ReadClustering

The C++ implmentation for ReadClustering

Generated by Doxygen 1.8.11

Contents

1	Read	dCluste	ring											1
2	Clas	s Index												3
	2.1	Class	List					 	 	 	 	 		3
3	File	Index												5
	3.1	File Lis	st					 	 	 	 	 		5
4	Clas	s Docu	mentatior	1										7
	4.1	Datase	et Struct R	eference .				 	 	 	 	 		7
		4.1.1	Detailed	Description				 	 	 	 	 		7
		4.1.2	Member	Data Docur	nentation			 	 	 	 	 		7
			4.1.2.1	array				 	 	 	 	 		7
			4.1.2.2	dataVec				 	 	 	 	 		7
			4.1.2.3	groupAgg	regate .			 	 	 	 	 		7
			4.1.2.4	maxGroup	·			 	 	 	 	 		8
			4.1.2.5	neighborV	'ec			 	 	 	 	 		8
			4.1.2.6	numOfEle	ments .			 	 	 	 	 		8
	4.2	ReadC	Clustering (Class Refer	ence			 	 	 	 	 		8
		4.2.1	Detailed	Description				 	 	 	 	 		9
		4.2.2	Construc	ctor & Destr	uctor Doc	umenta	tion .	 	 	 	 	 		9
			4.2.2.1	ReadClus	tering().			 	 	 	 	 		9
			4.2.2.2	\sim ReadCli	ustering()			 	 	 	 	 		9
		4.2.3	Member	Function De	ocumenta	ition .		 	 	 	 	 		9
			4.2.3.1	computeE	valuation	()		 	 	 	 	 		9

iv CONTENTS

			4.2.3.2	computeEvaluation(std::unordered_map< string, std::vector< int > >::const_← iterator &iter)	10
			4.2.3.3	getEvaluation(const char *fileName)	11
			4.2.3.4	performSVD(MatrixXf &cArray, const Eigen::MatrixXf &data, const int &Row, const int &Column, int &PC_Number)	12
			4.2.3.5	readData(const char *fileName)	12
			4.2.3.6	writeAnalysis()	15
		4.2.4	Member	Data Documentation	15
			4.2.4.1	activityList	15
			4.2.4.2	ds	15
			4.2.4.3	isPBF	15
			4.2.4.4	maxElements	15
			4.2.4.5	timeList	15
5	File	Docum	entation		17
	5.1	main.c	pp File Re	eference	17
		5.1.1	Function	Documentation	17
			5.1.1.1	main(int argc, char *argv[])	17
	5.2	ReadC	Clustering.	cpp File Reference	18
		5.2.1	Variable	Documentation	18
			5.2.1.1	TOR_1	18
	5.3	ReadC	Clustering.l	h File Reference	18
	5.4	READ	ME.md Fil	e Reference	19
Inc	lex				21

ReadClustering

The program will read from a generated .vtk file with clustering results (i.e., geometric coordinates and cluster labels of integral lines) to re-calculate the clustering evaluation metrics of the current result in case of miscalculated result.

2 ReadClustering

Class Index

21	Class	l iet

Here are the classes	s, stı	ruct	s, u	ınio	ns	anc	ri b	nte	rfa	ces	s v	vitl	h b	rie	ef c	est	scri	ipti	on	ıs:								
Dataset																												-
ReadClustering																										 		8

4 Class Index

File Index

3.1 File List

Here is a list of all files with brief descriptions:

main.cpp	17
ReadClustering.cpp	18
ReadClustering.h	18

6 File Index

Class Documentation

4.1 Dataset Struct Reference

#include <ReadClustering.h>

Public Attributes

- std::vector< std::vector< float > > dataVec
- Eigen::MatrixXf array
- $\bullet \ \ unordered_map{<} \ string, \ std::vector{<} \ int >{>} \ groupAggregate$
- unordered_map< string, int > maxGroup
- int numOfElements
- std::vector< std::vector< int > > neighborVec

4.1.1 Detailed Description

Definition at line 27 of file ReadClustering.h.

4.1.2 Member Data Documentation

4.1.2.1 Eigen::MatrixXf Dataset::array

Definition at line 33 of file ReadClustering.h.

