

MapperLib

Generated by Doxygen 1.12.0

1 Deprecated List	1
2 Namespace Index	3
2.1 Namespace List	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
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	13
6.1.1.7 PointId	13
6.1.1.8 Scalar	13
6.1.1.9 SimplexId	13
6.1.1.10 Vector	13
6.1.2 Function Documentation	13
6.1.2.1 check_data_equal_dimension()	13
6.1.2.2 euclididan_distance()	13
6.1.2.3 get_data_dimension()	14
6.1.2.4 maximum_distance()	14
6.1.2.5 print()	14
6.1.2.6 PYBIND11_MODULE()	14
7 Class Documentation	15
7.1 MapperLib::CechComplex Class Reference	15
7.1.1 Detailed Description	15
7.1.2 Constructor & Destructor Documentation	16
7.1.2.1 CechComplex()	16
7.1.3 Member Function Documentation	16
7.1.3.1 generate()	16
7.2 MapperLib::CechComplexFactory Class Reference	17
7.2.1 Detailed Description	17

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 CubelId	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 <code>get_num_cubes_in_dimension()</code>	26
7.7.4.6 <code>get_points_in_cube()</code>	26
7.7.4.7 <code>get_total_num_cubes()</code>	26
7.7.4.8 <code>is_vector_in_cube()</code>	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 <code>DataCoverFactory()</code> [1/2]	27
7.8.2.2 <code>DataCoverFactory()</code> [2/2]	28
7.8.3 Member Function Documentation	28
7.8.3.1 <code>create_data_cover()</code>	28
7.8.3.2 <code>make_shared()</code> [1/2]	28
7.8.3.3 <code>make_shared()</code> [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 <code>Mapper()</code>	30
7.9.3 Member Function Documentation	30
7.9.3.1 <code>map()</code>	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 <code>cluster_id</code>	31
7.10.2.2 <code>integer_cube_id</code>	31
7.10.2.3 <code>points</code>	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 <code>~Projection()</code>	32
7.11.3 Member Function Documentation	32
7.11.3.1 <code>project()</code>	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 <code>dimension()</code>	33
7.12.2.2 <code>get_points()</code>	33
7.12.2.3 <code>num_nodes()</code>	33
7.12.2.4 <code>operator[]()</code>	33
7.12.3 Member Data Documentation	33
7.12.3.1 <code>points</code>	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	37
8.2 CechComplex.h File Reference	37
8.3 CechComplex.h	38
8.4 Clusterer.h File Reference	39
8.5 Clusterer.h	39
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	45
8.14 Mapper.h File Reference	45
8.15 Mapper.h	46
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
Index	53

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:

MapperLib	11
---------------------------	-------	----

Chapter 3

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

Chapter 4

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	15
MapperLib::CechComplexFactory	Factory class for creating CechComplex objects	17
MapperLib::Clusterer	Abstract base class for clustering algorithms	19
MapperLib::Complex	Abstract base class for simplicial complexes	20
MapperLib::ComplexFactory	Abstract factory class for creating Complex objects	21
MapperLib::CoordinatePlaneProjection	Projection of data to coordinate planes	22
MapperLib::DataCover	Class for sectioning data into hypercubes	23
MapperLib::DataCoverFactory	Factory class creating DataCover objects	27
MapperLib::Mapper	Class implementing the Mapper algorithm	29
MapperLib::MapperCluster	Cluster containign additional information	31
MapperLib::Projection	Abstract base class for projection methods	31
MapperLib::Simplex	Struct representing a simplex	32
MapperLib::SingleLinkage	Primitive implementation of single linkage clustering	33
MapperLib::SLink_SingleLinkage	Efficient implementation of single linkage clustering	35

Chapter 5

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
main.cpp	45
Mapper.cpp	45
Mapper.h	45
Projection.cpp	46
Projection.h	47
PythonModule.cpp	47
SingleLinkage.cpp	48
SingleLinkage.h	48
SLink_SingleLinkage.cpp	49
SLink_SingleLinkage.h	49
typedefs.h	50

