

gift
2.0

Generated by Doxygen 1.8.12

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Namespace Documentation	7
4.1	gift Namespace Reference	7
4.1.1	Typedef Documentation	8
4.1.1.1	IntArrayList	8
4.1.1.2	IntList	8
4.1.1.3	name2IndexHash	8
4.1.1.4	nameList	8
4.1.1.5	numericMatrix	8
4.1.2	Function Documentation	8
4.1.2.1	author("Songpeng Zu")	8
4.1.2.2	BoolToString(bool b)	8
4.1.2.3	email("zusongpeng@gmail.com")	9
4.1.2.4	functionThread(func useFun, int thread, EM *point)	9
4.1.2.5	getIndexFromHash(const name2IndexHash &name2Index, const nameList fromNameList, IntList &toIntIndexList, nameList &existNameList)	9
4.1.2.6	helpGift()	9

4.1.2.7	Matrix2Fingerpints(const std::string inputFile, IntArrayList &getFp, std::string delims)	11
4.1.2.8	Matrix2FingerprintsByColumn(const std::string inputFile, IntArrayList &getFP, int rowNum, std::string delims)	11
4.1.2.9	outRecord(parameters &EMparameters)	12
4.1.2.10	printIntArrayList(const IntArrayList &fromIntArrayList)	12
4.1.2.11	printMatrix(const numericMatrix &fromMatrix)	13
4.1.2.12	readMatrix(const std::string inputFile, numericMatrix &getMat, std::string delims)	13
4.1.2.13	readName2IndexHash(const nameList fromNameList, name2IndexHash &name2Index)	14
4.1.2.14	readNameListFromFile(const std::string inputFile, nameList &tonameList)	14
4.1.2.15	readNameMatrixFromFile(const std::string inputFile, nameList &tonameList, IntArrayList &getFP, std::string delims)	14
4.1.2.16	rowColFile(const std::string inputFile, rowCol &matrixRec, std::string delims)	15
4.1.2.17	updateTime("2016-03-06")	16
4.1.2.18	version("gift-2.0")	16
4.1.2.19	writeMatrix(const std::string outFileName, numericMatrix &resultMat, std::string delims)	16
4.1.3	Variable Documentation	16
4.1.3.1	domain2proteinList	16
4.1.3.2	drug2proteinList	16
4.1.3.3	drug2subList	16
4.1.3.4	drugName2Index	16
4.1.3.5	drugNameList	16
4.1.3.6	drugSub2proteinSubMatrix	17
4.1.3.7	drugSubNameList	17
4.1.3.8	loglikelyArray	17
4.1.3.9	observedDrug2ProteinMatrix	17
4.1.3.10	predictDrug2SubList	17
4.1.3.11	predictDrugNameList	17
4.1.3.12	predictDrugNameList_WithSubs	17
4.1.3.13	predictProtein2SubList	17
4.1.3.14	predictProteinNameList	17
4.1.3.15	predictProteinNameList_WithSubs	17
4.1.3.16	protein2domainList	18
4.1.3.17	proteinName2Index	18
4.1.3.18	proteinNameList	18
4.1.3.19	proteinSubNameList	18
4.1.3.20	recLogLeastNum	18
4.1.3.21	sub2drugList	18
4.1.3.22	vardrugSub2proteinSubMatrix	18

5	Class Documentation	19
5.1	gift::EM Class Reference	19
5.1.1	Constructor & Destructor Documentation	20
5.1.1.1	EM(parameters ¶m)	20
5.1.1.2	~EM()	20
5.1.2	Member Function Documentation	20
5.1.2.1	EStep()	20
5.1.2.2	EStep(int)	20
5.1.2.3	EStepThread(int threadNth)	21
5.1.2.4	functionThread(void(EM::*function)(int), int thread)	21
5.1.2.5	iterdrugSub2ProteinSub(int drugIndex, int proteinIndex)	21
5.1.2.6	MStep()	21
5.1.2.7	MStep(int)	22
5.1.2.8	MStepThread(int threadNth)	22
5.1.2.9	outTrainResult()	22
5.1.2.10	outTrainVariance()	23
5.1.2.11	predictDrugs()	23
5.1.2.12	predictDrugsProteins()	24
5.1.2.13	predictDrugsProteinsWithSubs()	24
5.1.2.14	predictDrugsWithSubs()	25
5.1.2.15	predictDrugsWithSubsProteins()	26
5.1.2.16	predictDrugsWithSubsProteinsWithSubs()	26
5.1.2.17	predictEM()	27
5.1.2.18	predictProteins()	28
5.1.2.19	predictProteinsWithSubs()	28
5.1.2.20	recLoglikely()	29
5.1.2.21	setLoglikely(double logscore)	29
5.1.2.22	setPointerDrug2Protein(IntArrayList &d2p)	29
5.1.2.23	setPointerDrug2Sub(IntArrayList &d2s)	29
5.1.2.24	setPointerDrugSub2ProteinSub(numericMatrix &ds2ps)	30

5.1.2.25	setPointerProtein2Sub(IntArrayList &p2s)	30
5.1.2.26	trainEM()	30
5.1.2.27	varEM()	31
5.2	gift::parameters Class Reference	31
5.2.1	Constructor & Destructor Documentation	32
5.2.1.1	parameters(const std::string)	32
5.2.2	Member Function Documentation	34
5.2.2.1	InitDrugName2Index()	34
5.2.2.2	InitDrugSub2ProteinSub()	35
5.2.2.3	InitDrugSubNameList()	35
5.2.2.4	InitObservedDrug2ProteinMatrix()	36
5.2.2.5	InitPredictParameters()	36
5.2.2.6	InitProteinName2Index()	36
5.2.2.7	InitProteinSubNameList()	37
5.2.2.8	InitVarDrugSub2ProteinSub()	37
5.2.2.9	setDomainNum(int number)	37
5.2.2.10	setDrugNum(int number)	37
5.2.2.11	setProteinNum(int number)	37
5.2.2.12	setSubNum(int number)	37
5.2.3	Member Data Documentation	37
5.2.3.1	alphaEB	37
5.2.3.2	betaEB	38
5.2.3.3	chemfpRec	38
5.2.3.4	CPIsRec	38
5.2.3.5	domainNum	38
5.2.3.6	drug2proteinFileName	38
5.2.3.7	drug2subFileName	38
5.2.3.8	drugNameListFile	38
5.2.3.9	drugNum	38
5.2.3.10	drugSub2proteinSubFileName	38

5.2.3.11	drugSubNameListFile	38
5.2.3.12	fn	38
5.2.3.13	fp	38
5.2.3.14	inputDelims	38
5.2.3.15	iterNum	39
5.2.3.16	loglikelyRecord	39
5.2.3.17	outDrugSub2ProteinSubFileName	39
5.2.3.18	outPredictCPIsFileName	39
5.2.3.19	outputDelims	39
5.2.3.20	outRecordFileName	39
5.2.3.21	outVarDrugSub2proteinSubFileName	39
5.2.3.22	predictDrugsFileName	39
5.2.3.23	predictDrugsFileName_WithSubs	39
5.2.3.24	predictProteinsFileName	39
5.2.3.25	predictProteinsFileName_WithSubs	39
5.2.3.26	protein2subFileName	39
5.2.3.27	proteinfpRec	39
5.2.3.28	proteinNameListFile	39
5.2.3.29	proteinNum	40
5.2.3.30	proteinSubNameListFile	40
5.2.3.31	subNum	40
5.2.3.32	task	40
5.2.3.33	thread	40
5.3	gift::rowCol Class Reference	40
5.3.1	Constructor & Destructor Documentation	40
5.3.1.1	rowCol(int row, int col)	40
5.3.1.2	rowCol()	41
5.3.2	Member Data Documentation	41
5.3.2.1	colNum	41
5.3.2.2	rowNum	41
6	File Documentation	43
6.1	EMGift.cpp File Reference	43
6.2	functionsGift.cpp File Reference	43
6.3	gift.hpp File Reference	44
6.4	main.cpp File Reference	46
6.4.1	Function Documentation	46
6.4.1.1	main(int argc, char **argv)	46
6.5	parameterGift.cpp File Reference	47
	Index	49

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

gift	7
--------------------------------	-------------------

Chapter 2

Class Index

2.1 Class List

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

gift::EM	19
gift::parameters	31
gift::rowCol	40

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

EMGift.cpp	43
functionsGift.cpp	43
gift.hpp	44
main.cpp	46
parameterGift.cpp	47

Chapter 4

Namespace Documentation

4.1 gift Namespace Reference

Classes

- class [EM](#)
- class [parameters](#)
- class [rowCol](#)

Typedefs

- typedef std::vector< int > [IntList](#)
- typedef std::vector< std::vector< int > > [IntArrayList](#)
- typedef std::vector< std::vector< double > > [numericMatrix](#)
- typedef std::map< std::string, int > [name2IndexHash](#)
- typedef std::vector< std::string > [nameList](#)

Functions

- int [Matrix2Fingerprints](#) (const std::string inputFile, [IntArrayList](#) &getFp, std::string delims)
- int [readMatrix](#) (const std::string inputFile, [numericMatrix](#) &getMat, std::string delims)
- int [rowColFile](#) (const std::string inputFile, [rowCol](#) &matrixRec, std::string delims)
- int [writeMatrix](#) (const std::string outFileName, [numericMatrix](#) &resultMat, std::string delims)
- int [printIntArrayList](#) (const [IntArrayList](#) &fromIntArrayList)
- int [printMatrix](#) (const [numericMatrix](#) &fromMatrix)
- int [readNameListFromFile](#) (const std::string inputFile, [nameList](#) &tonameList)
- int [readNameMatrixFromFile](#) (const std::string inputFile, [nameList](#) &tonameList, [IntArrayList](#) &getFP, std::string delims)
- int [readName2IndexHash](#) (const [nameList](#) fromNameList, [name2IndexHash](#) &name2Index)
- int [getIndexFromHash](#) (const [name2IndexHash](#) &name2Index, const [nameList](#) fromNameList, [IntList](#) &toIndexList, [nameList](#) &existNameList)
- int [helpGift](#) ()
- int [outRecord](#) ([parameters](#) &EMparameters)
- int [Matrix2FingerprintsByColumn](#) (const std::string inputFile, [IntArrayList](#) &getFP, int rowNum, std::string delims)
- const std::string [author](#) ("Songpeng Zu")
- const std::string [email](#) ("zusongpeng@gmail.com")
- const std::string [version](#) ("gift-2.0")
- const std::string [updateTime](#) ("2016-03-06")
- template<typename func >
int [functionThread](#) (func useFun, int thread, [EM](#) *point)
- const char * [BoolToString](#) (bool b)

Variables

- const int [recLogLeastNum](#) = 5
- [IntArrayList](#) [drug2proteinList](#)
- [IntArrayList](#) [drug2subList](#)
- [IntArrayList](#) [sub2drugList](#)
- [IntArrayList](#) [protein2domainList](#)
- [IntArrayList](#) [domain2proteinList](#)
- [numericMatrix](#) [drugSub2proteinSubMatrix](#)
- [numericMatrix](#) [observedDrug2ProteinMatrix](#)
- [numericMatrix](#) [vardrugSub2proteinSubMatrix](#)
- [std::vector< double >](#) [loglikelyArray](#)
- [name2IndexHash](#) [drugName2Index](#)
- [name2IndexHash](#) [proteinName2Index](#)
- [nameList](#) [drugNameList](#)
- [nameList](#) [proteinNameList](#)
- [nameList](#) [drugSubNameList](#)
- [nameList](#) [proteinSubNameList](#)
- [nameList](#) [predictDrugNameList](#)
- [nameList](#) [predictProteinNameList](#)
- [nameList](#) [predictDrugNameList_WithSubs](#)
- [nameList](#) [predictProteinNameList_WithSubs](#)
- [IntArrayList](#) [predictDrug2SubList](#)
- [IntArrayList](#) [predictProtein2SubList](#)

4.1.1 Typedef Documentation

4.1.1.1 `typedef std::vector<std::vector<int> > gift::IntArrayList`

4.1.1.2 `typedef std::vector<int> gift::IntList`

4.1.1.3 `typedef std::map<std::string,int> gift::name2IndexHash`

4.1.1.4 `typedef std::vector<std::string> gift::nameList`

4.1.1.5 `typedef std::vector<std::vector<double> > gift::numericMatrix`

4.1.2 Function Documentation

4.1.2.1 `const std::string gift::author ("Songpeng Zu")`

Referenced by [helpGift\(\)](#), and [outRecord\(\)](#).

4.1.2.2 `const char* gift::BoolToString (bool b) [inline]`

Referenced by [gift::parameters::parameters\(\)](#).

```

35                                     {
36     return b ? "true" : "false";
37 } // end of function BoolToString

```


4.1.2.3 `const std::string gift::email ("zusongpeng@gmail.com")`

Referenced by `helpGift()`, and `outRecord()`.

4.1.2.4 `template<typename func > int gift::functionThread (func useFun, int thread, EM * point)`

Referenced by `gift::EM::EStep()`, and `gift::EM::MStep()`.

```

103                                     {
104     boost::thread * y;
105     boost::thread_group * x = new boost::thread_group;
106     for(int i=0;i<thread;++i){
107         y = new boost::thread(useFun,point,i);
108         x->add_thread(y);
109     } // end of loop i
110     x->join_all();
111     delete x;
112     return 0;
113 } // end of function.

```

4.1.2.5 `int gift::getIndexFromHash (const name2IndexHash & name2Index, const nameList fromNameList, IntList & toIndexList, nameList & existNameList)`

Referenced by `gift::EM::predictDrugs()`, `gift::EM::predictDrugsProteins()`, `gift::EM::predictDrugsProteinsWithSubs()`, `gift::EM::predictDrugsWithSubsProteins()`, and `gift::EM::predictProteins()`.

```

282                                     {
283     for(const auto fromName : fromNameList){
284         if (name2Index.find(fromName) != name2Index.end()){
285             existNameList.push_back(fromName);
286             toIndexList.push_back( (name2Index.find(fromName))>>second);
287         } else {
288             std::cout<<"Cannot find the key " << fromName
289                 <<" fromNameList. Continue..." <<std::endl;
290         } // end of if else
291     } // end of loop fromNameList
292     return 0;
293 } // end of function

```

4.1.2.6 `int gift::helpGift ()`

References `author()`, `email()`, `updateTime()`, and `version()`.

