analyzeAlignments

Generated by Doxygen 1.9.7

Chapter 1

Overview

A C++14 library and software to extract unique sequences from FASTA alignments. These unique sequences can be found by scanning the entire alignment, providing a position and window length, or providing a query sequence.

1.1 Dependencies

Building the library and binaries requires a C++14 compiler. The build process requires `cmake` version 3.11 or later. Sequence query uses Smith-Watemrman alignment, implemented here. This code is included as a submodule.

1.2 Download and install

The repository comes with a submodule, so to clone use

```
git clone --recurse-submodules https://github.com/tonymugen/analyzeAlignments
```

Next, create a build directory

```
cd vash
mkdir build
```

Finally, run `cmake` to build and install the software

```
cd build
cmake -DCMAKE_BUILD_TYPE=Release ..
cmake --build .
cmake --install .
```

Installation may require root privileges.

1.3 Tests

Optionally, one can also build the unit tests. These require Catch2, although its installation is taken care of by `cmake`. To build the tests, create a `build-Tests` directory, say, and run

```
cd build
cmake -DCMAKE_BUILD_TYPE=Test -DBUILD_TESTS=ON ..
cmake --build .
```

To run the tests from the build directory, simply run

```
./tests
```

1.4 Binaries

Two binaries are built as part of the project. Command line flags and their descriptions can be printed by running the programs without parameters.

1.4.1 homoruns

The `homoruns` binary takes an alignment and sliding window parameters (window and step size) and outputs unique sequence counts for each window. Sequences themselves are not saved, but counts are reported for each unique sequence.

1.4.2 extractWindow

The `extractWindow` binary takes an alignment and either a start window position and length or a query sequence. It returns all unique sequences in the window (or best matches to the query) with their counts. The sequences can be optionally sorted by their counts in descending order.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 BayesicSpace Namespace Reference

Classes

- struct AlignmentStatistics

  Collection of alignment statistics.
- class ParseFASTA

  FASTA alignment parser.

Functions

- void parseCL (int &argc, char **argv, std::unordered_map< std::string, std::string > &cli)

  Command line parser.
- void extractCLinfo (const std::unordered_map< std::string, std::string > &parsedCLI, std::unordered_map< std::string, int > &intVariables, std::unordered_map< std::string, std::string > &stringVariables)

  Extract parameters from parsed command line interface flags.
- void saveDiversityTable (const std::vector< std::pair< size_t, std::vector< uint32_t > > > &diversityTable, std::fstream &outFile)

  Save the diversity table.
- void saveUniqueSequences (const std::unordered_map< std::string, uint32_t > &uniqueSequences, const std::string &consensus, const std::string &fileType, std::fstream &outFile)

  Save unique sequences.
- void saveUniqueSequences (const std::vector< std::pair< std::string, uint32_t > > &uniqueSequences, const std::string &consensus, const std::string &fileType, std::fstream &outFile)

  Save sorted unique sequences.
- void saveUniqueSequences (const std::unordered_map< std::string, uint32_t > &uniqueSequences, const std::string &consensus, const AlignmentStatistics &alignStats, const std::string &query, const std::string &fileType, std::fstream &outFile)

  Save unique sequences with query.
- void saveUniqueSequences (const std::vector< std::pair< std::string, uint32_t > > &uniqueSequences, const std::string &consensus, const AlignmentStatistics &alignStats, const std::string &query, const std::string &fileType, std::fstream &outFile)

  Save sorted unique sequences with query.

5.1.1 Function Documentation

5.1.1.1 extractCLinfo()

```
void BayesicSpace::extractCLinfo (
            const std::unordered_map< std::string, std::string > & parsedCLI,
            std::unordered_map< std::string, int > & intVariables,
            std::unordered_map< std::string, std::string > & stringVariables )
```

Extract parameters from parsed command line interface flags.

Extracts needed variable values, indexed by `std::string` encoded variable names.

Parameters

`in`	*parsedCLI*	flag values parsed from the command line
`out`	*intVariables*	indexed `int` variables for use by `main()`
`out`	*stringVariables*	indexed `std::string` variables for use by `main()`

5.1.1.2 parseCL()

```
void BayesicSpace::parseCL (
            int & argc,
            char ** argv,
            std::unordered_map< std::string, std::string > & cli )
```

Command line parser.