4.1.2.2 std::vector < std::vector < float > Dataset::dataVec

Definition at line 30 of file ReadClustering.h.

 $\textbf{4.1.2.3} \quad unordered_map{<}string, std::vector{<}int{>}> Dataset::groupAggregate}$

Definition at line 36 of file ReadClustering.h.

4.1.2.4 unordered_map<string, int> Dataset::maxGroup

Definition at line 39 of file ReadClustering.h.

4.1.2.5 std::vector<std::vector<int>> Dataset::neighborVec

Definition at line 44 of file ReadClustering.h.

4.1.2.6 int Dataset::numOfElements

Definition at line 42 of file ReadClustering.h.

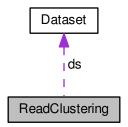
The documentation for this struct was generated from the following file:

· ReadClustering.h

4.2 ReadClustering Class Reference

#include <ReadClustering.h>

Collaboration diagram for ReadClustering:



Public Member Functions

- ReadClustering ()
- virtual ∼ReadClustering ()
- void getEvaluation (const char *fileName)

Private Member Functions

- void readData (const char *fileName)
- void computeEvaluation ()
- void writeAnalysis ()
- void computeEvaluation (std::unordered_map< string, std::vector< int > >::const_iterator &iter)
- void performSVD (MatrixXf &cArray, const Eigen::MatrixXf &data, const int &Row, const int &Column, int &PC_Number)

Private Attributes

- std::vector< string > activityList
- std::vector< string > timeList
- · Dataset ds
- int maxElements
- bool isPBF

4.2.1 Detailed Description

Definition at line 52 of file ReadClustering.h.

4.2.2 Constructor & Destructor Documentation

```
4.2.2.1 ReadClustering::ReadClustering ( )
```

Definition at line 20 of file ReadClustering.cpp.

```
20 {
21  // TODO Auto-generated constructor stub
22  
23 }
```

4.2.2.2 ReadClustering::~ReadClustering() [virtual]

Definition at line 29 of file ReadClustering.cpp.

```
29
30    // TODO Auto-generated destructor stub
31 }
```

4.2.3 Member Function Documentation

4.2.3.1 void ReadClustering::computeEvaluation() [private]

Definition at line 257 of file ReadClustering.cpp.

4.2.3.2 void ReadClustering::computeEvaluation (std::unordered_map< string, std::vector< int > >::const_iterator & iter) [private]