Referenced by `main()`.

```

295     {
296     // Output gift information and useness to standard output.
297     // Basic information about gift.
298     std::cout<<"Gift is used to predict compound-protein interactions based on "
299         <<std::endl;
300     std::cout<<"their substructures interactions."<<std::endl;
301     std::cout<<"It is also used to infer the substructures interactions from"
302         <<std::endl;
303     std::cout<<" the known drug-protein interactions."<<std::endl;
304     std::cout<<"If you want to know more about gift, please read the paper: "
305         <<std::endl;
306     std::cout<<"Global Optimization-based Inference of Chemogenomic Features "
307         <<std::endl;
308     std::cout<<"from Drug-Target Interactions, which is published " <<std::endl;
309     std::cout<<"on Bioinformatics, 2015. " <<std::endl;
310     std::cout<<"Author: " <<author<<std::endl;
311     std::cout<<"Email: " <<email<<std::endl;

```

```

312     std::cout<<"Current version: "<<version<<std::endl;
313     std::cout<<"Last update time: "<<updateTime<<std::endl;
314     std::cout<<"You can get the C++ source code from: "<<std::endl;
315     std::cout<<"https://github.com/songpeng/GIFT" << std::endl;
316     std::cout<<std::endl;
317
318     //Input parameters
319     std::cout<<"--help | -h to show the help information of gift."<<std::endl;
320     std::cout<<"--version | -v to show the version information of gift."
321         <<std::endl;
322     std::cout<<"Gift need one configure file for its running."<<std::endl;
323     std::cout<<"Please use --config to tell gift the configure file name."
324         <<std::endl;
325     std::cout<<"The content in the configure file are listed below: "<<std::endl;
326
327     // configure file information.
328     std::cout<<"[INPUT DATA FILE NAMES]" <<std::endl;
329     std::cout<<"drug2proteinFileName=<string> : "
330         <<"file name for drug protein interactions" <<std::endl;
331     std::cout<<"drug2subFilename=<string> : "
332         <<"file name for drug to substructure" <<std::endl;
333     std::cout<<"protein2subFileName=<string> : "
334         <<"file name for protein to substructure" << std::endl;
335     std::cout<<"drugSub2proteinSubfilename=<string> : "
336         <<"file name for drugSub to proteinSub interaction probability."
337         <<std::endl;
338     std::cout<<"drugNameListFile=<string> : "
339         <<"file name for drug names" << std::endl;
340     std::cout<<"drugSubNameListFile=<string> : "
341         <<"file name for drug substructures names." << std::endl;
342     std::cout<<"proteinNameListFile=<string> : "
343         <<"file name for protein names" << std::endl;
344     std::cout<<"proteinSubNameListFile=<string> : "
345         <<"file name for protein substructures names." << std::endl;
346
347     std::cout<<"[INPUT PARAMETERS FOR EM ALGORITHM]" <<std::endl;
348     std::cout<<"alphaEB=<double> : "
349         <<"parameter for Empricial Bayesian estimates for initEM."
350         <<std::endl;
351     std::cout<<"betaEB=<double> : "
352         <<"parameter for Empricial Bayesian estimates for initEM."
353         <<std::endl;
354     std::cout<<"fp=<double> : "<<"false positive rate"<<std::endl;
355     std::cout<<"fn=<double> : "<<"false negative rate"<<std::endl;
356     std::cout<<"threadNum=<int> : "<<"thread number for EM." <<std::endl;
357     std::cout<<"EMIterationNum=<int> : "<<"iteration numbers/steps for EM."<<std::endl;
358     std::cout<<"task=<string> : "<<"run gift for [train] or [predict]."<<std::endl;
359     std::cout<<"loglikelyRecord=<string> : "<<
360         "record [true] or not [false] the loglikely in in every step."<<std::endl;
361     std::cout<<"inputDelims=<string> : "
362         <<"sep character for input files, such as '\t',',' "<<std::endl;
363
364     std::cout<<"[INPUT FILE VERSION INFORMATION]"<<std::endl;
365     std::cout<<"chemFingerPrintRecord=<string> : "
366         <<"source and version of chemical fingerprints."<<std::endl;
367     std::cout<<"proteinFingerPrintRecord=<string> : "
368         <<"source and version of protein fingerprints/domains."<<std::endl;
369     std::cout<<"comProteinInteractionRecord=<string> : "
370         <<"source and version of compound-protein interactions."<<std::endl;
371
372     std::cout<<"[INPUT FILE NAME FOR PREDICTION]"<<std::endl;
373     std::cout<<"predictDrugsFileName=<string> : "
374         <<"file name for drug names used for prediction by gift."<<std::endl;
375     std::cout<<"predictProteinsFileName=<string> : "
376         <<"file name for protein names used for prediction by gift."<<std::endl;
377     std::cout<<"predictDrugsFileName_WithSubs=<string> : "
378         <<"file name for drug names together with their substructures."
379         <<std::endl;
380     std::cout<<"predictProteinsFileName_WithSubs=<string> : "
381         <<"file name for protein names together with their substructures."
382         <<std::endl;
383
384     std::cout<<"[OUTPUT FILE NAME AND FORMAT]"<<std::endl;
385     std::cout<<"outputDelims=<string> : "
386         <<"sep character for output files." <<std::endl;
387     std::cout<<"outRecordFileName=<string> : "<<"file name for output records."
388         <<std::endl;
389     std::cout<<"outPredictCPIsFileName=<string> : "
390         <<"file name for output CPIs." <<std::endl;
391     std::cout<<"outDrugSub2ProteinSubFileName=<string> : "
392         <<"file name for output drugSub2proteinSub matrix."<<std::endl;
393     std::cout<<"outVarDrugSub2proteinSubFileName=<string> : "
394         <<"file name for output variance of drugSub2proteinSub." <<std::endl;
395
396     return 0;
397 } // end of function

```

4.1.2.7 int gift::Matrix2Fingerprints (const std::string *inputFile*, IntArrayList & *getFp*, std::string *delims*)

Referenced by gift::parameters::parameters().

```

15         {
16     std::ifstream input (inputFile, std::ios::in);
17     std::string line;
18     std::vector<std::string> array;
19     std::vector<int> tempRec;
20     while (std::getline(input, line)) {
21         boost::algorithm::split(array, line, boost::is_any_of(delims));
22         int arraylen = array.size();
23         for (int i=0; i<arraylen; ++i) {
24             if (array[i].compare("1") == 0) {
25                 tempRec.push_back(i);
26             } // end of if
27         } // end of for
28         getFp.push_back(tempRec);
29         tempRec.clear();
30     } // end of while
31
32     // input.exceptions(std::ifstream::failbit | std::ifstream::badbit);
33     // try {
34     //     input.open(inputFile, std::ifstream::in);
35     //     if (input.peek() == std::ifstream::traits_type::eof()){
36     //         std::cerr <<inputFile <<" is empty. " <<std::endl;
37     //         return 1;
38     //     } // end of if
39     //     while (std::getline(input, line)) {
40     //         boost::algorithm::split(array, line, boost::is_any_of(delims));
41     //         int arraylen = array.size();
42     //         for (int i=0; i<arraylen; ++i) {
43     //             if (array[i].compare("1") == 0) {
44     //                 tempRec.push_back(i);
45     //             } // end of if
46     //         } // end of for
47     //         getFp.push_back(tempRec);
48     //         tempRec.clear();
49     //     } // end of while
50     // } catch (std::ifstream::failure e) {
51     //     std::cerr <<"Exceptions open/read file " <<inputFile<<std::endl;
52     //     return 1;
53     // } // end of catch
54
55     return 0;
56 } // end of function.

```

4.1.2.8 int gift::Matrix2FingerprintsByColumn (const std::string *inputFile*, IntArrayList & *getFP*, int *rowNum*, std::string *delims*)

Referenced by gift::parameters::parameters().

```

430         {
431     std::ifstream input (inputFile, std::ios::in);
432     std::string line;
433     std::vector<std::string> array;
434     int linenum = 0;
435     // init getFP first.
436     std::cout<<"Init getFP IntArrayList..."<<std::endl;
437     std::vector<int> tmpArray;
438     for(int i=0; i<rowNum; ++i) {
439         getFP.push_back(tmpArray);
440     } // end of loop
441     std::cout<<"End of Init getFP IntArrayList..."<<std::endl;
442
443     while (std::getline(input, line)) {
444         boost::algorithm::split(array, line, boost::is_any_of(delims));
445         int arraylen = array.size();
446         for (int i=0; i<arraylen; ++i) {
447             if (array[i].compare("1") == 0) {
448                 getFP[i].push_back(linenum);
449             } // end of if
450         } // end of loop for i
451         linenum += 1;
452         array.clear();
453     } // end of while for file read.

```

```

454
455 // input.exceptions(std::ifstream::failbit | std::ifstream::badbit);
456 // try {
457 //     input.open(inputFile, std::ifstream::in);
458 //     if(input.peek() == std::ifstream::traits_type::eof()){
459 //         std::cerr<< inputFile << " is empty. "<<std::endl;
460 //         return 1;
461 //     } // end of if
462
463 //     // init getFP first.
464 //     for(int i=0;i<rowNum;++i) {
465 //         std::vector<int> tmpArray;
466 //         getFP.push_back(tmpArray);
467 //     } // end of loop
468
469 //     // read file.
470 //     while (std::getline(input, line)){
471 //         boost::algorithm::split(array, line, boost::is_any_of(delims));
472 //         int arraylen = array.size();
473 //         for (int i=0;i<arraylen;++i){
474 //             if (array[i].compare("1") == 0){
475 //                 getFP[i].push_back(linenum);
476 //             } // end of if
477 //         } // end of loop for i
478 //         linenum += 1;
479 //     } // end of while for file read.
480 // } catch (std::ifstream::failure e) {
481 //     std::cerr<<"Exceptions open/read file "<<inputFile<<std::endl;
482 //     return 1;
483 // } // end of catch
484 return 0;
485 } // end of function

```

4.1.2.9 int gift::outRecord (parameters & EMparameters)

References author(), email(), loglikelyArray, gift::parameters::loglikelyRecord, gift::parameters::outRecordFileName, gift::parameters::task, updateTime(), and version().

Referenced by main().

```

399
400 std::ofstream output (EMparameters.outRecordFileName, std::ofstream::out);
401 if (!output.is_open()){
402     std::cerr<<"Error open file "<<EMparameters.outRecordFileName<<std::endl;
403     return 1;
404 } // end of if
405 // Basic information about gift.
406 output<<"The author of gift is " << author <<std::endl;
407 output<<"Contact information: " << email <<std::endl;
408 output<<"Current gift's version is "<< version <<std::endl;
409 output<<"Update time is " << updateTime <<std::endl;
410 // Running information.
411 std::chrono::system_clock::time_point timePos =
412     std::chrono::system_clock::now();
413 std::time_t timePosT = std::chrono::system_clock::to_time_t(timePos);
414 output<<"The job destination is "<<EMparameters.task;
415 output<<" , which is finished at "<<std::ctime(&timePosT) <<std::endl;
416 if (EMparameters.task.compare("train") == 0){
417     if (EMparameters.loglikelyRecord){
418         output<<"The loglikelihood values are followed: "<<std::endl;
419         for(const auto m : loglikelyArray){
420             output<<m<<std::endl;
421         } // end of loop for m.
422     } // end of if
423 } // end of if
424 output.close();
425 return 0;
426 } // end of function

```

4.1.2.10 int gift::printIntArrayList (const IntArrayList & fromIntArrayList)

Referenced by gift::parameters::parameters().

```

167                                     {
168     int lineNum = fromIntArrayList.size();
169     for(int i=0;i<lineNum;++i){
170         if (fromIntArrayList[i].empty()){
171             std::cerr<<"[ERROR]: Row "<<i<<" is empty for printing... "<<std::endl;
172             return 1;
173         } // end of if
174         for(const auto m : fromIntArrayList[i]){std::cout<<m<<" ";}
175         std::cout<<std::endl;
176     } // end of loop for i
177     return 0;
178 } // end of function.

```

4.1.2.11 int gift::printMatrix (const numericMatrix & fromMatrix)

```

180                                     {
181     int rowNum = fromMatrix.size();
182     int colNum = fromMatrix[0].size();
183     for(int i=0;i<rowNum;++i){
184         for(int j=0;j<colNum;++j){
185             std::cout<<fromMatrix[i][j]<<" ";
186         } // end of loop for j
187         std::cout<<std::endl;
188     } // end of loop for i
189     return 0;
190 } // end of function

```

4.1.2.12 int gift::readMatrix (const std::string inputFile, numericMatrix & getMat, std::string delims)

Referenced by gift::parameters::InitDrugSub2ProteinSub().

```

59                                     {
60     std::ifstream input (inputFile, std::ios::in);
61     std::string line;
62     std::vector<std::string> array;
63     std::vector<double> tempRec;
64     while(std::getline(input,line)){
65         boost::algorithm::split(array,line,boost::is_any_of(delims));
66         int arraylen = array.size();
67         for(int i=0;i<arraylen;++i){
68             std::string::size_type* idx = 0;
69             tempRec.push_back(std::stod(array[i], idx));
70         } // end of for
71         getMat.push_back(tempRec);
72         tempRec.clear();
73     } // end of while
74
75     // input.exceptions(std::ifstream::failbit | std::ifstream::badbit);
76     // try {
77     //     input.open(inputFile, std::ifstream::in);
78     //     if (input.peek() == std::ifstream::traits_type::eof()){
79     //         std::cerr <<inputFile <<" is empty. " <<std::endl;
80     //     } // end of if
81     //     while(std::getline(input,line)){
82     //         boost::algorithm::split(array,line,boost::is_any_of(delims));
83     //         int arraylen = array.size();
84     //         for(int i=0;i<arraylen;++i){
85     //             std::string::size_type* idx = 0;
86     //             tempRec.push_back(std::stod(array[i], idx));
87     //         } // end of for
88     //         getMat.push_back(tempRec);
89     //         tempRec.clear();
90     //     } // end of while
91     // } catch (std::ifstream::failure e) {
92     //     std::cerr <<"Exceptions open/read file "<<inputFile<<std::endl;
93     //     return 1;
94     // } // end of catch
95     return 0;
96 } // end of function.