Maps flags to values. Flags assumed to be of the form `--flag-name value`.

Parameters

`in`	*argc*	size of the `argv` array
`in`	*argv*	command line input array
`out`	*cli*	map of tags to values

5.1.1.3 saveDiversityTable()

```
void BayesicSpace::saveDiversityTable (
            const std::vector< std::pair< size_t, std::vector< uint32_t > > > & diversityTable,
            std::fstream & outFile )
```

Save the diversity table.

Save the diversity table. The output file will have two columns: (1) window start position (repeated for every unique sequence). (2) number of unique sequence occurrences.

Parameters

in	*diversityTable*	the diversity table data
in,out	*outFile*	output file stream

5.1.1.4 saveUniqueSequences() [1/4]

```
void BayesicSpace::saveUniqueSequences (
            const std::unordered_map< std::string, uint32_t > & uniqueSequences,
            const std::string & consensus,
            const AlignmentStatistics & alignStats,
            const std::string & query,
            const std::string & fileType,
            std::fstream & outFile )
```

Save unique sequences with query.

Save unique sequences in an alignment window. If in FASTA format, the number of times each sequence appears in an alignment is in the header. If in TAB format, sequence and the number of occurrences are on the same line, separated by a tab. The query sequence is displayed on the top line, may be different length than the rest of the sequences if there are insertions/deletions. The consensus is displayed on the second line, marked by ¨C¨ in the TAB format. The start position and length of the widow are also included. They are explicitly described in the consensus FASTA header, or included with a ¨|¨ delimiter in the TAB format. Nucleotides that are the same as the consensus are displayed as '.', the different residues are shown.

Parameters

in	*uniqueSequences*	table of unique sequences and their counts
in	*consensus*	consensus sequence for the window
in	*alignStats*	alignment statistics
in	*query*	query sequence
in	*fileType*	TAB or FASTA, otherwise throws
in,out	*outFile*	output stream

5.1.1.5 saveUniqueSequences() [2/4]

```
void BayesicSpace::saveUniqueSequences (
            const std::unordered_map< std::string, uint32_t > & uniqueSequences,
            const std::string & consensus,
            const std::string & fileType,
            std::fstream & outFile )
```

Save unique sequences.

Save unique sequences in an alignment window. If in FASTA format, the number of times each sequence appears in an alignment is in the header. If in TAB format, sequence and the number of occurrences are on the same line, separated by a tab. The consensus is displayed on the top line. Nucleotides that are the same as the consensus are displayed as '.', the different residues are shown.

Parameters

in	*uniqueSequences*	table of unique sequences and their counts
in	*consensus*	consensus sequence for the window
in	*fileType*	TAB or FASTA, otherwise throws
in,out	*outFile*	output stream

5.1.1.6 saveUniqueSequences() [3/4]

```
void BayesicSpace::saveUniqueSequences (
            const std::vector< std::pair< std::string, uint32_t > > & uniqueSequences,
            const std::string & consensus,
            const AlignmentStatistics & alignStats,
            const std::string & query,
            const std::string & fileType,
            std::fstream & outFile )
```

Save sorted unique sequences with query.

Save unique sequences in an alignment window. If in FASTA format, the number of times each sequence appears in an alignment is in the header. If in TAB format, sequence and the number of occurrences are on the same line, separated by a tab. The query sequence is displayed on the top line, may be different length than the rest of the sequences if there are insertions/deletions. The consensus is displayed on the second line, marked by ¨C¨ in the TAB format. The start position and length of the widow are also included. They are explicitly described in the consensus FASTA header, or included with a ¨|¨ delimiter in the TAB format. Nucleotides that are the same as the consensus are displayed as '.', the different residues are shown. Sequences are sorted by the number of occurrences in descending order.

Parameters

in	*uniqueSequences*	table of unique sequences and their counts
in	*consensus*	consensus sequence for the window
in	*alignStats*	alignment statistics
in	*query*	query sequence
in	*fileType*	TAB or FASTA, otherwise throws
in,out	*outFile*	output stream

5.1.1.7 saveUniqueSequences() [4/4]

```
void BayesicSpace::saveUniqueSequences (
            const std::vector< std::pair< std::string, uint32_t > > & uniqueSequences,
            const std::string & consensus,
            const std::string & fileType,
            std::fstream & outFile )
```

Save sorted unique sequences.