Definition at line 273 of file ReadClustering.cpp.

```
274 {
275
         if(iter->second.empty())
276
              return;
277
278
         ds.neighborVec.clear();
         ds.neighborVec = std::vector<std::vector<int> >(ds.maxGroup[iter->first]);
279
280
         const std::vector<int>& groupOfNorm = iter->second;
281
         const int& groupSize = groupOfNorm.size();
282
283
284
         Silhouette sil;
285
         ValidityMeasurement vm;
286
287
         int totalNum = 0;
288
         for(int i=0;i<groupSize;++i)</pre>
289
         {
290
              if(groupOfNorm[i]<0)</pre>
291
             continue;
ds.neighborVec[groupOfNorm[i]].push_back(i);
292
293
              ++totalNum;
294
295
         std::cout << totalNum << std::endl;</pre>
         /* the 'PCA' option */
296
297
         if(strcmp("PCA", iter->first.c_str())==0)
298
299
              IOHandler::expandArray(ds.array,ds.dataVec,3,maxElements);
              std::cout << "expanded!" << std::endl;
300
301
              Eigen::MatrixXf cArray;
302
              int PC_number;
303
              std::cout << ds.maxGroup[iter->first] << std::endl;</pre>
              performSVD(cArray, ds.array, ds.array.rows(), ds.
304
       array.cols(), PC_number);
305
             sil.computeValue(cArray, groupOfNorm, ds.maxGroup[iter->first],
       isPBF);
306
              vm.computeValue(ds.array, groupOfNorm);
307
308
         else
309
310
              /* count from "norm" for norm option */
             const int& normOption = std::atoi(iter->first.substr(4).c_str());
std::cout << "This is norm " << normOption << std::endl;</pre>
311
312
313
              //if(normOption!=4 && normOption!=15)
314
              // return;
315
316
             if (normOption==17)
                  IOHandler::expandArray(ds.array,ds.dataVec,3,
317
       maxElements);
318
                  IOHandler::sampleArray(ds.array,ds.dataVec,3,
319
       maxElements);
320
321
              MetricPreparation object(ds.array.rows(), ds.array.cols());
             object.preprocessing(ds.array, ds.array.rows(), ds.array.cols(), normOption);
/* if the dataset is not PBF, then should record distance matrix for Gamma matrix compution */
322
323
324
              if(!isPBF)
325
326
                  deleteDistanceMatrix(ds.array.rows());
327
328
                  std::ifstream distFile(("../dataset/"+to_string(normOption)).c_str(), ios::in);
329
                  if(distFile.fail())
330
331
                       distFile.close();
                       getDistanceMatrix(ds.array, normOption, object);
std::ofstream distFileOut(("../dataset/"+to_string(normOption)).c_str(), ios::out);
332
333
334
                       for(int i=0;i<ds.array.rows();++i)</pre>
335
336
                            for(int j=0; j<ds.array.rows(); ++j)</pre>
337
                                distFileOut << distanceMatrix[i][i] << " ";
338
339
340
                           distFileOut << std::endl;
341
342
                       distFileOut.close();
343
344
                  else
345
346
                       std::cout << "read distance matrix..." << std::endl;
347
```

```
348
                     distanceMatrix = new float*[ds.array.rows()];
349
                 #pragma omp parallel for schedule(static) num_threads(8)
350
                      for (int i = 0; i < ds.array.rows(); ++i)</pre>
351
352
                          distanceMatrix[i] = new float[ds.array.rows()];
353
                     int i=0, j;
354
355
                     string line;
356
                     stringstream ss;
357
                     while (getline (distFile, line))
358
359
                          i=0;
                          ss.str(line);
360
361
                          while (ss>>line)
362
363
                              <u>if</u>(i==j)
364
                                  distanceMatrix[i][j]=0;
                              else
365
366
                                  distanceMatrix[i][j] = std::atof(line.c_str());
367
                              ++j;
368
369
                          ++i;
                          ss.str("");
370
371
                          ss.clear();
372
373
                     distFile.close();
374
375
                 std::cout << "Distance between 0 and 1 is " << distanceMatrix[0][1] << std::endl;
376
377
378
            sil.computeValue(normOption,ds.array,ds.array.rows(),ds.
      array.cols(), groupOfNorm, object,
379
                                   ds.maxGroup[iter->first], isPBF, ds.
380
            vm.computeValue(normOption, ds.array, groupOfNorm, object, isPBF);
381
382
383
        /* compute the entropy */
384
        float entropy = 0, prob;
385
        for (int i = 0; i < ds.neighborVec.size(); ++i)</pre>
386
387
             if (ds.neighborVec[i].size()>0)
388
389
                 prob = float(ds.neighborVec[i].size())/float(totalNum);
390
                 entropy+=prob*log2f(prob);
391
392
393
        entropy = -entropy/log2f(ds.maxGroup[iter->first]);
std::cout << "Entropy is " << entropy << std::endl;</pre>
394
395
396
397
        activityList.push_back("Silhouette for "+iter->first+" is: ");
398
        timeList.push_back(to_string(sil.sAverage));
399
        activityList.push_back("Gamma statistic for "+iter->first+" is: ");
400
        timeList.push_back(to_string(sil.gammaStatistic));
401
402
403
        activityList.push_back("Entropy for "+iter->first+" is: ");
404
        timeList.push_back(to_string(entropy));
405
        activityList.push_back("DB Index for "+iter->first+" is: ");
406
407
        timeList.push_back(to_string(sil.dbIndex));
408
409
        activityList.push_back("Validity measurement on "+iter->first+" is: ");
410
        stringstream fc_ss;
411
        fc_ss << vm.f_c;</pre>
412
        timeList.push_back(fc_ss.str());
413
414
        /* record labeling information */
415
        // IOHandler::generateGroups(ds.neighborVec, iter->first+"_storage");
416 }
```