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 euclidian_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

<i>mat</i>	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

<i>vec1</i>	first data point
<i>vec2</i>	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

<i>mat</i>	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

<i>vec1</i>	first data point
<i>vec2</i>	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

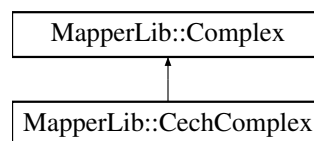
Class Documentation

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

<i>data_cover</i>	A reference to a data_cover object.
<i>max_dimension</i>	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

<i>clusters</i>	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:

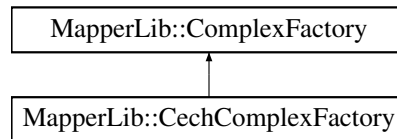
- [CechComplex.h](#)
- [CechComplex.cpp](#)

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

<i>max_dimension</i>	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

<i>data_cover</i>	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

<i>max_dimension</i>	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:

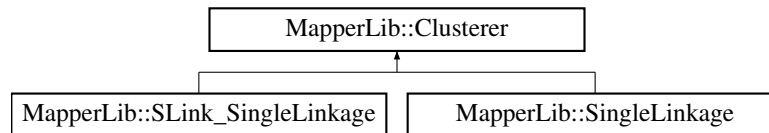
- [CechComplex.h](#)
- [CechComplex.cpp](#)

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:

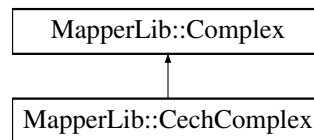
- [Clusterer.h](#)

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

<i>clusters</i>	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:

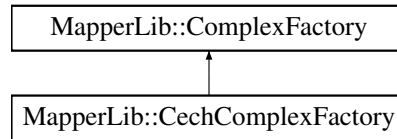
- [CechComplex.h](#)

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

<i>data_cover</i>	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:

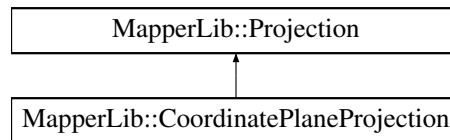
- [CechComplex.h](#)

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

<i>dimensions</i>	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

<i>data</i>	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 ([IntegerCubeld](#) cube_id) const
- std::vector< [IntegerCubeld](#) > get_neighbor_cubes ([IntegerCubeld](#) 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]

```
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

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

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

<i>integer_cube_id</i>	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

<i>cube_id</i>	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

<i>vec</i>	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

<i>integer_cube</i> ↔ _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

<i>cube</i> ↔ _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

<i>vec</i>	the data point
<i>cube</i> <i>_id</i>	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

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

<i>resolution</i>	Vector of resolution values for each dimension
<i>perc_overlap</i>	Percentage of overlap between hypercubes
<i>minima</i>	Optional minima for each dimension
<i>maxima</i>	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

<i>data</i>	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

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

<i>resolution</i>	Vector of resolution values for each dimension
<i>perc_overlap</i>	Percentage of overlap between hypercubes
<i>minima</i>	Optional minima for each dimension
<i>maxima</i>	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

<i>data_cover_factory</i>	factory creating the data cover to use
<i>complex_factory</i>	factory creating the complex to use
<i>clusterer</i>	the clustering algorithm to use
<i>projection</i>	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

<i>data</i>	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:

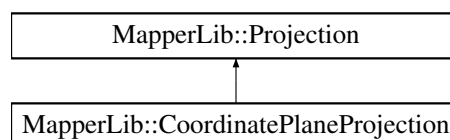
- [typedefs.h](#)

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:

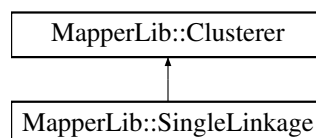
- [typedefs.h](#)

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:

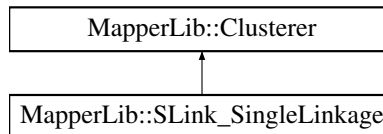
- [SingleLinkage.h](#)
- [SingleLinkage.cpp](#)

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 agglomerative 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

<i>num_clusters</i>	Optional integer specifying the desired number of clusters.
<i>distance_threshold</i>	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

<i>data</i>	A constant reference to a Matrix containing the data points to be clustered.
<i>data_filter</i>	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.](#)