Save unique sequences in an alignment window. If in FASTA format, the number of times each sequence appears in an alignment is in the header. If in TAB format, sequence and the number of occurrences are on the same line, separated by a tab. The consensus is displayed on the top line. Nucleotides that are the same as the consensus are displayed as '.', the different residues are shown. Sequences are sorted by the number of occurrences in descending order.

Parameters

in	*uniqueSequences*	table of unique sequences and their counts
in	*consensus*	consensus sequence for the window
in	*fileType*	TAB or FASTA, otherwise throws
in,out	*outFile*	output stream

Chapter 6

Class Documentation

6.1 BayesicSpace::AlignmentStatistics Struct Reference

Collection of alignment statistics.

`#include <fastaParser.hpp>`

Public Attributes

- size_t referenceStart
- size_t referenceLength
- size_t queryStart
- size_t queryLength

6.1.1 Detailed Description

Collection of alignment statistics.

Collects striped Smith-Waterman alignment statistics.

6.1.2 Member Data Documentation

6.1.2.1 queryLength

```
size_t BayesicSpace::AlignmentStatistics::queryLength
```

6.1.2.2 queryStart

```
size_t BayesicSpace::AlignmentStatistics::queryStart
```

6.1.2.3 referenceLength

```
size_t BayesicSpace::AlignmentStatistics::referenceLength
```

6.1.2.4 referenceStart

```
size_t BayesicSpace::AlignmentStatistics::referenceStart
```

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

- include/fastaParser.hpp

6.2 BayesicSpace::ParseFASTA Class Reference

FASTA alignment parser.

```
#include <fastaParser.hpp>
```

Public Member Functions

- ParseFASTA ()=default
  Default constructor.
- ParseFASTA (const std::string &fastaFileName)
  Constructor from FASTA file.
- ParseFASTA (const ParseFASTA &toCopy)
  Copy constructor.
- ParseFASTA (ParseFASTA &&toMove) noexcept
  Move constructor.
- ParseFASTA & operator= (const ParseFASTA &toCopy)
  Copy assignment operator.
- ParseFASTA & operator= (ParseFASTA &&toMove) noexcept
  Move assignment operator.
- ~ParseFASTA ()=default
  Destructor.
- size_t sequenceNumber () const noexcept
  Number of sequences in alignment.
- size_t alignmentLength () const
  Alignment length.
- std::string extractConsensusWindow (const size_t &startIdx, const size_t &windowLength) const
  Extract a consensus region.
- std::vector< std::pair< size_t, std::vector< uint32_t > > > diversityInWindows (const size_t &windowSize, const size_t &stepSize) const
  Sequence diversity in windows.

- std::unordered_map< std::string, uint32_t > extractWindow (const size_t &windowStartPosition, const size_t &windowSize) const

  Extract an alignment window.
- std::vector< std::pair< std::string, uint32_t > > extractWindowSorted (const size_t &windowStartPosition, const size_t &windowSize) const

  Extract an alignment window and sort.
- AlignmentStatistics extractSequence (const std::string &querySequence) const

  Extract a region matching a sequence.
- void imputeMissing ()

  Impute missing values.

6.2.1 Detailed Description

FASTA alignment parser.

Reads a FASTA alignment file, separates the sequences and headers, and provides analysis methods. The data are stored in memory, so users should pay attention to file sizes.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 ParseFASTA() [1/4]

```
BayesicSpace::ParseFASTA::ParseFASTA ( )  [default]
```

Default constructor.

6.2.2.2 ParseFASTA() [2/4]

```
ParseFASTA::ParseFASTA (
            const std::string & fastaFileName )
```

Constructor from FASTA file.

Read data from a FASTA file.

Parameters

in	*fastaFileName*	input FASTA file name

6.2.2.3 ParseFASTA() [3/4]

```
ParseFASTA::ParseFASTA (
            const ParseFASTA & toCopy )
```

Copy constructor.

Parameters

in	*toCopy*	object to copy

6.2.2.4 ParseFASTA() [4/4]

```
ParseFASTA::ParseFASTA (
            ParseFASTA && toMove )  [noexcept]
```

Move constructor.

Parameters

in	*toMove*	object to move

6.2.2.5 ~ParseFASTA()

```
BayesicSpace::ParseFASTA::~ParseFASTA ( )  [default]
```

Destructor.