4.2.3.3 void ReadClustering::getEvaluation (const char * fileName)

Definition at line 39 of file ReadClustering.cpp.

```
40 {
41    int isPBFInput;
42    std::cout << "Is it a PBF dataset? 1.Yes, 0.No." << std::endl;
43    std::cin >> isPBFInput;
```

```
44
       assert(isPBFInput==1||isPBFInput==0);
       isPBF = (isPBFInput==1);
46
47
       /* read data into ds */
       std::cout << fileName << std::endl;
48
       readData(fileName);
49
50
       /\star compute the evaluation \star/
52
       computeEvaluation();
53
       /* output the result to text file */
54
55
       writeAnalysis();
56 }
```

4.2.3.4 void ReadClustering::performSVD (MatrixXf & cArray, const Eigen::MatrixXf & data, const int & Row, const int & Column, int & PC_Number) [private]

Definition at line 428 of file ReadClustering.cpp.

```
430 {
431
        MatrixXf SingVec;
         VectorXf meanTrajectory(Column);
432
433
        Eigen::MatrixXf temp = data;
434
435 #pragma omp parallel for schedule(static) num_threads(8) 436 for (int i = 0; i < Column; ++i)
437
438
             meanTrajectory(i) = temp.transpose().row(i).mean();
439
440 #pragma omp parallel for schedule(static) num_threads(8)
441
         for (int i = 0; i < Row; ++i)
442
443
             temp.row(i) = temp.row(i)-meanTrajectory.transpose();
444
445
         /\star perform SVD decomposition for temp \star/
        JacobisVD<MatrixXf> svd(temp, ComputeThinU | ComputeThinV);
//const VectorXf& singValue = svd.singularValues();
446
447
448
        SingVec = svd.matrixV();
449
450
         /* compute new attribute space based on principal component */
451
        MatrixXf coefficient = temp*SingVec;
452
         /\star~ decide first r dorminant PCs with a threshold \star/
453
         const float& varianceSummation = coefficient.squaredNorm();
454
        float tempSum = 0.0;
        const float& threshold = TOR_1*varianceSummation;
455
456
457
         for (int i = 0; i < Column; ++i)
458
459
             tempSum+=(coefficient.transpose().row(i)).squaredNorm();
460
             if(tempSum>threshold)
461
             {
462
                 PC Number = i;
463
                 break;
464
465
466
467
        cArray = MatrixXf(Row, PC_Number);
468 #pragma omp parallel for schedule(static) num threads(8)
469
        for (int i = 0; i < PC_Number; ++i)</pre>
470
471
             cArray.transpose().row(i) = coefficient.transpose().row(i);
472
473
        std::cout << "SVD completed!" << std::endl;</pre>
474
475
         SingVec.transposeInPlace();
477 }
```

4.2.3.5 void ReadClustering::readData (**const char** * **fileName**) [private]