```

4.1.2.13 `int gift::readName2IndexHash (const nameList fromNameList, name2IndexHash & name2Index)`

Referenced by `gift::parameters::InitDrugName2Index()`, and `gift::parameters::InitProteinName2Index()`.

```

265                                     {
266     // fromNameList should be in order.
267     if (fromNameList.empty()){
268         std::cerr<<"The fromNameList is empty. "<<std::endl;
269         return 1;
270     } // end of if
271     int recordIndex = 0;
272     for(const auto fromName : fromNameList){
273         name2Index.insert(std::pair<std::string,int>(fromName,recordIndex));
274         ++recordIndex;
275     } // end of loop fromNameList
276     return 0;
277 } // end of function

```

4.1.2.14 `int gift::readNameListFromFile (const std::string inputFile, nameList & tonameList)`

Referenced by `gift::parameters::InitDrugName2Index()`, `gift::parameters::InitDrugSubNameList()`, `gift::parameters::InitPredictParameters()`, `gift::parameters::InitProteinName2Index()`, and `gift::parameters::InitProteinSubNameList()`.

```

192                                     {
193     // each line in the file represents one name.
194     // line should end with "\n", not "[\r\t]\n"
195     std::ifstream input (inputFile, std::ios::in);
196     std::string line;
197     while (std::getline(input,line)) {
198         tonameList.push_back(line);
199     } // end of while
200     input.close();
201
202     // input.exceptions(std::ifstream::failbit | std::ifstream::badbit);
203     // try {
204     //     input.open(inputFile,std::ifstream::in);
205     //     if (input.peek() == std::ifstream::traits_type::eof() ){
206     //         std::cerr << inputFile <<" is empty. "<<std::endl;
207     //         return 1;
208     //     } // end of if
209     //     while (std::getline(input,line)) {
210     //         tonameList.push_back(line);
211     //     } // end of while
212     //     input.close();
213     // } catch (std::ifstream::failure e) {
214     //     std::cerr<<"Exceptions open/read file " << inputFile<<std::endl;
215     // } // end of try catch
216     return 0;
217 } // end of function

```

4.1.2.15 `int gift::readNameMatrixFromFile (const std::string inputFile, nameList & tonameList, IntArrayList & getFP, std::string delims)`

Referenced by `gift::parameters::InitPredictParameters()`.

```

220                                     {
221     std::ifstream input (inputFile, std::ios::in);
222     std::string line;
223     std::vector<std::string> array;
224     std::vector<int> tempRec;
225     while (std::getline(input,line)) {
226         boost::algorithm::split(array,line,boost::is_any_of(delims));
227         tonameList.push_back(array[0]); // first column is name.
228         int arraylen = array.size();
229         for(int i=1;i<arraylen;++i){
230             if(array[i].compare("1") == 0) {
231                 tempRec.push_back(i-1); // Use i-1, since first column is name.

```

```

232         } // end of if
233     } // end of loop for i.
234     getFP.push_back(tempRec);
235     tempRec.clear();
236 } // end of while
237
238 // input.exceptions(std::ifstream::failbit | std::ifstream::badbit);
239 // try {
240 //     input.open(inputFile, std::ifstream::in);
241 //     if (input.peek() == std::ifstream::traits_type::eof()) {
242 //         std::cerr<<inputFile<<" is empty. "<<std::endl;
243 //         return 1;
244 //     } // end of if
245 //     while(std::getline(input,line)) {
246 //         boost::algorithm::split(array,line,boost::is_any_of(delims));
247 //         tonameList.push_back(array[0]); // first column is name.
248 //         int arraylen = array.size();
249 //         for(int i=1;i<arraylen;++i){
250 //             if(array[i].compare("1") == 0) {
251 //                 tempRec.push_back(i-1); // Use i-1, since first column is name.
252 //             } // end of if
253 //         } // end of loop for i.
254 //         getFP.push_back(tempRec);
255 //         tempRec.clear();
256 //     } // end of while
257 // } catch (std::ifstream::failure e) {
258 //     std::cerr<<"Exceptions open/read file "<<inputFile<<std::endl;
259 //     return 1;
260 // } // end of catch
261 return 0;
262 } // end of function

```

4.1.2.16 int gift::rowColFile (const std::string *inputFile*, rowCol & *matrixRec*, std::string *delims*)

References gift::rowCol::colNum, and gift::rowCol::rowNum.

Referenced by gift::parameters::parameters().

```

99         {
100     std::ifstream input (inputFile, std::ios::in);
101     std::string line;
102     int count = 0;
103     std::vector<std::string> array;
104
105     std::getline(input,line);
106     ++count;
107     boost::algorithm::split(array, line, boost::is_any_of(delims));
108     matrixRec.colNum = array.size();
109     // string getline func over istream.
110     while(std::getline(input,line)){
111         // QUESTION: how about empty line?
112         ++count;
113     } // end of while
114     input.close();
115
116     matrixRec.rowNum = count;
117     matrixRec.colNum = array.size();
118
119     // input.exceptions(std::ifstream::failbit | std::ifstream::badbit);
120     // try {
121     //     input.open(inputFile,std::ifstream::in);
122     //     if (input.peek() == std::ifstream::traits_type::eof()){
123     //         std::cerr <<inputFile<<" is empty."<<std::endl;
124     //         return 1;
125     //     }
126
127     //     std::getline(input,line);
128     //     ++count;
129     //     boost::algorithm::split(array, line, boost::is_any_of(delims));
130     //     matrixRec.colNum = array.size();
131     //     // string getline func over istream.
132     //     while(std::getline(input,line)){
133     //         // QUESTION: how about empty line?
134     //         ++count;
135     //     } // end of while
136     //     input.close();
137
138     //     matrixRec.rowNum = count;
139     //     matrixRec.colNum = array.size();

```

```

140     // } catch (std::ifstream::failure e) {
141     //     std::cerr << "Exceptions open/read file "<<inputFile<<std::endl;
142     //     return 1;
143     // } // end of catch
144     return 0;
145 } // end of function.

```

4.1.2.17 const std::string gift::updateTime ("2016-03-06")

Referenced by helpGift(), main(), and outRecord().

4.1.2.18 const std::string gift::version ("gift-2.0")

Referenced by helpGift(), main(), and outRecord().

4.1.2.19 int gift::writeMatrix (const std::string *outFileName*, numericMatrix & *resultMat*, std::string *delims*)

Referenced by gift::EM::outTrainResult(), and gift::EM::outTrainVariance().

```

148     {
149     std::ofstream output (outFileName, std::ios::out);
150     if (output.is_open()) {
151         using boost::algorithm::join;
152         using boost::adaptors::transformed;
153         for (numericMatrix::iterator it = resultMat.begin();
154              it != resultMat.end(); ++it) {
155             output << join(*it | transformed(static_cast<std::string(*)>(double)>
156                                             (std::to_string) ), delims);
157             output<<std::endl;
158         } // end of for
159         output.close();
160     } else {
161         std::cerr<< "Error opening file " <<outFileName<<std::endl;
162         return 1;
163     } // end of if else
164     return 0;
165 } // end of function

```

4.1.3 Variable Documentation

4.1.3.1 IntArrayList gift::domain2proteinList

4.1.3.2 IntArrayList gift::drug2proteinList

4.1.3.3 IntArrayList gift::drug2subList

4.1.3.4 name2IndexHash gift::drugName2Index

Referenced by gift::EM::predictDrugs(), gift::EM::predictDrugsProteins(), and gift::EM::predictDrugsProteinsWithSubs().

4.1.3.5 nameList gift::drugNameList

Referenced by gift::EM::predictProteins(), and gift::EM::predictProteinsWithSubs().

4.1.3.6 numericMatrix gift::drugSub2proteinSubMatrix

Referenced by gift::EM::MStep().

4.1.3.7 nameList gift::drugSubNameList

4.1.3.8 std::vector< double > gift::loglikelyArray

Referenced by outRecord().

4.1.3.9 numericMatrix gift::observedDrug2ProteinMatrix

Referenced by gift::EM::EStep().

4.1.3.10 IntArrayList gift::predictDrug2SubList

Referenced by gift::EM::predictDrugsWithSubs(), gift::EM::predictDrugsWithSubsProteins(), and gift::EM::predict↔DrugsWithSubsProteinsWithSubs().

4.1.3.11 nameList gift::predictDrugNameList

Referenced by gift::EM::predictDrugs(), gift::EM::predictDrugsProteins(), gift::EM::predictDrugsProteinsWithSubs(), and gift::EM::predictEM().

4.1.3.12 nameList gift::predictDrugNameList_WithSubs

Referenced by gift::EM::predictDrugsWithSubs(), gift::EM::predictDrugsWithSubsProteins(), gift::EM::predict↔DrugsWithSubsProteinsWithSubs(), and gift::EM::predictEM().

4.1.3.13 IntArrayList gift::predictProtein2SubList

Referenced by gift::EM::predictDrugsProteinsWithSubs(), gift::EM::predictDrugsWithSubsProteinsWithSubs(), and gift::EM::predictProteinsWithSubs().

4.1.3.14 nameList gift::predictProteinNameList

Referenced by gift::EM::predictDrugsProteins(), gift::EM::predictDrugsWithSubsProteins(), gift::EM::predictEM(), and gift::EM::predictProteins().

4.1.3.15 nameList gift::predictProteinNameList_WithSubs

Referenced by gift::EM::predictDrugsProteinsWithSubs(), gift::EM::predictDrugsWithSubsProteinsWithSubs(), gift::EM::predictEM(), and gift::EM::predictProteinsWithSubs().

4.1.3.16 `IntArrayList` `gift::protein2domainList`**4.1.3.17** `name2IndexHash` `gift::proteinName2Index`

Referenced by `gift::EM::predictDrugsProteins()`, `gift::EM::predictDrugsWithSubsProteins()`, and `gift::EM::predict↵Proteins()`.

4.1.3.18 `nameList` `gift::proteinNameList`

Referenced by `gift::EM::predictDrugs()`, and `gift::EM::predictDrugsWithSubs()`.

4.1.3.19 `nameList` `gift::proteinSubNameList`**4.1.3.20** `const int` `gift::recLogLeastNum = 5`

Referenced by `gift::EM::trainEM()`.

4.1.3.21 `IntArrayList` `gift::sub2drugList`**4.1.3.22** `numericMatrix` `gift::vardrugSub2proteinSubMatrix`

Chapter 5

Class Documentation

5.1 gift::EM Class Reference

```
#include <gift.hpp>
```

Public Member Functions

- [EM](#) ([parameters](#) ¶m)
- [~EM](#) ()
- int [setPointerDrug2Sub](#) ([IntArrayList](#) &d2s)
- int [setPointerProtein2Sub](#) ([IntArrayList](#) &p2s)
- int [setPointerDrug2Protein](#) ([IntArrayList](#) &d2p)
- int [setPointerDrugSub2ProteinSub](#) ([numericMatrix](#) &ds2ps)
- double [iterdrugSub2ProteinSub](#) (int drugIndex, int proteinIndex)
- int [functionThread](#) (void(EM::*function)(int), int thread)
- void [EStepThread](#) (int threadNth)
- int [EStep](#) ()
- int [EStep](#) (int)
- void [MStepThread](#) (int threadNth)
- int [MStep](#) ()
- int [MStep](#) (int)
- double [recLoglikely](#) ()
- int [setLoglikely](#) (double logscore)
- int [trainEM](#) ()
- int [predictDrugs](#) ()
- int [predictProteins](#) ()
- int [predictDrugsWithSubs](#) ()
- int [predictProteinsWithSubs](#) ()
- int [predictDrugsWithSubsProteinsWithSubs](#) ()
- int [predictDrugsProteinsWithSubs](#) ()
- int [predictDrugsWithSubsProteins](#) ()
- int [predictDrugsProteins](#) ()
- int [predictEM](#) ()
- int [varEM](#) ()
- int [outTrainResult](#) ()
- int [outTrainVariance](#) ()

5.1.1 Constructor & Destructor Documentation

5.1.1.1 gift::EM::EM(parameters & param) [inline]

```

202      : loglikelyRecord(param.loglikelyRecord)
203      , fn(param.fn)
204      , fp(param.fp)
205      , thread(param.thread)
206      , iterNum(param.iterNum)
207      , drugNum(param.drugNum)
208      , subNum(param.subNum)
209      , domainNum(param.domainNum)
210      , proteinNum(param.proteinNum)
211      , task(param.task)
212      , drug2sub(&drug2subList)
213      , sub2drug(&sub2drugList)
214      , protein2sub(&protein2domainList)
215      , sub2protein(&domain2proteinList)
216      , drug2protein(&drug2proteinList)
217      , drugSub2proteinSub(&drugSub2proteinSubMatrix)
218      , observedDrug2Protein(&observedDrug2ProteinMatrix)
219      , vardrugSub2proteinSub(&vardrugSub2proteinSubMatrix)
220      , loglikely(&loglikelyArray)
221      , predictDrugsFileName(param.predictDrugsFileName)
222      , predictProteinsFileName(param.predictProteinsFileName)
223      , predictDrugsFileName_WithSubs(param.predictDrugsFileName_WithSubs)
224      , predictProteinsFileName_WithSubs(param.predictProteinsFileName_WithSubs)
225      , outputDelims(param.outputDelims)
226      , outRecordFileName(param.outRecordFileName)
227      , outPredictCPIsFileName(param.outPredictCPIsFileName)
228      , outDrugSub2ProteinSubFileName(param.outDrugSub2ProteinSubFileName)
229      , outVarDrugSub2proteinSubFileName(param.outVarDrugSub2proteinSubFileName)
230      { } // end of constructor.

```

5.1.1.2 gift::EM::~~EM() [inline]

```

232 { } // end of default destruction.

```

5.1.2 Member Function Documentation

5.1.2.1 int gift::EM::EStep()

References EStepThread(), and gift::functionThread().