6.2.3 Member Function Documentation

6.2.3.1 alignmentLength()

```
size_t BayesicSpace::ParseFASTA::alignmentLength ( ) const  [inline]
```

Alignment length.

Returns

alignment length

6.2.3.2 diversityInWindows()

```
std::vector< std::pair< size_t, std::vector< uint32_t > > > ParseFASTA::diversityInWindows (
            const size_t & windowSize,
            const size_t & stepSize ) const
```

Sequence diversity in windows.

Calculate the number of different sequences in window sliding along a sequence alignment. Reports the number of times each unique sequence occurs by window position.

Parameters

in	*windowSize*	window size in base pairs
in	*stepSize*	window movement steps in base pairs

Returns

vector of pairs that contain window start positions and unique sequence counts

6.2.3.3 extractConsensusWindow()

```
std::string ParseFASTA::extractConsensusWindow (
            const size_t & startIdx,
            const size_t & windowLength ) const
```

Extract a consensus region.

Extract a window of the consensus sequence.

Parameters

in	*startIdx*	index of the window start
in	*windowLength*	number of nucleotides in the window

6.2.3.4 extractSequence()

```
AlignmentStatistics ParseFASTA::extractSequence (
            const std::string & querySequence ) const
```

Extract a region matching a sequence.

Report all unique sequences (and their counts) matching the query sequence. Matching performed using striped Smith-Waterman alignment.

Parameters

in	*querySequence*	the query sequence

Returns

matching window start and length

6.2.3.5 extractWindow()

```
std::unordered_map< std::string, uint32_t > ParseFASTA::extractWindow (
```

```
            const size_t & windowStartPosition,
            const size_t & windowSize ) const
```

Extract an alignment window.

Calculates the number of different sequences in a window. Reports the number of times each unique sequence occurs in the provided window.

Parameters

in	*windowStartPosition*	window start
in	*windowSize*	window size in base pairs

Returns

map of sequences to the number of times each occurs in the alignment

6.2.3.6 extractWindowSorted()

```
std::vector< std::pair< std::string, uint32_t > > ParseFASTA::extractWindowSorted (
            const size_t & windowStartPosition,
            const size_t & windowSize ) const
```

Extract an alignment window and sort.

Calculates the number of different sequences in a window. Reports the number of times each unique sequence occurs in the provided window. The output is sorted by the number of times a sequence is present, in descending order.

Parameters

in	*windowStartPosition*	window start
in	*windowSize*	window size in base pairs

Returns

map of sequences to the number of times each occurs in the alignment, sorted

6.2.3.7 imputeMissing()

```
void ParseFASTA::imputeMissing ( )
```

Impute missing values.

Replaces missing (N or other variants, e.g. Y, S, etc.) nucleotides with the consensus value.

6.2.3.8 operator=() [1/2]

```
ParseFASTA & ParseFASTA::operator= (
            const ParseFASTA & toCopy )
```

Copy assignment operator.

Parameters

in	*toCopy*	object to copy

6.2.3.9 operator=() [2/2]

```
ParseFASTA & ParseFASTA::operator= (
            ParseFASTA && toMove )  [noexcept]
```

Move assignment operator.

Parameters

in	*toMove*	object to move

6.2.3.10 sequenceNumber()

```
size_t BayesicSpace::ParseFASTA::sequenceNumber ( ) const  [inline], [noexcept]
```

Number of sequences in alignment.

Returns

number of sequences in the alignment

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

- include/fastaParser.hpp
- src/fastaParser.cpp

Chapter 7

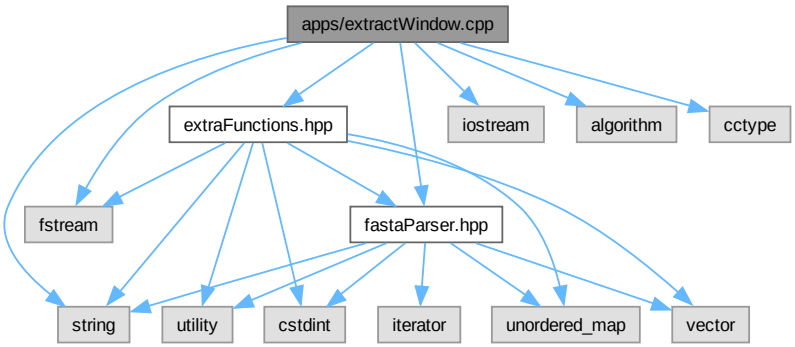
File Documentation

7.1 apps/extractWindow.cpp File Reference

Extract unique sequences from an alignment segment.

```
#include <fstream>
#include <iostream>
#include <algorithm>
#include <cctype>
#include <string>
#include ¨extraFunctions.hpp¨
#include ¨fastaParser.hpp¨
```

