00206
          [[nodiscard]] std::unique_ptr<DataCover> create_data_cover(Matrix const& data) const;
00207
00208 private:
          std::vector<size_t> _resolution;
00209
          std::optional<size_t> _single_resolution;
00210
00211
          double _perc_overlap;
          std::optional<Vector> _minima;
std::optional<Vector> _maxima;
00212
00213
00214 };
00215
00216
00218 } // Mapper
00219 std::ostream& operator«(std::ostream& stream, MapperLib::DataCover::CubeId const& vec);
00220
00221
00222
```

```
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)</pre>
```

8.9.1.2 operator <<() [2/2]

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

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]
```

8.10.1.2 operator <<() [2/2]

8.11 LinalgHelpers.h

Go to the documentation of this file.

```
00001 //
00002 // Created by jakob on 23.06.24.
00003 //
00004
00005 #ifndef LINALGHELPERS_H
00006 #define LINALGHELPERS_H
00007 #include <ostream>
80000
00009 #include "typedefs.h"
00010
00011 namespace MapperLib {
00018 Scalar euclididan_distance(Vector const& vec1, Vector const& vec2);
00019
00026 Scalar maximum_distance(Vector const& vec1, Vector const& vec2);
00027
00033 bool check_data_equal_dimension(Matrix const& mat);
00034
00041 Dimension get_data_dimension(Matrix const& mat);
00042
00043 void print (Vector const& vec);
00044 } // Helper
00045
00046 std::ostream& operator«(std::ostream& os, MapperLib::Vector const& vec);
00047 std::ostream& operator«(std::ostream& os, MapperLib::Matrix const& mat);
00048
00049 #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.

```
00001 //
00002 // Created by jakob on 26.06.24.
00003 //
00004
00005 #ifndef MAPPER_H
00006 #define MAPPER_H
00007 #include <vector>
00008 #include <memory>
00009 #include "typedefs.h"
00010 #include "DataCover.h"
00012
00013
00014 namespace MapperLib {
00015 class ComplexFactory;
00016 class Complex;
00017 class Clusterer;
00018 class Projection;
00019
00029 class Mapper {
00030 public:
00038
           Mapper(
00039
               std::shared_ptr<DataCoverFactory> data_cover_factory,
00040
                std::shared_ptr<ComplexFactory> complex_factory,
00041
                std::shared_ptr<Clusterer> clusterer,
00042
               std::shared_ptr<Projection> projection
00043
           );
00044
00050
           [[nodiscard]] std::vector<Simplex> map(Matrix const& data) ;
00051
00052 private:
00053
           std::shared_ptr<DataCoverFactory> _data_cover_factory;
00054
           std::shared_ptr<ComplexFactory> _complex_factory;
           std::shared_ptr<Clusterer> _clusterer;
std::shared_ptr<Projection> _projection;
std::unique_ptr<DataCover> _data_cover;
00055
00056
00058
           std::unique_ptr<Complex> _complex;
00059 };
00060
00061 } // Mapper
00062
00063 #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.

```
00002 // Created by jgier on 27.06.2024.
00003 //
00004
00005 #ifndef MAPPER_PROJECTION_H
00006 #define MAPPER_PROJECTION_H
00007 #include <memory>
80000
00009 #include "typedefs.h"
00010
00011 namespace MapperLib(
00016 class Projection {
00017 public:
00018
         virtual ~Projection() = default;
00019
          [[nodiscard]] virtual Matrix project(Matrix const& data) const = 0;
00020 };
00021
00028 class CoordinatePlaneProjection final : public Projection{
00034
         explicit CoordinatePlaneProjection(std::vector<Dimension> dimensions);
00035
         [[nodiscard]] static std::shared_ptr<Projection> make_shared(std::vector<Dimension> dimensions);
00036
00042
         [[nodiscard]] Matrix project (Matrix const& data) const override;
00043 private:
00044
         std::vector<Dimension> _dimensions;
00045 };
00046
00047 } // Mapper
00048
00049 #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 49

8.22 SingleLinkage.h

```
Go to the documentation of this file. 00001 // 00002 // Created by jakob on 14.06
```