Referenced by trainEM().

```

48      {
49      // both of them works.
50      return gift::functionThread(&EM::EStepThread,thread,this);
51      //return functionThread(&EM::EStepThread, thread);
52      } // end of function

```

5.1.2.2 int gift::EM::EStep(int)

References iterdrugSub2ProteinSub(), and gift::observedDrug2ProteinMatrix.

```

37      {
38      for(int i=0;i<drugNum;++i){
39      for(int j=0;j<proteinNum;++j){
40      double tmp = iterdrugSub2ProteinSub(i,j);
41      observedDrug2ProteinMatrix[i][j] = (1-fn)*(1-tmp) + fp*tmp;
42      //(*observedDrug2Protein).at(i).at(j) = (1-fn)*(1-tmp) + fp*tmp;
43      } // end of for loop j
44      } // end of for loop i
45      return 0;
46      } // end of function

```

5.1.2.3 void gift::EM::EStepThread (int *threadNth*)

References iterdrugSub2ProteinSub().

Referenced by EStep().

```

26                                     {
27     //std::cout<<"This is thread "<<threadNth<<" for EStep..."<<std::endl;
28     for(int i=threadNth;i<drugNum;i+=thread){
29         for(int j=0;j<proteinNum;++j){
30             double tmp = iterdrugSub2ProteinSub(i,j);
31             (*observedDrug2Protein).at(i).at(j) = (1-fn)*(1-tmp) + fp*tmp;
32         } // end of for loop j
33     } // end of for loop i
34     //return 0;
35 } // end of function

```

5.1.2.4 int gift::EM::functionThread (void(EM::*)(int) *function*, int *thread*) [inline]

```

258                                     {
259     boost::thread *y;
260     boost::thread_group * x = new boost::thread_group;
261     for(int i=0;i<thread;++i){
262         y = new boost::thread(function,this,i);
263         x->add_thread(y);
264     } // end of loop for i
265     x->join_all();
266     delete x;
267     return 0;
268 } // end of function.

```

5.1.2.5 double gift::EM::iterdrugSub2ProteinSub (int *drugIndex*, int *proteinIndex*)

Referenced by EStep(), EStepThread(), predictDrugs(), predictDrugsProteins(), predictProteins(), recLoglikely(), and varEM().

```

12                                     {
13     double tmp = 0;
14     if ((*drug2sub)[drugIndex].empty() || (*protein2sub)[proteinIndex].empty()){
15         std::cerr<<"[ERROR]: some row of drug2sub or protein2sub is empty..."<<std::endl;
16         return 1;
17     } // end of if
18     for(auto const & m : (*drug2sub)[drugIndex]){
19         for(auto const & n : (*protein2sub)[proteinIndex]){
20             tmp += log(1 - (*drugSub2proteinSub)[m][n]);
21         } // end of loop n
22     } // end of loop m
23     return exp(tmp);
24 } // end of function.

```

5.1.2.6 int gift::EM::MStep ()

References gift::functionThread(), and MStepThread().

Referenced by trainEM().

```

93     {
94         // both of them works.
95         return gift::functionThread(&EM::MStepThread,thread,this);
96         //return functionThread(&EM::MStepThread,thread);
97     } // end of function

```

5.1.2.7 int gift::EM::MStep (int)

References gift::drugSub2proteinSubMatrix.

```

73         {
74     for(int i=0;i<subNum;++i){
75         for(int j=0;j<domainNum;++j){
76             double tmp = 0;
77             for(auto &m : (*sub2drug)[i]){
78                 for(auto &n : (*sub2protein)[j]){
79                     double observed = (*observedDrug2Protein)[m][n];
80                     tmp = (std::find((*drug2protein)[m].begin(),(*drug2protein)[m].end(),n)
81                         != (*drug2protein)[m].end() ) ? (1-fn)/observed : fn/(1-observed);
82                 } // end of loop n
83             } // end of loop m
84             int tmpNum = (*sub2drug)[i].size() + (*sub2protein)[j].size();
85             tmp = log((*drugSub2proteinSub)[i][j]) + log(tmp/tmpNum);
86             drugSub2proteinSubMatrix[i][j] = exp(tmp);
87             // (*drugSub2proteinSub).at(i).at(j) = exp(tmp);
88         } // end of loop j
89     } // end of for loop i
90     return 0;
91 } // end of function

```

5.1.2.8 void gift::EM::MStepThread (int threadNth)

Referenced by MStep().

```

54         {
55     //std::cout<<"This is thread "<<threadNth<<" for MStep..."<<std::endl;
56     for(int i=threadNth;i<subNum;i+=thread){
57         for(int j=0;j<domainNum;++j){
58             double tmp = 0;
59             for(auto const &m : (*sub2drug)[i]){
60                 for(auto const &n : (*sub2protein)[j]){
61                     double observed = (*observedDrug2Protein)[m][n];
62                     tmp = (std::find((*drug2protein)[m].begin(),(*drug2protein)[m].end(),n)
63                         != (*drug2protein)[m].end() ) ? (1-fn)/observed : fn/(1-observed);
64                 } // end of loop n
65             } // end of loop m
66             int tmpNum = (*sub2drug)[i].size() + (*sub2protein)[j].size();
67             tmp = log((*drugSub2proteinSub)[i][j]) + log(tmp/tmpNum);
68             (*drugSub2proteinSub)[i][j] = exp(tmp);
69         } // end of loop j
70     } // end of for loop i
71 } // end of function

```

5.1.2.9 int gift::EM::outTrainResult ()

References gift::writeMatrix().

Referenced by main().

```

565         {
566     writeMatrix(outDrugSub2ProteinSubFileName, *drugSub2proteinSub,
567         outputDelims);
568     return 0;
569 } // end of function

```

5.1.2.10 int gift::EM::outTrainVariance ()

References gift::writeMatrix().

Referenced by main().

```

571         {
572     writeMatrix(outVarDrugSub2proteinSubFileName, *vardrugSub2proteinSub,
573               outputDelims);
574     return 0;
575 } // end of function

```

5.1.2.11 int gift::EM::predictDrugs ()

References gift::drugName2Index, gift::getIndexFromHash(), iterdrugSub2ProteinSub(), gift::predictDrugNameList, and gift::proteinNameList.

Referenced by predictEM().

```

232     {
233     std::cout<<"Now predict given drugs against all the proteins, "
234             <<"which are in our training data set." << std::endl;
235     IntList predictDrugIndex;
236     std::vector<double> tmpCalc;
237     nameList existNameList;
238     getIndexFromHash(drugName2Index,
predictDrugNameList,
239                     predictDrugIndex, existNameList);
240     if (existNameList.size() < 1) {
241         std::cerr<<"No drug Index found, and quit." <<std::endl;
242         return 1;
243     } // end of if
244     // output the result.
245     std::ofstream output (outPredictCPIsFileName,std::ofstream::out);
246     if (!output.is_open()){
247         std::cerr<<"Error open file "<<outPredictCPIsFileName<<std::endl;
248         return 1;
249     } // end of if
250     using boost::algorithm::join;
251     using boost::adaptors::transformed;
252     // print the first row as protein names.
253     output<<"proteins"<<outputDelims;
254     output<<join(proteinNameList,outputDelims)<<std::endl;
255     int num = 0;
256     for(const auto & drug : predictDrugIndex){
257         for(int j=0;j<proteinNum;++j){
258             tmpCalc.push_back(1 - iterdrugSub2ProteinSub(drug,j));
259         } // end of loop for j
260         output<<existNameList[num] << outputDelims;
261         // transformed without static_cast should also work?
262         output<<join(tmpCalc |
263                     transformed(static_cast<std::string(*)>(double)>(std::to_string) ),
264                     outputDelims)
265         << std::endl;
266         tmpCalc.clear();
267         ++num;
268     } // end of loop for drug
269     return 0;
270 } // end of functions.

```

5.1.2.12 int gift::EM::predictDrugsProteins ()

References gift::drugName2Index, gift::getIndexFromHash(), iterdrugSub2ProteinSub(), gift::predictDrugNameList, gift::predictProteinNameList, and gift::proteinName2Index.

Referenced by predictEM().

```

389         {
390             std::cout<<"Now predict given drugs against given proteins." << std::endl;
391             IntList predictDrugIndex;
392             IntList predictProteinIndex;
393             std::vector<double> tmpCalc;
394             nameList existdrugNameList;
395             nameList existproteinNameList;
396             getIndexFromHash(drugName2Index,
predictDrugNameList,
397                             predictDrugIndex, existdrugNameList);
398             getIndexFromHash(proteinName2Index,
predictProteinNameList,
399                             predictProteinIndex, existproteinNameList);
400             if (existdrugNameList.size()<1) {
401                 std::cerr<<"No drug Index found, and quit." <<std::endl;
402                 return 1;
403             } // end of if
404             if (existproteinNameList.size()<1) {
405                 std::cerr<<"No protein Index found, and quit." << std::endl;
406                 return 1;
407             } // end of if
408             // output the result.
409             std::ofstream output (outPredictCPIsFileName,std::ofstream::out);
410             if (!output.is_open()){
411                 std::cerr<<"Error open file "<<outPredictCPIsFileName<<std::endl;
412                 return 1;
413             } // end of if
414             using boost::algorithm::join;
415             using boost::adaptors::transformed;
416             // print the first row as protein names.
417             output<<"Names"<<outputDelims;
418             output<<join(existproteinNameList,outputDelims)<<std::endl;
419             int num = 0;
420             for(const auto & drug : predictDrugIndex){
421                 for(const auto & protein : predictProteinIndex){
422                     tmpCalc.push_back(1 - iterdrugSub2ProteinSub(drug,protein));
423                 } // end of loop for protein
424                 output<<existdrugNameList[num] <<outputDelims;
425                 // transformed without static_cast should also work?
426                 output<<join(tmpCalc |
427                             transformed(static_cast<std::string(*)>(double)>(std::to_string) ),
428                                     outputDelims)
429                     << std::endl;
430                 tmpCalc.clear();
431                 ++num;
432             } // end of loop for drug
433             return 0;
434         } // end of functions.

```

5.1.2.13 int gift::EM::predictDrugsProteinsWithSubs ()

References gift::drugName2Index, gift::getIndexFromHash(), gift::predictDrugNameList, gift::predictProtein2SubList, and gift::predictProteinNameList_WithSubs.

Referenced by predictEM().

```

436         {
437             std::cout<<"Now predict given drugs against given proteins with subs."
438                 << std::endl;
439             IntList predictDrugIndex;
440             std::vector<double> tmpCalc;
441             nameList existdrugNameList;
442             getIndexFromHash(drugName2Index,
predictDrugNameList,
443                             predictDrugIndex, existdrugNameList);
444             if (existdrugNameList.size()<1) {
445                 std::cerr<<"No drug Index found, and quit." <<std::endl;

```



```

446     return 1;
447 } // end of if
448 // output the result.
449 std::ofstream output (outPredictCPIsFileName, std::ofstream::out);
450 if (!output.is_open()){
451     std::cerr<<"Error open file "<<outPredictCPIsFileName<<std::endl;
452     return 1;
453 } // end of if
454 using boost::algorithm::join;
455 using boost::adaptors::transformed;
456 // print the first row as protein names.
457 output<<"Names"<<outputDelims;
458 output<<join(predictProteinNameList_WithSubs, outputDelims)<<std::endl;
459 int num = 0;
460 double tmp;
461 for(const auto & drug : predictDrugIndex){
462     for(int j=0; j<predictProteinNameList_WithSubs.size(); ++j){
463         tmp = 0;
464         for(auto const & m : (*drug2sub)[drug]){
465             for(auto const & n : predictProtein2SubList[j]){
466                 tmp += log(1 - (*drugSub2proteinSub)[m][n]);
467             } // end of loop n
468         } // end of loop for m
469         tmpCalc.push_back(1 - exp(tmp));
470     } // end of loop for protein
471     output<<existDrugNameList[num]<<outputDelims;
472     // transformed without static_cast should also work?
473     output<<join(tmpCalc |
474                 transformed(static_cast<std::string(*) (double)>(std::to_string) ),
475                 outputDelims)
476     << std::endl;
477     tmpCalc.clear();
478     ++num;
479 } // end of loop for drug
480 return 0;
481 } // end of functions.

```

5.1.2.14 int gift::EM::predictDrugsWithSubs ()

References gift::predictDrug2SubList, gift::predictDrugNameList_WithSubs, and gift::proteinNameList.

Referenced by predictEM().

```

272     {
273         std::cout<<"Now predict given drugs with subs against all the proteins, "
274         <<"which are in our training data set." << std::endl;
275         std::vector<double> tmpCalc;
276         nameList existNameList;
277         // output the result.
278         std::ofstream output (outPredictCPIsFileName, std::ofstream::out);
279         if (!output.is_open()){
280             std::cerr<<"Error open file "<<outPredictCPIsFileName<<std::endl;
281             return 1;
282         } // end of if
283         using boost::algorithm::join;
284         using boost::adaptors::transformed;
285         // print the first row as protein names.
286         output<<"proteins"<<outputDelims;
287         output<<join(proteinNameList, outputDelims)<<std::endl;
288         double tmp;
289         for(int i=0; i<predictDrugNameList_WithSubs.size(); ++i){
290             for(int j=0; j<proteinNum; ++j){
291                 tmp = 0;
292                 for(auto const & m : predictDrug2SubList[i]){
293                     for(auto const & n : (*protein2sub)[j]){
294                         tmp += log(1 - (*drugSub2proteinSub)[m][n]);
295                     } // end of loop for n
296                 } // end of loop for m
297                 tmpCalc.push_back(1 - exp(tmp));
298             } // end of loop for j
299             output<<predictDrugNameList_WithSubs[i]<<outputDelims;
300             // transformed without static_cast should also work?
301             output<<join(tmpCalc |
302                         transformed(static_cast<std::string(*) (double)>(std::to_string) ),
303                         outputDelims)
304             << std::endl;
305             tmpCalc.clear();
306         } // end of loop for drug
307         return 0;
308     } // end of functions.

```

5.1.2.15 int gift::EM::predictDrugsWithSubsProteins ()

References `gift::getIndexFromHash()`, `gift::predictDrug2SubList`, `gift::predictDrugNameList_WithSubs`, `gift::predictProteinNameList`, and `gift::proteinName2Index`.