Include dependency graph for extractWindow.cpp:



Functions

- int main (int argc, char ∗argv[ ])

7.1.1 Detailed Description

Extract unique sequences from an alignment segment.

Author

Anthony J. Greenberg

Copyright

Version

0.1

Read a FASTA alignment file and, extract a segment, and save to a separate file.

7.1.2 Function Documentation

7.1.2.1 main()

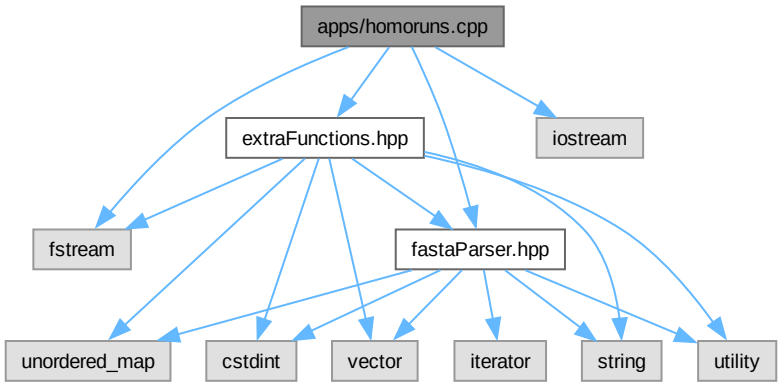
```
int main (
            int argc,
            char * argv[] )
```

7.2 apps/homoruns.cpp File Reference

Identify homozygosity runs.

```
#include <fstream>
#include <iostream>
#include ¨extraFunctions.hpp¨
#include ¨fastaParser.hpp¨
```

Include dependency graph for homoruns.cpp:



Functions

- int main (int argc, char ∗argv[ ])

7.2.1 Detailed Description

Identify homozygosity runs.

Author

Anthony J. Greenberg

Copyright

Copyright (c) 2023

Version

0.1

Read a FASTA alignment file and identify low-diversity regions.

7.2.2 Function Documentation

7.2.2.1 main()

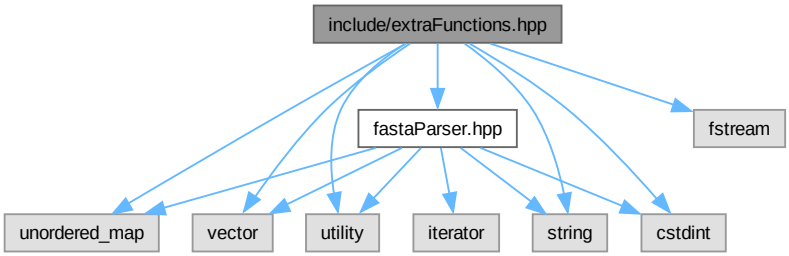
```
int main (
            int argc,
            char * argv[] )
```

7.3 include/extraFunctions.hpp File Reference

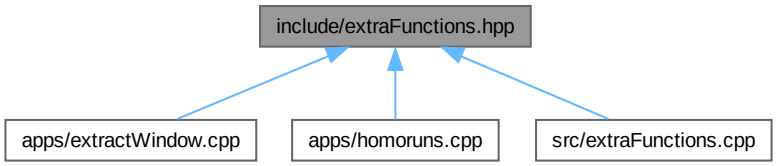
Extra functions.

```
#include <unordered_map>
#include <vector>
#include <utility>
#include <string>
#include <cstdint>
#include <fstream>
```

`#include ¨fastaParser.hpp¨`
Include dependency graph for extraFunctions.hpp:



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



Namespaces

- namespace BayesicSpace

Functions

- void BayesicSpace::parseCL (int &argc, char **argv, std::unordered_map< std::string, std::string > &cli)