```
00002 // Created by jakob on 14.06.24.
00004
00005 #ifndef SINGLELINKAGE_H
00006 #define SINGLELINKAGE_H
00007 #include <limits>
00008 #include <memorv>
00009 #include <optional>
00011 #include "Clusterer.h"
00012 #include "LinalgHelpers.h"
00013
00014 namespace MapperLib {
00020 class SingleLinkage : public Clusterer {
00021 public:
00022
00023
          SingleLinkage(std::optional<int> num_clusters, std::optional<Scalar> distance_threshold);
00024
          [[nodiscard]] static std::shared_ptr<Clusterer> make_shared(std::optional<int> num_clusters,
     std::optional<Scalar> distance_threshold);
00025
          ClusterAssignment predict (Matrix const &data, std::vector<PointId> data_filter) override;
00027
00028 private:
00029 [[nodiscard]] static Scalar min_distance(
          Matrix const& distances,
00030
00031
              std::vector<PointId> const& cluster1.
00032
              std::vector<PointId> const& cluster2
00033
         );
00034
00035
          std::optional<size_t> _num_clusters;
00036
          std::optional<Scalar> _distance_threshold;
00037
00038 };
00039
00040 } // Cluster
00041
00042 #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.

```
00001 //
00002 // Created by jakob on 10.07.24.
00003 //
00004
00005 #ifndef SLINK_SINGLELINKAGE_H
00006 #define SLINK_SINGLELINKAGE_H
00007
00008 #include <memory>
00009
00010 #include "Clusterer.h"
00011
00012 namespace MapperLib {
00020 class SLink_SingleLinkage : public Clusterer{
00021 public:
00022
00032
          SLink_SingleLinkage(std::optional<int> num_clusters, std::optional<Scalar> distance_threshold);
00033
           [[nodiscard]] static std::shared_ptr<Clusterer> make_shared(std::optional<int> num_clusters,
      std::optional<Scalar> distance_threshold);
00034
00044
          ClusterAssignment predict (Matrix const &data, std::vector<PointId> data_filter) override;
00045 private:
          std::optional<size_t> _num_clusters;
std::optional<Scalar> _distance_threshold;
00046
00047
00048 };
00049
00050 } // Mapper
00051
00052 #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

8.27 typedefs.h 51

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

8.27 typedefs.h

Go to the documentation of this file.

```
00001 //
00002 // Created by jakob on 23.06.24.
00003 //
00004
00005 #ifndef TYPEDEFS_H
00006 #define TYPEDEFS_H
00007 #include <vector>
00008 #include <iostream>
00009
00010 namespace MapperLib{
00011
00012 using Scalar = double;
00013 using Vector = std::vector<Scalar>;
00014 using Matrix = std::vector<std::vector<Scalar»;</pre>
00015 using PointId = size_t;
00016 using Dimension = size_t;
00017 using SimplexId = size_t;
00018 using ClusterId = size_t;
00019 using IntegerCubeId = size_t;
00020
00027 struct Simplex{
          std::vector<PointId> points;
00028
             [[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(); }
00029
00030
00031
00032
             [[nodiscard]] PointId operator[](size_t index) const {return points[index]; }
00033 };
00034
00041 struct MapperCluster{
00042
          std::vector<PointId> points;
            ClusterId cluster_id;
```