Referenced by `predictEM()`.

```

483     {
484         std::cout<<"Now predict given drugs with subs against given proteins."
485             << std::endl;
486         IntList predictProteinIndex;
487         std::vector<double> tmpCalc;
488         nameList existproteinNameList;
489         getIndexFromHash(proteinName2Index,
predictProteinNameList,
490             predictProteinIndex, existproteinNameList);
491         if (existproteinNameList.size()<1) {
492             std::cerr<<"No protein Index found, and quit." <<std::endl;
493             return 1;
494         } // end of if
495         // output the result.
496         std::ofstream output (outPredictCPIsFileName,std::ofstream::out);
497         if (!output.is_open()){
498             std::cerr<<"Error open file "<<outPredictCPIsFileName<<std::endl;
499             return 1;
500         } // end of if
501         using boost::algorithm::join;
502         using boost::adaptors::transformed;
503         // print the first row as protein names.
504         output<<"Names"<<outputDelims;
505         output<<join(existproteinNameList,outputDelims)<<std::endl;
506         double tmp;
507         for(int i=0;i<predictDrugNameList_WithSubs.size();++i){
508             for(auto const protein : predictProteinIndex){
509                 tmp = 0;
510                 for(auto const & m : predictDrug2SubList[i]){
511                     for(auto const & n : (*protein2sub)[protein]){
512                         tmp += log(1 - (*drugSub2proteinSub)[m][n]);
513                     } // end of loop n
514                 } // end of loop for m
515                 tmpCalc.push_back(1 - exp(tmp));
516             } // end of loop for protein
517             output<<predictDrugNameList_WithSubs[i]<<outputDelims;
518             // transformed without static_cast should also work?
519             output<<join(tmpCalc |
520                 transformed(static_cast<std::string>(*) (double)>(std::to_string) ),
521                 outputDelims)
522                 << std::endl;
523             tmpCalc.clear();
524         } // end of loop for i
525         return 0;
526     } // end of functions.

```

5.1.2.16 int gift::EM::predictDrugsWithSubsProteinsWithSubs ()

References `gift::predictDrug2SubList`, `gift::predictDrugNameList_WithSubs`, `gift::predictProtein2SubList`, and `gift::predictProteinNameList_WithSubs`.

Referenced by `predictEM()`.

```

528     {
529         std::cout<<"Now predict given drugs with subs against given proteins with subs."
530             << std::endl;
531         std::vector<double> tmpCalc;
532         // output the result.
533         std::ofstream output (outPredictCPIsFileName,std::ofstream::out);
534         if (!output.is_open()){
535             std::cerr<<"Error open file "<<outPredictCPIsFileName<<std::endl;
536             return 1;
537         } // end of if
538         using boost::algorithm::join;
539         using boost::adaptors::transformed;
540         // print the first row as protein names.
541         output<<"Names"<<outputDelims;

```

```

542     output<<join(predictProteinNameList_WithSubs,outputDelims)<<std::endl;
543     double tmp;
544     for(int i=0;i<predictDrugNameList_WithSubs.size();++i){
545         for(int j=0;j<predictProteinNameList_WithSubs.size();++j){
546             tmp = 0;
547             for(auto const & m : predictDrug2SubList[i]){
548                 for(auto const & n : predictProtein2SubList[j]){
549                     tmp += log(1 - (*drugSub2proteinSub)[m][n]);
550                 } // end of loop n
551             } // end of loop for m
552             tmpCalc.push_back(1 - exp(tmp));
553         } // end of loop for protein
554         output<<predictDrugNameList_WithSubs[i]<<outputDelims;
555         // transformed without static_cast should also work?
556         output<<join(tmpCalc |
557             transformed(static_cast<std::string(*) (double)>(std::to_string) ),
558             outputDelims)
559             << std::endl;
560         tmpCalc.clear();
561     } // end of loop for i
562     return 0;
563 } // end of functions.

```

5.1.2.17 int gift::EM::predictEM()

References `gift::predictDrugNameList`, `gift::predictDrugNameList_WithSubs`, `predictDrugs()`, `predictDrugsProteins()`, `predictDrugsProteinsWithSubs()`, `predictDrugsWithSubs()`, `predictDrugsWithSubsProteins()`, `predictDrugsWithSubsProteinsWithSubs()`, `gift::predictProteinNameList`, `gift::predictProteinNameList_WithSubs`, `predictProteins()`, and `predictProteinsWithSubs()`.

Referenced by `main()`.

```

193     {
194         // Note: we only run one situation one time:
195         // - Provide both drugs and proteins Names.
196         // - Provide both drugs and proteins with Subs.
197         // - Provide both drugs Names and proteins with Subs.
198         // - Provide both drugs with Subs and proteins Names.
199         // - Provide only drugs Names.
200         // - Provide only drug with Subs.
201         // - Provide only protein Names.
202         // - Provide only protein with Subs.
203         std::cout<<"Now Run predictEM for task: predict..." << std::endl;
204
205         if (!predictDrugNameList_WithSubs.empty() &
206             !predictProteinNameList_WithSubs.empty()){
207             predictDrugsWithSubsProteinsWithSubs();
208         } else if (!predictDrugNameList_WithSubs.empty() &
209             !predictProteinNameList.empty()){
210             predictDrugsWithSubsProteins();
211         } else if (!predictProteinNameList_WithSubs.empty() &
212             !predictDrugNameList.empty()){
213             predictDrugsProteinsWithSubs();
214         } else if (!predictDrugNameList.empty() &
215             !predictProteinNameList.empty()){
216             predictDrugsProteins();
217         } else if (!predictDrugNameList.empty()){
218             predictDrugs();
219         } else if (!predictDrugNameList_WithSubs.empty()){
220             predictDrugsWithSubs();
221         } else if (!predictProteinNameList.empty()){
222             predictProteins();
223         } else if (!predictProteinNameList_WithSubs.empty()){
224             predictProteinsWithSubs();
225         } else {
226             std::cerr<<"No files for task: predict, and quit."<< std::endl;
227             return 1;
228         } // end of if else if else.
229         return 0;
230     } // end of function

```

5.1.2.18 int gift::EM::predictProteins ()

References gift::drugNameList, gift::getIndexFromHash(), iterdrugSub2ProteinSub(), gift::predictProteinNameList, and gift::proteinName2Index.

Referenced by predictEM().

```

310         {
311             std::cout<<"Now predict given proteins against all the drugs, "
312                 <<"which are in our training data set." << std::endl;
313             IntList predictProteinsIndex;
314             std::vector<double> tmpCalc;
315             nameList existNameList;
316             getIndexFromHash(proteinName2Index,
predictProteinNameList,
317                 predictProteinsIndex, existNameList);
318             if (existNameList.size() < 1) {
319                 std::cerr<<"No protein Index found, and quit." <<std::endl;
320                 return 1;
321             } // end of if
322             // output the result.
323             std::ofstream output (outPredictCPIsFileName,std::ofstream::out);
324             if (!output.is_open()){
325                 std::cerr<<"Error open file "<<outPredictCPIsFileName<<std::endl;
326                 return 1;
327             } // end of if
328             using boost::algorithm::join;
329             using boost::adaptors::transformed;
330             // print the first row as drug names.
331             output<<"drugs"<<outputDelims;
332             output<<join(drugNameList,outputDelims)<<std::endl;
333             int num = 0;
334             for(const auto & protein : predictProteinsIndex){
335                 for(int j=0;j<drugNum;++j){
336                     tmpCalc.push_back(1 - iterdrugSub2ProteinSub(j,protein));
337                 } // end of loop for j
338                 output<<existNameList[num]<<outputDelims;
339                 // transformed without static_cast should also work?
340                 output<<join(tmpCalc |
341                     transformed(static_cast<std::string(*)>(double)>(std::to_string) ),
342                     outputDelims)
343                     << std::endl;
344                 tmpCalc.clear();
345                 ++num;
346             } // end of loop for protein
347             return 0;
348         } // end of functions.

```

5.1.2.19 int gift::EM::predictProteinsWithSubs ()

References gift::drugNameList, gift::predictProtein2SubList, and gift::predictProteinNameList_WithSubs.

Referenced by predictEM().

```

350         {
351             std::cout<<"Now predict given proteins with subs against all the drugs, "
352                 <<"which are in our training data set." << std::endl;
353             IntList predictProteinsIndex;
354             std::vector<double> tmpCalc;
355             nameList existNameList;
356             // output the result.
357             std::ofstream output (outPredictCPIsFileName,std::ofstream::out);
358             if (!output.is_open()){
359                 std::cerr<<"Error open file "<<outPredictCPIsFileName<<std::endl;
360                 return 1;
361             } // end of if
362             using boost::algorithm::join;
363             using boost::adaptors::transformed;
364             // print the first row as drug names.
365             output<<"drugs"<<outputDelims;
366             output<<join(drugNameList,outputDelims)<<std::endl;
367             double tmp;
368             for(int i=0;i<predictProteinNameList_WithSubs.size();++i){
369                 for(int j=0;j<drugNum;++j){

```

```

370         tmp = 0;
371         for(auto const & m : (*drug2sub)[j]){
372             for(auto const & n : predictProtein2SubList[i]){
373                 tmp += log(1 - (*drugSub2proteinSub)[m][n]);
374             } // end of loop for n
375         } // end of loop for m
376         tmpCalc.push_back(1 - exp(tmp));
377     } // end of loop for j
378     output<<predictProteinNameList_WithSubs[i]<<outputDelims;
379     // transformed without static_cast should also work?
380     output<<join(tmpCalc |
381         transformed(static_cast<std::string(*) (double)>(std::to_string) ),
382         outputDelims)
383         << std::endl;
384     tmpCalc.clear();
385 } // end of loop for protein
386 return 0;
387 } // end of functions.

```

5.1.2.20 double gift::EM::recLoglikely ()

References iterdrugSub2ProteinSub().

Referenced by trainEM().

```

99     {
100     double loglikely = 0;
101     double tmp;
102     std::vector<int>::iterator it;
103     for(int i=0;i<drugNum;++i) {
104         for(int j=0;j<proteinNum;++j) {
105             tmp = iterdrugSub2ProteinSub(i,j);
106             it = std::find((*drug2protein)[i].begin(),
107                 (*drug2protein)[i].end(),j);
108             loglikely += it==(*drug2protein)[i].end() ?
109                 log(1-(1-fn)*(1-tmp)-fp*tmp) : log((1-fn)*(1-tmp) + fp*tmp);
110         } // end of loop j
111     } // end of loop i
112     return loglikely;
113 } // end of function.

```

5.1.2.21 int gift::EM::setLoglikely (double logscore) [inline]

Referenced by trainEM().

```

277     {
278         (*loglikely).push_back(logscore);
279         return 0;
280     } // end of function

```

5.1.2.22 int gift::EM::setPointerDrug2Protein (IntArrayList & d2p) [inline]

```

242     {
243         drug2protein = &d2p;
244         return 0;
245     } // end of func

```

5.1.2.23 int gift::EM::setPointerDrug2Sub (IntArrayList & d2s) [inline]

```

234     {
235         drug2sub = &d2s;
236         return 0;
237     } // end of func

```

5.1.2.24 int gift::EM::setPointerDrugSub2ProteinSub (numericMatrix & ds2ps) [inline]

```

246                                     {
247     drugSub2proteinSub = &ds2ps;
248     return 0;
249 } // end of func

```

5.1.2.25 int gift::EM::setPointerProtein2Sub (IntArrayList & p2s) [inline]

```

238                                     {
239     protein2sub = &p2s;
240     return 0;
241 } // end of func

```

5.1.2.26 int gift::EM::trainEM ()

References EStep(), MStep(), gift::recLogLeastNum, recLoglikely(), and setLoglikely().

Referenced by main().

```

115     {
116     // lack of loglikely record
117     for(int i=0;i<iterNum;++i){
118         std::cout<<"Current iteration number is " << i << std::endl;
119         std::chrono::steady_clock::time_point tBegin =
120             std::chrono::steady_clock::now();
121         EStep();
122         //std::cout<<"EStep Testing..."<<std::endl;
123         //EStep(1); // for testEM
124         std::chrono::steady_clock::time_point tEnd =
125             std::chrono::steady_clock::now();
126         std::cout<<"EStep Time difference (s): "
127             <<std::chrono::duration_cast<std::chrono::seconds>(tBegin-tEnd).count()
128             <<std::endl;
129
130         tBegin = std::chrono::steady_clock::now();
131         MStep();
132         //MStep(1); // for testEM
133         tEnd = std::chrono::steady_clock::now();
134         std::cout<<"MStep Time difference (s): "
135             <<std::chrono::duration_cast<std::chrono::seconds>(tBegin-tEnd).count()
136             <<std::endl;
137
138         if (loglikelyRecord || i<=recLogLeastNum || i >= iterNum-recLogLeastNum) {
139             double tmplog = recLoglikely();
140             std::cout<<"Current loglikelyhood is " << tmplog << std::endl;
141             // // for test
142             // std::cout<<"Current observedDrug2Protein Matrix is: "<<std::endl;
143             // printMatrix(*observedDrug2Protein);
144             // std::cout<<"Current drugSub2proteinSub Matrix is: "<<std::endl;
145             // printMatrix(*drugSub2proteinSub);
146
147             setLoglikely(tmplog);
148         } // end of if
149     } // end of loop i
150     std::chrono::system_clock::time_point endTime =
151         std::chrono::system_clock::now();
152     std::time_t endTimeT = std::chrono::system_clock::to_time_t(endTime);
153     std::cout<<"Finished computation at " << std::ctime(&endTimeT) << std::endl;
154     return 0;
155 } // end of function

```

5.1.2.27 int gift::EM::varEM ()

References iterdrugSub2ProteinSub().