  Command line parser.
- void BayesicSpace::extractCLinfo (const std::unordered_map< std::string, std::string > &parsedCLI, std::unordered_map< std::string, int > &intVariables, std::unordered_map< std::string, std::string > &stringVariables)

  Extract parameters from parsed command line interface flags.
- void BayesicSpace::saveDiversityTable (const std::vector< std::pair< size_t, std::vector< uint32_t > > > &diversityTable, std::fstream &outFile)

  Save the diversity table.
- void BayesicSpace::saveUniqueSequences (const std::unordered_map< std::string, uint32_t > &uniqueSequences, const std::string &consensus, const std::string &fileType, std::fstream &outFile)

  Save unique sequences.

- void BayesicSpace::saveUniqueSequences (const std::vector< std::pair< std::string, uint32_t > > &unique↩ Sequences, const std::string &consensus, const std::string &fileType, std::fstream &outFile)

  Save sorted unique sequences.
- void BayesicSpace::saveUniqueSequences (const std::unordered_map< std::string, uint32_t > &unique↩ Sequences, const std::string &consensus, const AlignmentStatistics &alignStats, const std::string &query, const std::string &fileType, std::fstream &outFile)

  Save unique sequences with query.
- void BayesicSpace::saveUniqueSequences (const std::vector< std::pair< std::string, uint32_t > > &unique↩ Sequences, const std::string &consensus, const AlignmentStatistics &alignStats, const std::string &query, const std::string &fileType, std::fstream &outFile)

  Save sorted unique sequences with query.

7.3.1 Detailed Description

Extra functions.

Author

Anthony J. Greenberg

Copyright

Version

0.1

Definitions of extra utility functions for the FASTA alignment analysis project.

7.4 extraFunctions.hpp

Go to the documentation of this file.

```
00001 /*
00002  * Copyright (c) 2023 Anthony J. Greenberg
00003  *
00004  * Redistribution and use in source and binary forms, with or without modification, are permitted provided
      that the following conditions are met:
00005  *
00006  * 1. Redistributions of source code must retain the above copyright notice, this list of conditions and
      the following disclaimer.
00007  *
00008  * 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions
      and the following disclaimer in the documentation and/or other materials provided with the distribution.
00009  *
00010  * 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or
      promote products derived from this software without specific prior written permission.
00011  *
00012  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED
      WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
00013  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
      EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
```

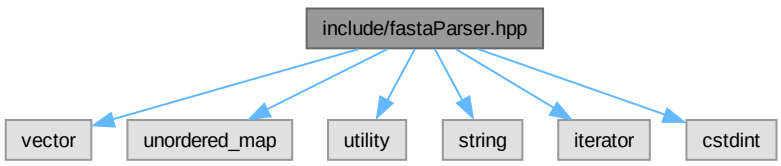
```
00014  * BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
       (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
00015  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
       AND ON ANY THEORY OF LIABILITY, WHETHER
00016  * IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
       THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
00017  * THE POSSIBILITY OF SUCH DAMAGE.
00018  */
00019
00021
00030 #pragma once
00031
00032 #include <unordered_map>
00033 #include <vector>
00034 #include <utility> // for std::pair
00035 #include <string>
00036 #include <cstdint>
00037 #include <fstream>
00038
00039 #include "fastaParser.hpp"
00040
00041 namespace BayesicSpace {
00050     void parseCL(int &argc, char **argv, std::unordered_map<std::string, std::string> &cli);
00059     void extractCLinfo(const std::unordered_map<std::string, std::string> &parsedCLI,
       std::unordered_map<std::string, int> &intVariables, std::unordered_map<std::string, std::string>
       &stringVariables);
00069     void saveDiversityTable(const std::vector< std::pair< size_t, std::vector<uint32_t> > >
       &diversityTable, std::fstream &outFile);
00082     void saveUniqueSequences(const std::unordered_map<std::string, uint32_t> &uniqueSequences, const
       std::string &consensus, const std::string &fileType, std::fstream &outFile);
00096     void saveUniqueSequences(const std::vector< std::pair<std::string, uint32_t> > &uniqueSequences, const
       std::string &consensus, const std::string &fileType, std::fstream &outFile);
00114     void saveUniqueSequences(const std::unordered_map<std::string, uint32_t> &uniqueSequences, const
       std::string &consensus,
00115                              const AlignmentStatistics &alignStats, const std::string &query,
00116                              const std::string &fileType, std::fstream &outFile);
00135     void saveUniqueSequences(const std::vector< std::pair<std::string, uint32_t> > &uniqueSequences, const
       std::string &consensus,
00136                              const AlignmentStatistics &alignStats, const std::string &query,
00137                              const std::string &fileType, std::fstream &outFile);
00138 }
```