```

00001 //
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;
00027
00036         [[nodiscard]] virtual std::vector<Simplex> generate(std::vector<MapperCluster> const& clusters)
00037         const = 0;
00038     };
00039
00047     class CechComplex final : public Complex {
00048     static constexpr size_t NUM_THREADS = 8;
00049     public:
00059         CechComplex(DataCover const& data_cover, Dimension max_dimension);
00060
00071         [[nodiscard]] std::vector<Simplex> generate(std::vector<MapperCluster> const& clusters) const
00072         override;
00073     private:
00074         using Iterator = std::vector<MapperCluster>::const_iterator;
00075         using ClustersByCube = std::vector<std::vector<MapperCluster const*>>;
00076
00083         class SimplexComputer {
00084         Iterator _begin;
00085         Iterator _end;
00086         ClustersByCube& _clusters_by_cube;
00087         Dimension const _dim;
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:
00095         static inline int id_counter = 0;
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
00127     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&
00146     clusters, Dimension k) const;
00156     static std::generator<std::vector<size_t>> generate_k_subsets_of_range(size_t index_max, size_t k);
00157
00167     static bool check_cluster_intersection(std::vector<MapperCluster const*> const& all_clusters,
00168     std::vector<size_t> const& relevant_indices);
00178     static std::vector<size_t> get_vector_intersection(std::vector<size_t> vec_1, std::vector<size_t>
00179     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 =
00203     0;
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_CECHECOMPLEX_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 {

```

```

00012
00013 using Cluster = std::vector<PointId>;
00014 using ClusterAssignment = std::vector<Cluster>;
00015
00020 class Clusterer {
00021 public:
00022     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::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.](#)

```
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,
00037         std::optional<Vector> minima = std::nullopt,
00038         std::optional<Vector> maxima = std::nullopt
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;
00091
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;
00124     double const _perc_overlap;
00125     Matrix const& _data;
00126     size_t const _data_dimension;
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:
00150     DataCoverFactory(
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,
00198         std::optional<Vector> maxima = std::nullopt
00199     );
00200
00206     [[nodiscard]] std::unique_ptr<DataCover> create_data_cover(Matrix const& data) const;
00207
00208 private:
00209     std::vector<size_t> _resolution;
00210     std::optional<size_t> _single_resolution;
00211     double _perc_overlap;
00212     std::optional<Vector> _minima;
00213     std::optional<Vector> _maxima;
00214 };
00215
00216
00217
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)
```

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.](#)

```
00001 //
00002 // Created by jakob on 23.06.24.
00003 //
00004
00005 #ifndef LINALGHELPER_H
00006 #define LINALGHELPER_H
00007 #include <ostream>
00008
00009 #include "typedefs.h"
00010
00011 namespace MapperLib {
00012     Scalar euclididan_distance(Vector const& vec1, Vector const& vec2);
00013
00014     Scalar maximum_distance(Vector const& vec1, Vector const& vec2);
00015
00016     bool check_data_equal_dimension(Matrix const& mat);
00017
00018     Dimension get_data_dimension(Matrix const& mat);
00019
00020     void print(Vector const& vec);
00021 } // Helper
00022
00023 std::ostream& operator<<(std::ostream& os, MapperLib::Vector const& vec);
00024 std::ostream& operator<<(std::ostream& os, MapperLib::Matrix const& mat);
00025
00026 #endif //LINALGHELPER_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"
00011
00012
00013
00014 namespace MapperLib {
00015     class ComplexFactory;
00016     class Complex;
00017     class Clusterer;
00018     class Projection;
00019
00020     class Mapper {
00021     public:
00022         Mapper(
00023             std::shared_ptr<DataCoverFactory> data_cover_factory,
00024             std::shared_ptr<ComplexFactory> complex_factory,
00025             std::shared_ptr<Clusterer> clusterer,
00026             std::shared_ptr<Projection> projection
00027         );
00028
00029         [[nodiscard]] std::vector<Simplex> map(Matrix const& data) ;
00030
00031     private:
00032         std::shared_ptr<DataCoverFactory> _data_cover_factory;
00033         std::shared_ptr<ComplexFactory> _complex_factory;
00034         std::shared_ptr<Clusterer> _clusterer;
00035         std::shared_ptr<Projection> _projection;
00036         std::unique_ptr<DataCover> _data_cover;
00037         std::unique_ptr<Complex> _complex;
00038     };
00039
00040 } // Mapper
00041
00042 #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.](#)