Referenced by main().

```

157     {
158         numericMatrix quesidev;
159         std::vector<double> tmpDev;
160         // Note: not check drugSub2proteinsub values larger than 0.95.
161         for (int i=0; i<drugNum; ++i) {
162             for (int j=0; j<proteinNum; ++j) {
163                 tmpDev.push_back(iterdrugSub2ProteinSub(i, j));
164             } // end of loop j
165             quesidev.push_back(tmpDev);
166             tmpDev.clear();
167         } // end of loop i
168
169         for(int i=0; i<subNum; ++i) {
170             for(int j=0; j<domainNum; ++j) {
171                 double tmp_t = 0;
172                 //std::vector<double> tmp_t_array;
173                 double tmp_s = 0;
174                 //std::vector<double> tmp_s_array;
175                 double tmpLikely = (*observedDrug2Protein)[i][j];
176                 double tmp_sum = 0;
177                 for(auto const & m : (*sub2drug)[i]) {
178                     for(auto const & n : (*sub2protein)[j]) {
179                         tmp_t = std::find((*drug2protein)[m].begin(), (*drug2protein)[m].end(), n)
180                             == (*drug2protein)[m].end() ? 1/pow(1-tmpLikely, 2.0) : 1/pow(tmpLikely, 2.0);
181                         // tmp_t_array.push_back(tmp_t);
182                         tmp_s = (1-fn-fp) * quesidev[m][n] / (1 - (*drugSub2proteinsub)[i][j]);
183                         //tmp_s_array.push_back(tmp_s);
184                         tmp_sum += pow(tmp_s, 2.0) * tmp_t;
185                     } // end of loop n
186                 } // end of loop m
187                 (*vardrugSub2proteinSub)[i][j] = tmp_sum;
188             } // end of loop j
189         } // end of loop i
190         return 0;
191     } // end of function

```

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

- [gift.hpp](#)
- [EMGift.cpp](#)

5.2 gift::parameters Class Reference

```
#include <gift.hpp>
```

Public Member Functions

- [parameters](#) (const std::string) throw (std::string)
- int [setDrugNum](#) (int number)
- int [setSubNum](#) (int number)
- int [setProteinNum](#) (int number)
- int [setDomainNum](#) (int number)
- int [InitDrugSub2ProteinSub](#) ()
- int [InitVarDrugSub2ProteinSub](#) ()
- int [InitDrugName2Index](#) ()
- int [InitProteinName2Index](#) ()
- int [InitDrugSubNameList](#) ()
- int [InitProteinSubNameList](#) ()
- int [InitPredictParameters](#) () throw (std::string)
- int [InitObservedDrug2ProteinMatrix](#) ()

Public Attributes

- `std::string drug2proteinFileName`
- `std::string drug2subFileName`
- `std::string protein2subFileName`
- `std::string drugSub2proteinSubFileName`
- `std::string drugNameListFile`
- `std::string drugSubNameListFile`
- `std::string proteinNameListFile`
- `std::string proteinSubNameListFile`
- `bool loglikelyRecord`
- `double alphaEB`
- `double betaEB`
- `double fn`
- `double fp`
- `int thread`
- `int iterNum`
- `int drugNum`
- `int subNum`
- `int domainNum`
- `int proteinNum`
- `std::string inputDelims`
- `std::string task`
- `std::string chemfpRec`
- `std::string proteinfpRec`
- `std::string CPlsRec`
- `std::string predictDrugsFileName`
- `std::string predictProteinsFileName`
- `std::string predictDrugsFileName_WithSubs`
- `std::string predictProteinsFileName_WithSubs`
- `std::string outputDelims`
- `std::string outRecordFileName`
- `std::string outPredictCPlsFileName`
- `std::string outDrugSub2ProteinSubFileName`
- `std::string outVarDrugSub2proteinSubFileName`

5.2.1 Constructor & Destructor Documentation

5.2.1.1 `gift::parameters::parameters (const std::string configFile) throw std::string`

References `gift::BoolToString()`, `gift::rowCol::colNum`, `gift::Matrix2Fingerprints()`, `gift::Matrix2FingerprintsByColumn()`, `gift::printIntArrayList()`, `gift::rowColFile()`, and `gift::rowCol::rowNum`.

```

39
40     std::cout<<"Now set parameters with configFile."<<std::endl;
41     // use boost program_options to read configs from a given file.
42     namespace po = boost::program_options;
43     po::options_description desc("GIFT Parameter options");
44     desc.add_options()
45         // input data file name
46         ("drug2proteinFileName",
47          po::value<std::string>(&drug2proteinFileName),
48          "file name for drug protein interactions")
49         ("drug2subFileName",po::value<std::string>(&drug2subFileName),
50          "file name for drug to substructure")
51         ("protein2subFileName",po::value<std::string>(&protein2subFileName),
52          "file name for protein to substructure")
53         ("drugSub2proteinSubFileName",

```



```

54     po::value<std::string>(&drugSub2proteinSubFileName),
55     "file name for drugSub to proteinSub interaction probability")
56 // input name list file name
57 ("drugNameListFile", po::value<std::string>(&drugNameListFile),
58  "file name for drug names")
59 ("drugSubNameListFile", po::value<std::string>(&drugSubNameListFile),
60  "file name for drug substructures names")
61 ("proteinNameListFile", po::value<std::string>(&proteinNameListFile),
62  "file name for protein names")
63 ("proteinSubNameListFile", po::value<std::string>(&proteinSubNameListFile),
64  "file name for protein substructures names")
65 // input parameters for EM
66 ("alphaEB", po::value<double>(&alphaEB)->default_value(0.05),
67  "parameter for Empirical Bayesian estimates for initEM")
68 ("betaEB", po::value<double>(&betaEB)->default_value(0.05),
69  "parameter for Empirical Bayesian estimates for initEM")
70 ("fp", po::value<double>(&fp)->default_value(0.85),
71  "false positive rate")
72 ("fn", po::value<double>(&fn)->default_value(0.0001),
73  "false negative rate")
74 ("threadNum", po::value<int>(&thread)->default_value(1),
75  "thread number for EM")
76 ("EMIterationNum", po::value<int>(&iterNum)->default_value(300),
77  "iteration numbers for EM")
78 ("task", po::value<std::string>(&task)->default_value("train"),
79  "run gift for train or predict")
80 ("loglikelyRecord",
81  po::value<bool>(&loglikelyRecord)->default_value(false),
82  "whether or not to record the loglikely in every step")
83 ("inputDelims", po::value<std::string>(&inputDelims)->default_value("\t,"),
84  "sep character for input files")
85 // input file version information.
86 ("chemFingerPrintRecord",
87  po::value<std::string>(&chemfpRec)->default_value("ComFP: PUBCHEM"),
88  "source and version of chemical fingerprints")
89 ("proteinFingerPrintRecord",
90  po::value<std::string>(&proteinfpRec)->default_value("Pfam: 2011-07"),
91  "source and version of protein fingerprints/domains")
92 ("comProteinInteractionRecord",
93  po::value<std::string>(&CPIsRec)->default_value("DrugBank: 2011-07"),
94  "source and version of compound-protein interactions")
95 // input file names for prediction
96 ("predictDrugsFileName", po::value<std::string>(&predictDrugsFileName),
97  "file name for drug names used for prediction by gift")
98 ("predictProteinFileName", po::value<std::string>(&predictProteinsFileName),
99  "file name for protein names used for prediction by gift")
100 ("predictDrugsFileName_WithSubs",
101  po::value<std::string>(&predictDrugsFileName_WithSubs),
102  "file name for drugs names together with their substructures.")
103 ("predictProteinsFileName_WithSubs",
104  po::value<std::string>(&predictProteinsFileName_WithSubs),
105  "file name for protein names together with their substructures.")
106 // output file name and format
107 ("outputDelims", po::value<std::string>(&outputDelims)->default_value("\t,"),
108  "sep character for output files")
109 ("outRecordFileName",
110  po::value<std::string>(&outRecordFileName)->default_value("CPIs"),
111  "file name for output records")
112 ("outPredictCPIsFileName",
113  po::value<std::string>(&outPredictCPIsFileName),
114  "file name for output CPIs")
115 ("outDrugSub2ProteinSubFileName",
116  po::value<std::string>(&outDrugSub2ProteinSubFileName),
117  "file name for output drugSub2proteinSub")
118 ("outVarDrugSub2proteinSubFileName",
119  po::value<std::string>(&outVarDrugSub2proteinSubFileName),
120  "file name for output variance of drugSub2proteinSub");
121 po::variables_map vm;
122 std::ifstream input (configFile, std::ios::in);
123 po::store(po::parse_config_file(input, desc), vm);
124 po::notify(vm);
125 // input.exceptions(std::ifstream::failbit | std::ifstream::badbit);
126 // try {
127 //     input.open(configFile, std::ifstream::in);
128 //     if (input.peek() == std::ifstream::traits_type::eof()) {
129 //         std::cerr<<configFile<<" is empty." << std::endl;
130 //     } // end of if.
131 //     po::store(po::parse_config_file(input, desc), vm);
132 //     po::notify(vm);
133 // } catch (std::ifstream::failure e) {
134 //     std::cerr <<"Exceptions open/read file "<<configFile<<std::endl;
135 // } // end of try catch
136 // default training data parameters.
137 // they will be set when read data files.
138
139 rowCol tmp;
140 rowColFile(drugSubFileName, tmp, inputDelims);

```

```

141     drugNum = tmp.rowNum;
142     subNum = tmp.colNum;
143     std::cout<<"Drug Number is "<<drugNum<<std::endl;
144     std::cout<<"DrugSub Number is "<<subNum<<std::endl;
145     rowColFile(protein2subFileName,tmp,inputDelims);
146     domainNum = tmp.colNum;
147     proteinNum = tmp.rowNum;
148     std::cout<<"Protein Number is "<<proteinNum<<std::endl;
149     std::cout<<"ProteinSub Number is "<<domainNum<<std::endl;
150
151
152     // load global data for gift.
153     Matrix2Fingerprints(drug2proteinFileName,
drug2proteinList,inputDelims);
154     //printIntArrayList(drug2proteinList); // for test
155     Matrix2Fingerprints(protein2subFileName,
protein2domainList,inputDelims);
156     //printIntArrayList(protein2domainList); // for test
157     Matrix2Fingerprints(drug2subFileName,
drug2subList,inputDelims);
158     //printIntArrayList(drug2subList); // for test
159
160     IntList tmpIntArray;
161     std::cout<<"Initialize the sub2drugList..."<<std::endl;
162     Matrix2FingerprintsByColumn(drug2subFileName,
sub2drugList,subNum,inputDelims);
163     std::cout<<"Finish the init of sub2drugList..."<<std::endl;
164     printIntArrayList(sub2drugList); // for test
165
166     std::cout<<"Initialize the domain2proteinList..."<<std::endl;
167     Matrix2FingerprintsByColumn(protein2subFileName,
domain2proteinList,
168                               domainNum, inputDelims);
169     std::cout<<"Finish the init of domain2proteinList..."<<std::endl;
170     printIntArrayList(domain2proteinList); // for test
171
172     InitDrugSub2ProteinSub();
173     InitVarDrugSub2ProteinSub();
174
175     InitObservedDrug2ProteinMatrix();
176     // load NameList.
177     InitDrugName2Index();
178     InitProteinName2Index();
179     InitDrugSubNameList();
180     InitProteinSubNameList();
181
182     // load predicted name list and possible subs if task is prediction.
183     InitPredictParameters(); // throw std::string.
184
185     // print the setting results.
186     std::cout<<"parameters have been set."<<std::endl;
187     for (const auto& it : vm){
188         std::cout<< it.first.c_str() << ": ";
189         auto& value = it.second.value(); // return boost::any reference type.
190         // any_cast use the any * as input and return the pointer with type infor.
191         if (auto v = boost::any_cast<int>(&value) ) {
192             std::cout<< *v <<std::endl;
193         } else if (auto v = boost::any_cast<double>(&value) ) {
194             std::cout<< *v <<std::endl;
195         } else if (auto v = boost::any_cast<bool>(&value) ) {
196             std::cout<< BoolToString(*v) <<std::endl;
197         } else if (auto v = boost::any_cast<std::string>(&value) ) {
198             std::cout<< *v <<std::endl;
199         } else {
200             std::cout<< "Error type"<<std::endl;
201         } // end of if
202     } // end of for
203     std::cout<<"drugNum: " <<drugNum<<std::endl;
204     std::cout<<"subNum: " <<subNum<<std::endl;
205     std::cout<<"domainNum: " <<domainNum<<std::endl;
206     std::cout<<"proteinNum: " <<proteinNum<<std::endl;
207 } // end of class parameter constructor.

```

5.2.2 Member Function Documentation

5.2.2.1 int gift::parameters::InitDrugName2Index ()

References drugNameListFile, gift::readName2IndexHash(), and gift::readNameListFromFile().

```

261         {
262             std::cout<<"Initialize DrugName2Index Hash..."<<std::endl;
263             readNameListFromFile(drugNameListFile,
drugNameList);
264             readName2IndexHash(drugNameList,drugName2Index);
265             std::cout<<"Finish DrugName2Index Hash."<<std::endl;
266             return 0;
267         } // end of function

```

5.2.2.2 int gift::parameters::InitDrugSub2ProteinSub ()

References `alphaEB`, `betaEB`, `domainNum`, `drugSub2proteinSubFileName`, `inputDelims`, `gift::readMatrix()`, `subNum`, and `task`.

```

209         {
210             // This function must be run after class parameter initialization.
211             std::cout<< "Initialize the drugSub2proteinSub Matrix." << std::endl;
212             if (task.compare("predict") == 0 ) {
213                 readMatrix(drugSub2proteinSubFileName,
drugSub2proteinSubMatrix,
214                             inputDelims);
215                 std::cout<< "Finish: read from file."<<std::endl;
216             } else {
217                 std::vector<double> assoTmp; // temp array based on the assocaiton method.
218                 int N;
219                 int subNumTmp;
220                 int I = 0;
221                 std::vector<int>::iterator it;
222                 for (int i=0;i<subNum;++i){
223                     subNumTmp = sub2drugList[i].size();
224                     for (int j=0;j<domainNum;++j){
225                         N = domain2proteinList[j].size() * subNumTmp;
226                         for (const auto drug : sub2drugList[i]){
227                             for (const auto protein : domain2proteinList[j]){
228                                 it = std::find(drug2proteinList[drug].begin(),
drug2proteinList[drug].end(), protein);
229                                 I += (it==drug2proteinList[drug].end() ? 0 : 1);
230                             } // end of loop protein
231                         } // end of loop drug
232                     } // revise association method with Emiprical Bayes.
233                     assoTmp.push_back((I+alphaEB)/(alphaEB+betaEB+N));
234                     I = 0;
235                 } // end of loop j
236                 drugSub2proteinSubMatrix.push_back(assoTmp);
237                 assoTmp.clear();
238             } // end of loop i
239             std::cout<<"Finish: initialize with association method and emprical Bayes."
240                 <<std::endl;
241         } // end of if else
242         return 0;
243     } // end of function

```

5.2.2.3 int gift::parameters::InitDrugSubNameList ()

References `drugSubNameListFile`, and `gift::readNameListFromFile()`.

```

275         {
276             readNameListFromFile(drugSubNameListFile,
drugSubNameList);
277             return 0;
278         } // end of function

```

5.2.2.4 `int gift::parameters::InitObservedDrug2ProteinMatrix ()`

References `drugNum`, `proteinNum`, and `task`.

```

317                                     {
318     std::cout<<"Init ObservedDrug2ProteinMatrix ..."<<std::endl;
319     if(task.compare("predict") == 0){
320         std::cout<<"Job is to predict, and ignore the initation of observedDrug2Protein."
321             <<std::endl;
322     } else{
323         std::vector<double> tmpArray(proteinNum,0.1);
324         for(int i=0;i<drugNum;++i){
325             observedDrug2ProteinMatrix.push_back(tmpArray);
326         } // end of loop for i.
327     } // end of if else
328     std::cout<<"Finish the init of observedDrug2ProteinMatrix."<<std::endl;
329     return 0;
330 } // end of function.