7.5 include/fastaParser.hpp File Reference

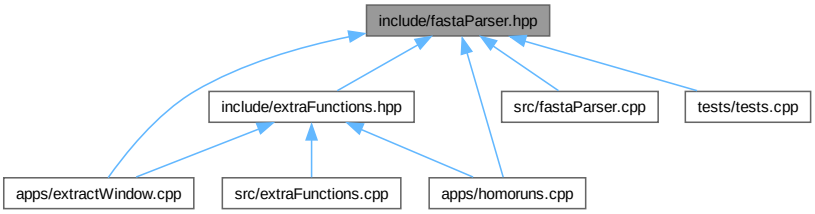
Class definitions for FASTA alignment parsing.

```
#include <vector>
#include <unordered_map>
#include <utility>
#include <string>
#include <iterator>
#include <cstdint>
```

Include dependency graph for fastaParser.hpp:



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



Classes

- struct BayesicSpace::AlignmentStatistics

  Collection of alignment statistics.
- class BayesicSpace::ParseFASTA

  FASTA alignment parser.

Namespaces

- namespace BayesicSpace

7.5.1 Detailed Description

Class definitions for FASTA alignment parsing.

Author

Anthony J. Greenberg

Copyright

Copyright (c) 2023

Version

0.1

Class for reading, parsing, and manipulating DNA sequence alignments in FASTA format.

7.6 fastaParser.hpp

Go to the documentation of this file.

```
00001 /*
00002  * Copyright (c) 2023 Anthony J. Greenberg
00003  *
00004  * Redistribution and use in source and binary forms, with or without modification, are permitted provided
      that the following conditions are met:
00005  *
00006  * 1. Redistributions of source code must retain the above copyright notice, this list of conditions and
      the following disclaimer.
00007  *
00008  * 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions
      and the following disclaimer in the documentation and/or other materials provided with the distribution.
00009  *
00010  * 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or
      promote products derived from this software without specific prior written permission.
00011  *
00012  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED
      WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
00013  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
      EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
00014  * BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
      (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
00015  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
      AND ON ANY THEORY OF LIABILITY, WHETHER
00016  * IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
      THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
00017  * THE POSSIBILITY OF SUCH DAMAGE.
00018  */
00019
00021
00030 #pragma once
00031
00032 #include <vector>
00033 #include <unordered_map>
00034 #include <utility> // for std::pair
00035 #include <string>
00036 #include <iterator>
00037 #include <cstdint>
00038
00039 namespace BayesicSpace {
00040     struct AlignmentStatistics;
00041     class ParseFASTA;
00042
00047     struct AlignmentStatistics {
00048         size_t referenceStart;
00049         size_t referenceLength;
00050         size_t queryStart;
00051         size_t queryLength;
00052     };
00059     class ParseFASTA {
00060     public:
00062         ParseFASTA() = default;
00069         ParseFASTA(const std::string &fastaFileName);
00074         ParseFASTA(const ParseFASTA &toCopy);
00079         ParseFASTA(ParseFASTA &&toMove) noexcept;
00084         ParseFASTA& operator=(const ParseFASTA &toCopy);
00089         ParseFASTA& operator=(ParseFASTA &&toMove) noexcept;
00091         ~ParseFASTA() = default;
00096         size_t sequenceNumber() const noexcept {return fastaAlignment_.size(); };
00101         size_t alignmentLength() const {return fastaAlignment_.at(0).second.size(); };
00109         std::string extractConsensusWindow(const size_t &startIdx, const size_t &windowLength) const;
00119         std::vector< std::pair< size_t, std::vector<uint32_t> > > diversityInWindows(const size_t
      &windowSize, const size_t &stepSize) const;
00129         std::unordered_map<std::string, uint32_t> extractWindow(const size_t &windowStartPosition, const
      size_t &windowSize) const;
00140         std::vector< std::pair<std::string, uint32_t> > extractWindowSorted(const size_t
      &windowStartPosition, const size_t &windowSize) const;
00149         AlignmentStatistics extractSequence(const std::string &querySequence) const;
00154         void imputeMissing();
00155     private:
00161         std::vector< std::pair<std::string, std::string> > fastaAlignment_;
00163         std::string consensus_;
00169         void makeConsensus_();
00170     };
00171 }
```