Definition at line 75 of file ReadClustering.cpp.

```
76 {
77
       ifstream fin(fileName, ios::in);
78
       if(!fin)
79
       {
            std::cout << "Error opening the file!" << std::endl;</pre>
80
81
           exit(1):
82
83
84
       Eigen::MatrixXf vertexCoordinate;
8.5
       /* omit first four lines */
86
87
       string line:
       for (int i=0; i<5; ++i)</pre>
88
89
90
           getline(fin, line);
91
       /\star split the string into three parts \star/
92
93
       stringstream ss(line);
94
       ss >> line;
       ss >> line;
95
96
97
       ss.str(std::string());
98
99
       /* get how many vertex inside */
100
        const int& vertexCount = atoi(line.c_str());
101
102
        vertexCoordinate = Eigen::MatrixXf(vertexCount, 3);
103
104
        /\star read in vertex coordinates \star/
105
        for(int i=0;i<vertexCount;++i)</pre>
106
107
             /* read one line */
108
            getline(fin, line);
109
             /\star split and analyze the string \star/
110
             ss.str(line);
            for (int j=0; j<3; ++j)
111
112
            {
113
                 ss >> line;
114
                 vertexCoordinate(i,j) = atof(line.c_str());
115
116
            ss.str(std::string());
        }
117
118
119
        /* get how many streamlines you'll have */
120
        getline(fin, line);
121
122
        ss.str(line);
123
        ss >> line;
        ss >> line;
124
125
126
        ss.str(std::string());
127
128
        ds.numOfElements = atoi(line.c_str());
129
130
        ds.dataVec = std::vector<std::vector<float> >(ds.numOfElements);
131
132
        /* read vertex coordinates into dataVec */
133
134
        maxElements = INT_MIN;
135
136
        int vertexNum, index;
        for(int i=0;i<ds.numOfElements;++i)</pre>
137
138
139
             getline(fin,line);
140
            ss.str(line);
141
142
            /* explicate vertex count in each line */
143
            ss>>line:
144
            vertexNum = atoi(line.c_str());
145
146
             /* assign memory */
147
            std::vector<float>& tempVec = ds.dataVec[i];
148
            tempVec = std::vector<float>(vertexNum*3);
149
150
151
            maxElements = std::max(maxElements, vertexNum*3);
152
153
             for(int j=0; j<vertexNum; ++j)</pre>
154
155
                 ss>>line:
                 index = atoi(line.c_str());
156
                 for (int k=0; k<3; ++k)
157
158
                     tempVec[3*j+k] = vertexCoordinate(index,k);
159
160
161
             ss.str(std::string());
162
```

```
for (int i=0;i<3;++i)</pre>
163
164
165
             getline(fin,line);
166
167
         for(int i=0;i<vertexCount;++i)</pre>
168
169
             getline(fin,line);
170
171
        std::size_t found_int, found, found_scalars;
172
173
        int normOption, totalLine, groupIndex;
174
        while (getline (fin, line))
175
176
             found_int = line.find("int");
177
             found_scalars = line.find("SCALARS");
178
             /\star has int, should be group information \star/
179
             if(found_int!=std::string::npos && found_scalars!=std::string::npos)
180
             {
181
                 ss.str(line);
182
                 ss>>line;
183
                 ss>>line;
184
185
                 string norm_choice;
                 if(strcmp(line.substr(0,3).c_str(), "PCA")==0)
186
187
                     norm_choice="PCA";
188
189
                 {
                      found = line.find("_");
found_int = line.find("norm");
190
191
                      if (found==std::string::npos)
192
193
194
                          norm_choice = line;
195
196
                      else if(found_int>found)
197
                          norm_choice = line.substr(found_int);
198
199
200
                      else if(found_int<found)</pre>
201
                      {
202
                          norm_choice = line.substr(found_int, found);
203
2.04
                 }
205
206
                 getline(fin,line);
207
208
                 std::vector<int> tempGroup(ds.numOfElements);
209
210
                 for (int i=0;i<ds.numOfElements;++i)</pre>
211
212
                      for (int j=0; j<ds.dataVec[i].size()/3;++j)</pre>
                          getline(fin,line);
213
214
                      tempGroup[i] = atoi(line.c_str());
215
216
                 ds.groupAggregate.insert(std::make_pair(norm_choice, tempGroup));
217
218
                 totalLine = vertexCount+2;
                 for(int i=0; i<totalLine; ++i)</pre>
220
221
                      getline(fin,line);
222
223
             }
224
225
226
        fin.close();
227
228
        /\star compute cluster number \star/
229
        std::vector<int> groupArray;
230
231
        int max num;
232
        std::unordered_map<string, std::vector<int> >::const_iterator iter;
233
         for(iter=ds.groupAggregate.begin(); iter!=ds.
      groupAggregate.end(); ++iter)
234
             groupArray = iter->second;
max_num = -1;
235
236
237
             if (groupArray.empty())
238
                 continue;
239
240
             index = groupArray.size();
241
             for(int j=0;j<index;++j)</pre>
242
243
                 max_num = std::max(max_num, groupArray[j]);
244
245
             max_num+=1;
246
             ds.maxGroup.insert(make_pair(iter->first, max_num));
2.47
         }
248
```

```
249 }
```

4.2.3.6 void ReadClustering::writeAnalysis () [private]

Definition at line 62 of file ReadClustering.cpp.

```
63 {
64    /* write information */
65    IOHandler::generateReadme(activityList,timeList);
66
67 }
```

4.2.4 Member Data Documentation

4.2.4.1 std::vector<string> ReadClustering::activityList [private]

Definition at line 79 of file ReadClustering.h.