```

5.2.2.5 `int gift::parameters::InitPredictParameters () throw std::string`

References `predictDrugsFileName`, `predictDrugsFileName_WithSubs`, `predictProteinsFileName`, `predictProteinsFileName_WithSubs`, `gift::readNameListFromFile()`, and `gift::readNameMatrixFromFile()`.

```

285                                     {
286     // when task is predict, we use this function to init the corresponding
287     // parameters.
288     bool checkStatus = false;
289     if (!predictDrugsFileName.empty()){
290         checkStatus = true;
291         readNameListFromFile(predictDrugsFileName,
292             predictDrugNameList);
293     } // end of if for drugfile
294     if (!predictProteinsFileName.empty()){
295         checkStatus = true;
296         readNameListFromFile(predictProteinsFileName,
297             predictProteinNameList);
298     } // end of if for proteinfile
299     if (!predictDrugsFileName_WithSubs.empty()){
300         checkStatus = true;
301         // NEED FUNCTION.
302         readNameMatrixFromFile(predictDrugsFileName_WithSubs
303             ,
304             predictDrugNameList_WithSubs,
305             predictDrug2SubList);
306     } // end of if for drug_withsubs file.
307     if (!predictProteinsFileName_WithSubs.empty()){
308         checkStatus = true;
309         // NEED FUNCTION.
310         readNameMatrixFromFile(
311             predictProteinsFileName_WithSubs,
312             predictProteinNameList_WithSubs,
313             predictProtein2SubList);
314     } // end of if for protein_withsubs file.
315     if (!checkStatus){
316         throw("Task for prediction, but no predicted files!");
317     } // end of if for checkStatus.
318     return 0;
319 } // end of function

```

5.2.2.6 `int gift::parameters::InitProteinName2Index ()`

References `proteinNameListFile`, `gift::readName2IndexHash()`, and `gift::readNameListFromFile()`.

```

269                                     {
270     readNameListFromFile(proteinNameListFile,
271         proteinNameList);
272     readName2IndexHash(proteinNameList,
273         proteinName2Index);
274     return 0;
275 } // end of function

```

5.2.2.7 int gift::parameters::InitProteinSubNameList ()

References proteinSubNameListFile, and gift::readNameListFromFile().

```

280                                     {
281     readNameListFromFile(proteinSubNameListFile,
        proteinSubNameList);
282     return 0;
283 } // end of function

```

5.2.2.8 int gift::parameters::InitVarDrugSub2ProteinSub ()

References domainNum, subNum, and task.

```

246                                     {
247     if (task.compare("predict") == 0 ) {
248         std::cout<< "Job is to predict, skip init variance matrix." << std::endl;
249     } else {
250         std::cout<<"Initialize the variance matrix for drugSub2ProteinSub..."
251             <<std::endl;
252         std::vector<double> tmpArray (domainNum,1.01);
253         for(int i=0;i<subNum;++i){
254             vardrugSub2proteinSubMatrix.push_back(tmpArray);
255         } // end of loop for i.
256     } // end of if else
257     std::cout<<"Finish init of variance matrix for drugSub2ProteinSub." <<std::endl;
258     return 0;
259 } // end of function

```

5.2.2.9 int gift::parameters::setDomainNum (int *number*) [inline]

```

140 { domainNum = number; return 0; }

```

5.2.2.10 int gift::parameters::setDrugNum (int *number*) [inline]

```

137 { drugNum = number; return 0; }

```

5.2.2.11 int gift::parameters::setProteinNum (int *number*) [inline]

```

139 {proteinNum = number; return 0; }

```

5.2.2.12 int gift::parameters::setSubNum (int *number*) [inline]

```

138 { subNum = number; return 0; }

```

5.2.3 Member Data Documentation

5.2.3.1 double gift::parameters::alphaEB

Referenced by InitDrugSub2ProteinSub().

5.2.3.2 double gift::parameters::betaEB

Referenced by InitDrugSub2ProteinSub().

5.2.3.3 std::string gift::parameters::chemfpRec

5.2.3.4 std::string gift::parameters::CPIsRec

5.2.3.5 int gift::parameters::domainNum

Referenced by InitDrugSub2ProteinSub(), and InitVarDrugSub2ProteinSub().

5.2.3.6 std::string gift::parameters::drug2proteinFileName

5.2.3.7 std::string gift::parameters::drug2subFileName

5.2.3.8 std::string gift::parameters::drugNameListFile

Referenced by InitDrugName2Index().

5.2.3.9 int gift::parameters::drugNum

Referenced by InitObservedDrug2ProteinMatrix().

5.2.3.10 std::string gift::parameters::drugSub2proteinSubFileName

Referenced by InitDrugSub2ProteinSub().

5.2.3.11 std::string gift::parameters::drugSubNameListFile

Referenced by InitDrugSubNameList().

5.2.3.12 double gift::parameters::fn

5.2.3.13 double gift::parameters::fp

5.2.3.14 std::string gift::parameters::inputDelims

Referenced by InitDrugSub2ProteinSub().

5.2.3.15 `int gift::parameters::iterNum`

5.2.3.16 `bool gift::parameters::loglikelyRecord`

Referenced by `gift::outRecord()`.

5.2.3.17 `std::string gift::parameters::outDrugSub2ProteinSubFileName`

5.2.3.18 `std::string gift::parameters::outPredictCPIsFileName`

5.2.3.19 `std::string gift::parameters::outputDelims`

5.2.3.20 `std::string gift::parameters::outRecordFileName`

Referenced by `gift::outRecord()`.

5.2.3.21 `std::string gift::parameters::outVarDrugSub2proteinSubFileName`

5.2.3.22 `std::string gift::parameters::predictDrugsFileName`

Referenced by `InitPredictParameters()`.

5.2.3.23 `std::string gift::parameters::predictDrugsFileName_WithSubs`

Referenced by `InitPredictParameters()`.

5.2.3.24 `std::string gift::parameters::predictProteinsFileName`

Referenced by `InitPredictParameters()`.

5.2.3.25 `std::string gift::parameters::predictProteinsFileName_WithSubs`

Referenced by `InitPredictParameters()`.

5.2.3.26 `std::string gift::parameters::protein2subFileName`

5.2.3.27 `std::string gift::parameters::proteinfpRec`

5.2.3.28 `std::string gift::parameters::proteinNameListFile`

Referenced by `InitProteinName2Index()`.

5.2.3.29 `int gift::parameters::proteinNum`

Referenced by `InitObservedDrug2ProteinMatrix()`.

5.2.3.30 `std::string gift::parameters::proteinSubNameListFile`

Referenced by `InitProteinSubNameList()`.

5.2.3.31 `int gift::parameters::subNum`

Referenced by `InitDrugSub2ProteinSub()`, and `InitVarDrugSub2ProteinSub()`.

5.2.3.32 `std::string gift::parameters::task`

Referenced by `InitDrugSub2ProteinSub()`, `InitObservedDrug2ProteinMatrix()`, `InitVarDrugSub2ProteinSub()`, `main()`, and `gift::outRecord()`.

5.2.3.33 `int gift::parameters::thread`

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

- [gift.hpp](#)
- [parameterGift.cpp](#)

5.3 `gift::rowCol` Class Reference

```
#include <gift.hpp>
```

Public Member Functions

- [rowCol](#) (int row, int col)
- [rowCol](#) ()

Public Attributes

- int [rowNum](#)
- int [colNum](#)

5.3.1 Constructor & Destructor Documentation

5.3.1.1 `gift::rowCol::rowCol (int row, int col)` `[inline]`

```
118 : rowNum(row), colNum(col) {}
```


5.3.1.2 gift::rowCol::rowCol () [inline]

```
119 : rowNum(1), colNum(1) {}
```

5.3.2 Member Data Documentation

5.3.2.1 int gift::rowCol::colNum

Referenced by gift::parameters::parameters(), and gift::rowColFile().

5.3.2.2 int gift::rowCol::rowNum

Referenced by gift::parameters::parameters(), and gift::rowColFile().

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

- [gift.hpp](#)

Chapter 6

File Documentation

6.1 EMGift.cpp File Reference

```
#include <cmath>
#include <ctime>
#include <chrono>
#include <boost/algorithm/string.hpp>
#include <boost/algorithm/string/join.hpp>
#include <boost/range/adaptor/transformed.hpp>
#include "gift.hpp"
```

Namespaces

- [gift](#)

6.2 functionsGift.cpp File Reference

```
#include <ctime>
#include <chrono>
#include <boost/algorithm/string.hpp>
#include <boost/algorithm/string/join.hpp>
#include <boost/range/adaptor/transformed.hpp>
#include "gift.hpp"
```

Namespaces

- [gift](#)

Functions

- int [gift::Matrix2Fingerprints](#) (const std::string inputFile, IntArrayList &getFp, std::string delims)
- int [gift::readMatrix](#) (const std::string inputFile, numericMatrix &getMat, std::string delims)
- int [gift::rowColFile](#) (const std::string inputFile, rowCol &matrixRec, std::string delims)
- int [gift::writeMatrix](#) (const std::string outFileName, numericMatrix &resultMat, std::string delims)
- int [gift::printIntArrayList](#) (const IntArrayList &fromIntArrayList)
- int [gift::printMatrix](#) (const numericMatrix &fromMatrix)
- int [gift::readNameListFromFile](#) (const std::string inputFile, nameList &tonameList)
- int [gift::readNameMatrixFromFile](#) (const std::string inputFile, nameList &tonameList, IntArrayList &getFP, std::string delims)
- int [gift::readName2IndexHash](#) (const nameList fromNameList, name2IndexHash &name2Index)
- int [gift::getIndexFromHash](#) (const name2IndexHash &name2Index, const nameList fromNameList, IntList &toIndexList, nameList &existNameList)
- int [gift::helpGift](#) ()
- int [gift::outRecord](#) (parameters &EMparameters)
- int [gift::Matrix2FingerprintsByColumn](#) (const std::string inputFile, IntArrayList &getFP, int rowNum, std::string delims)

6.3 gift.hpp File Reference

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <map>
#include <boost/thread/thread.hpp>
#include <boost/bind.hpp>
```

Classes

- class [gift::rowCol](#)
- class [gift::parameters](#)
- class [gift::EM](#)

Namespaces

- [gift](#)

Typedefs

- typedef std::vector< int > [gift::IntList](#)
- typedef std::vector< std::vector< int > > [gift::IntArrayList](#)
- typedef std::vector< std::vector< double > > [gift::numericMatrix](#)
- typedef std::map< std::string, int > [gift::name2IndexHash](#)
- typedef std::vector< std::string > [gift::nameList](#)

Functions

- const std::string [gift::author](#) ("Songpeng Zu")
- const std::string [gift::email](#) ("zusongpeng@gmail.com")
- const std::string [gift::version](#) ("gift-2.0")
- const std::string [gift::updateTime](#) ("2016-03-06")
- int [gift::Matrix2Fingerprints](#) (const std::string inputFile, IntArrayList &getFp, std::string delims)
- int [gift::Matrix2FingerprintsByColumn](#) (const std::string inputFile, IntArrayList &getFP, int rowNum, std::string delims)
- int [gift::writeMatrix](#) (const std::string outFileName, numericMatrix &resultMat, std::string delims)
- int [gift::printIntArrayList](#) (const IntArrayList &fromIntArrayList)
- int [gift::printMatrix](#) (const numericMatrix &fromMatrix)
- int [gift::readMatrix](#) (const std::string inputFile, numericMatrix &getMat, std::string delims)
- int [gift::readNameListFromFile](#) (const std::string inputFile, nameList &tonameList)
- int [gift::readNameMatrixFromFile](#) (const std::string inputFile, nameList &tonameList, IntArrayList &getFP, std::string delims)
- int [gift::readName2IndexHash](#) (const nameList fromNameList, name2IndexHash &name2Index)
- int [gift::getIndexFromHash](#) (const name2IndexHash &name2Index, const nameList fromNameList, IntList &toIndexList, nameList &existNameList)
- int [gift::rowColFile](#) (const std::string inputFile, rowCol &matrixRec, std::string delims)
- int [gift::helpGift](#) ()
- int [gift::outRecord](#) (parameters &EMparameters)
- template<typename func >
int [gift::functionThread](#) (func useFun, int thread, EM *point)

Variables

- const int [gift::recLogLeastNum](#) = 5
- IntArrayList [gift::drug2proteinList](#)
- IntArrayList [gift::drug2subList](#)
- IntArrayList [gift::sub2drugList](#)
- IntArrayList [gift::protein2domainList](#)
- IntArrayList [gift::domain2proteinList](#)
- numericMatrix [gift::drugSub2proteinSubMatrix](#)
- numericMatrix [gift::observedDrug2ProteinMatrix](#)
- numericMatrix [gift::vardrugSub2proteinSubMatrix](#)
- std::vector< double > [gift::loglikelyArray](#)
- name2IndexHash [gift::drugName2Index](#)
- name2IndexHash [gift::proteinName2Index](#)
- nameList [gift::drugNameList](#)
- nameList [gift::proteinNameList](#)
- nameList [gift::drugSubNameList](#)
- nameList [gift::proteinSubNameList](#)
- nameList [gift::predictDrugNameList](#)
- nameList [gift::predictProteinNameList](#)
- nameList [gift::predictDrugNameList_WithSubs](#)
- nameList [gift::predictProteinNameList_WithSubs](#)
- IntArrayList [gift::predictDrug2SubList](#)
- IntArrayList [gift::predictProtein2SubList](#)

6.4 main.cpp File Reference

```
#include <ctime>
#include <chrono>
#include <iostream>
#include <boost/program_options.hpp>
#include "gift.hpp"
```

Functions

- `int main (int argc, char **argv)`

6.4.1 Function Documentation

6.4.1.1 `int main (int argc, char ** argv)`

References `gift::helpGift()`, `gift::outRecord()`, `gift::EM::outTrainResult()`, `gift::EM::outTrainVariance()`, `gift::EM::predictEM()`, `gift::parameters::task`, `gift::EM::trainEM()`, `gift::updateTime()`, `gift::EM::varEM()`, and `gift::version()`.