```
00001 //
00002 // Created by jgier on 27.06.2024.
00003 //
00004
00005 #ifndef MAPPER_PROJECTION_H
00006 #define MAPPER_PROJECTION_H
00007 #include <memory>
00008
00009 #include "typedefs.h"
00010
00011 namespace MapperLib{
00012 class Projection {
00013 public:
00014     virtual ~Projection() = default;
00015     [[nodiscard]] virtual Matrix project(Matrix const& data) const = 0;
00016 };
00017
00018 class CoordinatePlaneProjection final : public Projection{
00019 public:
00020     explicit CoordinatePlaneProjection(std::vector<Dimension> dimensions);
00021     [[nodiscard]] static std::shared_ptr<Projection> make_shared(std::vector<Dimension> dimensions);
00022     [[nodiscard]] Matrix project (Matrix const& data) const override;
00023 private:
00024     std::vector<Dimension> _dimensions;
00025 };
00026
00027 } // Mapper
00028
00029 #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.](#)

```
00001 //
00002 // Created by jakob on 14.06.24.
00003 //
00004
00005 #ifndef SINGLELINKAGE_H
00006 #define SINGLELINKAGE_H
00007 #include <limits>
00008 #include <memory>
00009 #include <optional>
00010
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
00026     ClusterAssignment predict(Matrix const &data, std::vector<PointId> data_filter) override;
00027
00028 private:
00029     [[nodiscard]] static Scalar min_distance(
00030         Matrix const& distances,
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,
00034         std::optional<Scalar> distance_threshold);
00034
00044     ClusterAssignment predict(Matrix const &data, std::vector<PointId> data_filter) override;
00045 private:
00046     std::optional<size_t> _num_clusters;
00047     std::optional<Scalar> _distance_threshold;
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::IntegerCubeld](#) = 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.](#)

```
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>;
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
00021 struct Simplex{
00022     std::vector<PointId> points;
00023     [[nodiscard]] std::vector<PointId> get_points(){ return points; }
00024     [[nodiscard]] Dimension dimension() const { return points.size() - 1;}
00025     [[nodiscard]] size_t num_nodes() const { return points.size(); }
00026     [[nodiscard]] PointId operator[] (size_t index) const {return points[index]; }
00027 };
00028
00029 struct MapperCluster{
00030     std::vector<PointId> points;
00031     ClusterId cluster_id;
00032 }
```

```
00044     IntegerCubeId integer_cube_id;
00045 };
00046 }
00047
00048
00049
00050 inline std::ostream& operator<<(std::ostream& os, std::vector<size_t> const& vec)
00051 {
00052     os << "[";
00053     for(auto const elt: vec) {
00054         os << elt << ", ";
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 {
00067     os << "[";
00068     for(auto const& elt: vec) {
00069         os << elt << ", ";
00070     }
00071     os << "\b\b]";
00072     return os;
00073 }
00074
00075
00076 #endif //TYPEDEFS_H
```

Index

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

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

- 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](#)