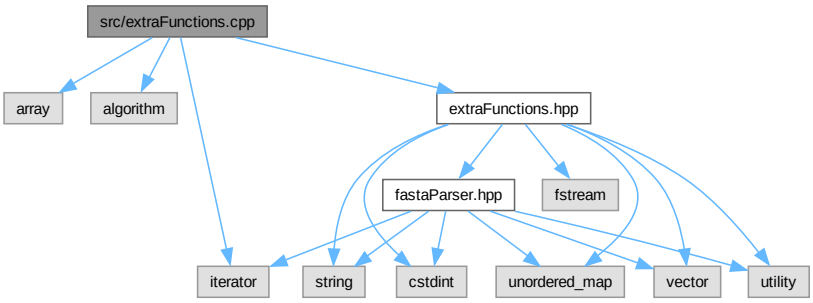
7.7 README.md File Reference

7.8 src/extraFunctions.cpp File Reference

Extra functions.

```
#include <array>
#include <algorithm>
#include <iterator>
#include ¨extraFunctions.hpp¨
```

Include dependency graph for extraFunctions.cpp:



7.8.1 Detailed Description

Extra functions.

Author

Anthony J. Greenberg

Copyright

Copyright (c) 2023

Version

0.1

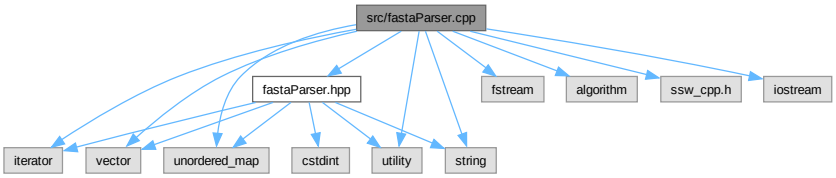
Implementation of extra utility functions for the FASTA alignment analysis project.

7.9 src/fastaParser.cpp File Reference

Implementation of FASTA alignment parsing.

```
#include <iterator>
#include <vector>
#include <unordered_map>
#include <utility>
#include <string>
#include <fstream>
#include <algorithm>
#include ¨fastaParser.hpp¨
#include ¨ssw_cpp.h¨
#include <iostream>
```

Include dependency graph for fastaParser.cpp:



7.9.1 Detailed Description

Implementation of FASTA alignment parsing.

Author

Anthony J. Greenberg

Copyright

Copyright (c) 2023

Version

0.1

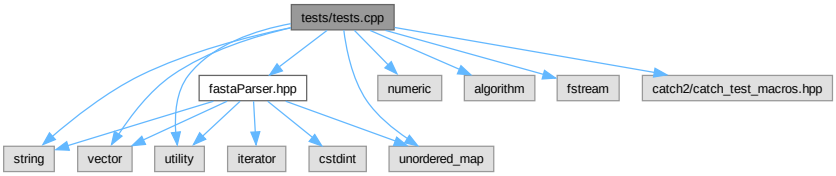
Implements the class for reading, parsing, and manipulating DNA sequence alignments in FASTA format.

7.10 tests/tests.cpp File Reference

Tests.

```
#include <string>
#include <vector>
#include <utility>
#include <numeric>
#include <algorithm>
#include <unordered_map>
#include <fstream>
#include ¨catch2/catch_test_macros.hpp¨
#include ¨fastaParser.hpp¨
```

Include dependency graph for tests.cpp:



Functions

- TEST_CASE (¨A FASTA file is properly parsed¨, ¨[parser]¨)

7.10.1 Detailed Description

Tests.

Author

Anthony J. Greenberg

Copyright

Copyright (c) 2023

Version

0.1

Tests using Catch2.

7.10.2 Function Documentation

7.10.2.1 TEST_CASE()

```
TEST_CASE (
            ¨A FASTA file is properly parsed¨ ,
            ¨¨ [parser] )
```