```
25                                     {
26     if (argc < 2){
27         gift::helpGift();
28         return SUCCESS;
29     } // end of if.
30     // read program options.
31     std::string configureFileName;
32     // reference to: radmangames online post.
33     try{
34         namespace po = boost::program_options;
35         po::options_description desc("Options");
36         desc.add_options()
37             ("help,h", "Print help messages.")
38             ("version,v", "Print version information.")
39             ("configure,c", po::value<std::string>(&configureFileName)->required(),
40              "Read the configure file.");
41         po::variables_map vm;
42         try{
43             po::store(po::parse_command_line(argc, argv, desc), vm);
44             if(vm.count("help") || vm.count("-h")) {
45                 gift::helpGift();
46             } // end of if
47             if(vm.count("version") || vm.count("-v")) {
48                 std::cout<<"GIFT VERSION: "<<gift::version<<std::endl;
49                 std::cout<<"UPDATE TIME: "<<gift::updateTime<<std::endl;
50             } // end of if
51         }
52         po::notify(vm); // throw an error if there are any problems.
53     } catch(po::error& e){
54         std::cerr<<"ERROR: " <<e.what()<<std::endl<<std::endl;
55         std::cerr<< desc <<std::endl;
56         return ERROR_IN_COMMAND_LINE;
57     } // end of catch
58 } catch(std::exception& e){
59     std::cerr<< "Unhandled Exception reached the top of main: "
60         <<e.what() <<", gift will now exit."<<std::endl;
61     return ERROR_UNHANDLED_EXCEPTION;
62 } // end of catch
63
64 // run program based on task.
65 try{
66     std::cout<<"Now start running gift..."<<std::endl;
67     std::chrono::system_clock::time_point timeS =
68         std::chrono::system_clock::now();
69     std::time_t PtimeS = std::chrono::system_clock::to_time_t(timeS);
70     std::cout<<"Job start at "<<std::ctime(&PtimeS)<<std::endl;
71
72     gift::parameters getParameters(configureFileName);
73     gift::EM EMgiftor(getParameters);
```

```
75     if (getParameters.task.compare("predict") == 0 ) {
76         // check is it enough?
77         EMgiftor.predictEM();
78         gift::outRecord(getParameters);
79     } else if (getParameters.task.compare("train") == 0){
80         // check is it enough?
81         EMgiftor.trainEM();
82         EMgiftor.varEM();
83         // out train result.
84         gift::outRecord(getParameters);
85         EMgiftor.outTrainResult();
86         EMgiftor.outTrainVariance();
87     } else {
88         std::cout<<"NO TASK IS SPECIFIED."<<std::endl;
89         return ERROR_IN_TASK;
90     } // end of if else if else.
91 } catch (const std::string & e){
92     std::cerr<<"ERROR: " <<e<<std::endl;
93     return ERROR_IN_READFILE;
94 }
95 return SUCCESS;
96 } // end of main
```

6.5 parameterGift.cpp File Reference

```
#include <boost/program_options.hpp>
#include <boost/any.hpp>
#include "gift.hpp"
```

Namespaces

- [gift](#)

Functions

- `const char * gift::BoolToString (bool b)`

Index

~EM
 gift::EM, 20

alphaEB
 gift::parameters, 37

author
 gift, 8

betaEB
 gift::parameters, 37

BoolToString
 gift, 8

CPIsRec
 gift::parameters, 38

chemfpRec
 gift::parameters, 38

colNum
 gift::rowCol, 41

domain2proteinList
 gift, 16

domainNum
 gift::parameters, 38

drug2proteinFileName
 gift::parameters, 38

drug2proteinList
 gift, 16

drug2subFileName
 gift::parameters, 38

drug2subList
 gift, 16

drugName2Index
 gift, 16

drugNameList
 gift, 16

drugNameListFile
 gift::parameters, 38

drugNum
 gift::parameters, 38

drugSub2proteinSubFileName
 gift::parameters, 38

drugSub2proteinSubMatrix
 gift, 16

drugSubNameList
 gift, 17

drugSubNameListFile
 gift::parameters, 38

EMGift.cpp, 43

EStep
 gift::EM, 20

EStepThread
 gift::EM, 20

EM
 gift::EM, 20

email
 gift, 8

fn
 gift::parameters, 38

fp
 gift::parameters, 38

functionThread
 gift, 9
 gift::EM, 21

functionsGift.cpp, 43

getIndexFromHash
 gift, 9

gift, 7
 author, 8
 BoolToString, 8
 domain2proteinList, 16
 drug2proteinList, 16
 drug2subList, 16
 drugName2Index, 16
 drugNameList, 16
 drugSub2proteinSubMatrix, 16
 drugSubNameList, 17
 email, 8
 functionThread, 9
 getIndexFromHash, 9
 helpGift, 9
 IntArrayList, 8
 IntList, 8
 loglikelyArray, 17
 Matrix2Fingerprints, 10
 Matrix2FingerprintsByColumn, 11
 name2IndexHash, 8
 nameList, 8
 numericMatrix, 8
 observedDrug2ProteinMatrix, 17
 outRecord, 12
 predictDrug2SubList, 17
 predictDrugNameList, 17
 predictDrugNameList_WithSubs, 17
 predictProtein2SubList, 17
 predictProteinNameList, 17
 predictProteinNameList_WithSubs, 17
 printIntArrayList, 12

- printMatrix, 13
- protein2domainList, 17
- proteinName2Index, 18
- proteinNameList, 18
- proteinSubNameList, 18
- readMatrix, 13
- readName2IndexHash, 13
- readNameListFromFile, 14
- readNameMatrixFromFile, 14
- recLogLeastNum, 18
- rowColFile, 15
- sub2drugList, 18
- updateTime, 16
- vardrugSub2proteinSubMatrix, 18
- version, 16
- writeMatrix, 16
- gift.hpp, 44
- gift::EM, 19
 - ~EM, 20
 - EStep, 20
 - EStepThread, 20
 - EM, 20
 - functionThread, 21
 - iterdrugSub2ProteinSub, 21
 - MStep, 21
 - MStepThread, 22
 - outTrainResult, 22
 - outTrainVariance, 22
 - predictDrugs, 23
 - predictDrugsProteins, 23
 - predictDrugsProteinsWithSubs, 24
 - predictDrugsWithSubs, 25
 - predictDrugsWithSubsProteins, 25
 - predictDrugsWithSubsProteinsWithSubs, 26
 - predictEM, 27
 - predictProteins, 27
 - predictProteinsWithSubs, 28
 - recLoglikely, 29
 - setLoglikely, 29
 - setPointerDrug2Protein, 29
 - setPointerDrug2Sub, 29
 - setPointerDrugSub2ProteinSub, 29
 - setPointerProtein2Sub, 30
 - trainEM, 30
 - varEM, 30
- gift::parameters, 31
 - alphaEB, 37
 - betaEB, 37
 - CPIsRec, 38
 - chemfpRec, 38
 - domainNum, 38
 - drug2proteinFileName, 38
 - drug2subFileName, 38
 - drugNameListFile, 38
 - drugNum, 38
 - drugSub2proteinSubFileName, 38
 - drugSubNameListFile, 38
 - fn, 38
 - fp, 38
 - InitDrugName2Index, 34
 - InitDrugSub2ProteinSub, 35
 - InitDrugSubNameList, 35
 - InitObservedDrug2ProteinMatrix, 35
 - InitPredictParameters, 36
 - InitProteinName2Index, 36
 - InitProteinSubNameList, 36
 - InitVarDrugSub2ProteinSub, 37
 - inputDelims, 38
 - iterNum, 38
 - loglikelyRecord, 39
 - outDrugSub2ProteinSubFileName, 39
 - outPredictCPIsFileName, 39
 - outRecordFileName, 39
 - outVarDrugSub2proteinSubFileName, 39
 - outputDelims, 39
 - parameters, 32
 - predictDrugsFileName, 39
 - predictDrugsFileName_WithSubs, 39
 - predictProteinsFileName, 39
 - predictProteinsFileName_WithSubs, 39
 - protein2subFileName, 39
 - proteinNameListFile, 39
 - proteinNum, 39
 - proteinSubNameListFile, 40
 - proteinfRec, 39
 - setDomainNum, 37
 - setDrugNum, 37
 - setProteinNum, 37
 - setSubNum, 37
 - subNum, 40
 - task, 40
 - thread, 40
- gift::rowCol, 40
 - colNum, 41
 - rowCol, 40
 - rowNum, 41
- helpGift
 - gift, 9
- InitDrugName2Index
 - gift::parameters, 34
- InitDrugSub2ProteinSub
 - gift::parameters, 35
- InitDrugSubNameList
 - gift::parameters, 35
- InitObservedDrug2ProteinMatrix
 - gift::parameters, 35
- InitPredictParameters
 - gift::parameters, 36
- InitProteinName2Index
 - gift::parameters, 36
- InitProteinSubNameList
 - gift::parameters, 36
- InitVarDrugSub2ProteinSub
 - gift::parameters, 37
- inputDelims

- gift::parameters, 38
- IntArrayList
 - gift, 8
- IntList
 - gift, 8
- iterNum
 - gift::parameters, 38
- iterdrugSub2ProteinSub
 - gift::EM, 21
- loglikelyArray
 - gift, 17
- loglikelyRecord
 - gift::parameters, 39
- MStep
 - gift::EM, 21
- MStepThread
 - gift::EM, 22
- main
 - main.cpp, 46
- main.cpp, 46
 - main, 46
- Matrix2Fingerprints
 - gift, 10
- Matrix2FingerprintsByColumn
 - gift, 11
- name2IndexHash
 - gift, 8
- nameList
 - gift, 8
- numericMatrix
 - gift, 8
- observedDrug2ProteinMatrix
 - gift, 17
- outDrugSub2ProteinSubFileName
 - gift::parameters, 39
- outPredictCPIsFileName
 - gift::parameters, 39
- outRecord
 - gift, 12
- outRecordFileName
 - gift::parameters, 39
- outTrainResult
 - gift::EM, 22
- outTrainVariance
 - gift::EM, 22
- outVarDrugSub2proteinSubFileName
 - gift::parameters, 39
- outputDelims
 - gift::parameters, 39
- parameterGift.cpp, 47
- parameters
 - gift::parameters, 32
- predictDrug2SubList
 - gift, 17
- predictDrugNameList
 - gift, 17
- predictDrugNameList_WithSubs
 - gift, 17
- predictDrugs
 - gift::EM, 23
- predictDrugsFileName
 - gift::parameters, 39
- predictDrugsFileName_WithSubs
 - gift::parameters, 39
- predictDrugsProteins
 - gift::EM, 23
- predictDrugsProteinsWithSubs
 - gift::EM, 24
- predictDrugsWithSubs
 - gift::EM, 25
- predictDrugsWithSubsProteins
 - gift::EM, 25
- predictDrugsWithSubsProteinsWithSubs
 - gift::EM, 26
- predictEM
 - gift::EM, 27
- predictProtein2SubList
 - gift, 17
- predictProteinNameList
 - gift, 17
- predictProteinNameList_WithSubs
 - gift, 17
- predictProteins
 - gift::EM, 27
- predictProteinsFileName
 - gift::parameters, 39
- predictProteinsFileName_WithSubs
 - gift::parameters, 39
- predictProteinsWithSubs
 - gift::EM, 28
- printIntArrayList
 - gift, 12
- printMatrix
 - gift, 13
- protein2domainList
 - gift, 17
- protein2subFileName
 - gift::parameters, 39
- proteinName2Index
 - gift, 18
- proteinNameList
 - gift, 18
- proteinNameListFile
 - gift::parameters, 39
- proteinNum
 - gift::parameters, 39
- proteinSubNameList
 - gift, 18
- proteinSubNameListFile
 - gift::parameters, 40
- proteinfpRec
 - gift::parameters, 39

readMatrix
 gift, 13
readName2IndexHash
 gift, 13
readNameListFromFile
 gift, 14
readNameMatrixFromFile
 gift, 14
recLogLeastNum
 gift, 18
recLoglikely
 gift::EM, 29
rowCol
 gift::rowCol, 40
rowColFile
 gift, 15
rowNum
 gift::rowCol, 41

setDomainNum
 gift::parameters, 37
setDrugNum
 gift::parameters, 37
setLoglikely
 gift::EM, 29
setPointerDrug2Protein
 gift::EM, 29
setPointerDrug2Sub
 gift::EM, 29
setPointerDrugSub2ProteinSub
 gift::EM, 29
setPointerProtein2Sub
 gift::EM, 30
setProteinNum
 gift::parameters, 37
setSubNum
 gift::parameters, 37
sub2drugList
 gift, 18
subNum
 gift::parameters, 40

task
 gift::parameters, 40
thread
 gift::parameters, 40
trainEM
 gift::EM, 30

updateTime
 gift, 16

varEM
 gift::EM, 30
vardrugSub2proteinSubMatrix
 gift, 18
version
 gift, 16

writeMatrix

gift, 16