4.2.4.2 Dataset ReadClustering::ds [private]

Definition at line 89 of file ReadClustering.h.

4.2.4.3 bool ReadClustering::isPBF [private]

Definition at line 99 of file ReadClustering.h.

4.2.4.4 int ReadClustering::maxElements [private]

Definition at line 94 of file ReadClustering.h.

4.2.4.5 std::vector<string> ReadClustering::timeList [private]

Definition at line 84 of file ReadClustering.h.

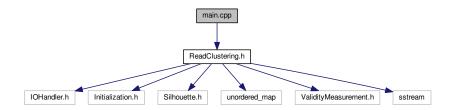
The documentation for this class was generated from the following files:

- · ReadClustering.h
- ReadClustering.cpp

File Documentation

5.1 main.cpp File Reference

#include "ReadClustering.h"
Include dependency graph for main.cpp:



Functions

• int main (int argc, char *argv[])

5.1.1 Function Documentation

5.1.1.1 int main (int argc, char * argv[])

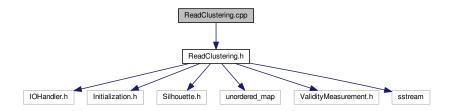
Definition at line 10 of file main.cpp.

```
11 {
12     if(argc!=2)
13     {
14         std::cout << "Error for argument input!" << std::endl;
15         exit(1);
16     }
17
18     /* create ReadClustering object and get evaluation */
19
20     ReadClustering rc;
21
22     rc.getEvaluation(argv[1]);
23
24     return 0;</pre>
```

18 File Documentation

5.2 ReadClustering.cpp File Reference

#include "ReadClustering.h"
Include dependency graph for ReadClustering.cpp:



Variables

• const float & TOR 1 = 0.999

5.2.1 Variable Documentation

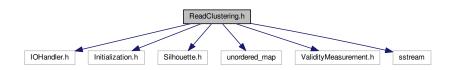
5.2.1.1 const float& TOR_1 = 0.999

Definition at line 14 of file ReadClustering.cpp.

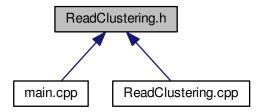
5.3 ReadClustering.h File Reference

```
#include "IOHandler.h"
#include "Initialization.h"
#include "Silhouette.h"
#include <unordered_map>
#include "ValidityMeasurement.h"
#include <sstream>
```

Include dependency graph for ReadClustering.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct Dataset
- class ReadClustering

5.4 README.md File Reference

20 File Documentation

Index

\sim ReadClustering ReadClustering, 9
activityList ReadClustering, 15
Dataset, 7
computeEvaluation ReadClustering, 9
dataVec
Dataset, 7
Dataset, 7 array, 7
dataVec, 7
groupAggregate, 7
maxGroup, 7
neighborVec, 8
numOfElements, 8
ds
ReadClustering, 15
getEvaluation
ReadClustering, 11
groupAggregate
Dataset, 7
isPBF
ReadClustering, 15
main
main.cpp, 17 main.cpp, 17
main, 17
maxElements
ReadClustering, 15
maxGroup
Dataset, 7
neighborVec
Dataset, 8
numOfElements
Dataset, 8
performSVD
ReadClustering, 12
README.md, 19
ReadClustering, 8
\sim ReadClustering, 9

```
activityList, 15
    computeEvaluation, 9
    ds, 15
    getEvaluation, 11
    isPBF, 15
    maxElements, 15
    performSVD, 12
    ReadClustering, 9
    readData, 12
    timeList, 15
    writeAnalysis, 15
ReadClustering.cpp, 18
    TOR_1, 18
ReadClustering.h, 18
readData
    ReadClustering, 12
TOR_1
    ReadClustering.cpp, 18
timeList
    ReadClustering, 15
writeAnalysis
    ReadClustering, 15
```