```
00044
          IntegerCubeId integer_cube_id;
00045 };
00046 }
00048
00049
00050 inline std::ostream& operator«(std::ostream& os, std::vector<size_t> const& vec)
00051 {
00052
          os « "[";
          for(auto const elt: vec) {
   os « elt « ", ";
}
00053
00054
00055
00056
          os « "\b\b]";
00057
          return os;
00058 }
00059 00060 inline std::ostream& operator«(std::ostream& os, MapperLib::Simplex const& simplex)
00061 {
00062
          return os « simplex.points;
00063 }
00064
00065 inline std::ostream& operator«(std::ostream& os, std::vector<MapperLib::Simplex> const& vec)
00066 {
          os « "[";
for(auto const& elt: vec) {
   os « elt « ", ";
00067
00068
00069
00070
00071
         os « "\b\b]";
00072
          return os;
00073 }
00074
00075
00076 #endif //TYPEDEFS_H
```

Index

| \sim Clusterer | MapperLib, 12 |
|--|--|
| MapperLib::Clusterer, 19 | dimension |
| ~Complex | MapperLib::Simplex, 33 |
| MapperLib::Complex, 20 | |
| ~ComplexFactory | euclididan_distance |
| MapperLib::ComplexFactory, 21 | MapperLib, 13 |
| ~Projection | |
| MapperLib::Projection, 32 | generate |
| | MapperLib::CechComplex, 16 |
| CechComplex | MapperLib::Complex, 20 |
| MapperLib::CechComplex, 16 | get_data_dimension |
| CechComplex.cpp, 37 | MapperLib, 14 |
| CechComplex.h, 37 | get_native_cube_id |
| CechComplexFactory | MapperLib::DataCover, 25 |
| MapperLib::CechComplexFactory, 17 | get_neighbor_cubes |
| check_data_equal_dimension | MapperLib::DataCover, 25 |
| MapperLib, 13 | get_num_cubes_in_dimension |
| Cluster | MapperLib::DataCover, 26 |
| MapperLib, 12 | get_points |
| cluster_id | MapperLib::Simplex, 33 |
| MapperLib::MapperCluster, 31 | get_points_in_cube |
| ClusterAssignment | MapperLib::DataCover, 26 |
| MapperLib, 12 | get_total_num_cubes |
| Clusterer.h, 39 | MapperLib::DataCover, 26 |
| ClusterId | |
| MapperLib, 12 | integer_cube_id |
| convert_to_cube_id | MapperLib::MapperCluster, 31 |
| MapperLib::DataCover, 25 | IntegerCubeId |
| convert_to_integer_cube_id | MapperLib, 12 |
| MapperLib::DataCover, 25 | is_vector_in_cube |
| CoordinatePlaneProjection | MapperLib::DataCover, 26 |
| MapperLib::CoordinatePlaneProjection, 22 | LinalgHelpers.cpp, 43 |
| create_complex | operator<<, 43 |
| MapperLib::CechComplexFactory, 18 | LinalgHelpers.h, 43 |
| MapperLib::ComplexFactory, 21 | operator<<, 44 |
| create_data_cover | operator < , 44 |
| MapperLib::DataCoverFactory, 28 | main |
| Cubeld | main.cpp, 45 |
| MapperLib::DataCover, 24 | main.cpp, 45 |
| , | main, 45 |
| DataCover | make shared |
| MapperLib::DataCover, 24 | MapperLib::CechComplexFactory, 18 |
| DataCover.cpp, 40 | MapperLib::CoordinatePlaneProjection, 23 |
| operator<<, 40 | MapperLib::DataCoverFactory, 28, 29 |
| DataCover.h, 40 | MapperLib::SingleLinkage, 34 |
| operator<<, 41 | MapperLib::SLink_SingleLinkage, 36 |
| DataCoverFactory | map |
| MapperLib::DataCoverFactory, 27, 28 | MapperLib::Mapper, 30 |
| Deprecated List, 1 | Mapper Mapper, 88 |
| Dimension | Mannerl ih: Manner 30 |

54 INDEX

| Mapper.cpp, 45 Mapper.h, 45 MapperLib, 11 | cluster_id, 31 integer_cube_id, 31 points, 31 |
|---|---|
| • • | • |
| check_data_equal_dimension, 13 | MapperLib::Projection, 31 |
| Cluster, 12 | ~Projection, 32 |
| ClusterAssignment, 12 | project, 32 |
| ClusterId, 12 | MapperLib::Simplex, 32 |
| Dimension, 12 | dimension, 33 |
| euclididan_distance, 13 | get_points, 33 |
| get_data_dimension, 14 | num_nodes, 33 |
| IntegerCubeld, 12 | operator[], 33 |
| Matrix, 12 | points, 33 |
| maximum_distance, 14 | MapperLib::SingleLinkage, 33 |
| Pointld, 13 | make_shared, 34 |
| print, 14 | predict, 34 |
| PYBIND11_MODULE, 14 | SingleLinkage, 34 |
| Scalar, 13 | MapperLib::SLink_SingleLinkage, 35 |
| SimplexId, 13 | make_shared, 36 |
| Vector, 13 | predict, 36 |
| MapperLib::CechComplex, 15 | SLink_SingleLinkage, 35 |
| CechComplex, 16 | Matrix |
| generate, 16 | MapperLib, 12 |
| MapperLib::CechComplexFactory, 17 | maximum_distance |
| CechComplexFactory, 17 | MapperLib, 14 |
| create_complex, 18 | |
| make_shared, 18 | num_nodes |
| MapperLib::Clusterer, 19 | MapperLib::Simplex, 33 |
| ∼Clusterer, 19 | |
| predict, 19 | operator<< |
| MapperLib::Complex, 20 | DataCover.cpp, 40 |
| ~Complex, 20 | DataCover.h, 41 |
| generate, 20 | LinalgHelpers.cpp, 43 |
| MapperLib::ComplexFactory, 21 | LinalgHelpers.h, 44 |
| ~ComplexFactory, 21 | typedefs.h, 51 |
| create complex, 21 | operator[] |
| MapperLib::CoordinatePlaneProjection, 22 | MapperLib::Simplex, 33 |
| CoordinatePlaneProjection, 22 | |
| make shared, 23 | PointId |
| project, 23 | MapperLib, 13 |
| MapperLib::DataCover, 23 | points |
| convert to cube id, 25 | MapperLib::MapperCluster, 31 |
| convert to integer cube id, 25 | MapperLib::Simplex, 33 |
| Cubeld, 24 | predict |
| DataCover, 24 | MapperLib::Clusterer, 19 |
| get_native_cube_id, 25 | MapperLib::SingleLinkage, 34 |
| get_neighbor_cubes, 25 | MapperLib::SLink_SingleLinkage, 36 |
| get_num_cubes_in_dimension, 26 | print |
| get points in cube, 26 | MapperLib, 14 |
| get_total_num_cubes, 26 | project |
| is_vector_in_cube, 26 | MapperLib::CoordinatePlaneProjection, 23 |
| | MapperLib::Projection, 32 |
| MapperLib::DataCoverFactory, 27 | Projection.cpp, 46 |
| create_data_cover, 28 | Projection.h, 47 |
| DataCoverFactory, 27, 28 | PYBIND11_MODULE |
| make_shared, 28, 29 | MapperLib, 14 |
| MapperLib::Mapper, 29 | PythonModule.cpp, 47 |
| map, 30 | |
| Mapper, 30 | Scalar |
| MapperLib::MapperCluster, 31 | MapperLib, 13 |

INDEX 55

```
SimplexId
MapperLib, 13
SingleLinkage
MapperLib::SingleLinkage, 34
SingleLinkage.cpp, 48
SingleLinkage.h, 48
SLink_SingleLinkage
MapperLib::SLink_SingleLinkage, 35
SLink_SingleLinkage.cpp, 49
SLink_SingleLinkage.h, 49

typedefs.h, 50
operator<<, 51

Vector
MapperLib, 13
```