align2bed

Generated by Doxygen 1.8.20

Chapter 1

Overview

align2bed is a light-weight tool that extracts single nucleotide polymorphisms (SNPs) from FASTA alignments and saves them to a `plink` binary-format BED file. It is fast and can process whole-genome data on a laptop in minutes. The F↩ ASTA alignments should be in the format provided by the *Drosophila* Genome Nexus (`http://www.johnpool.↩ net/genomes.html`), i.e. the data for each line (or individual) are in a separate file with nucleotides in one line and without the customary FASTA header. Each chromosome arm has its own set of files.

The sofware uses control files that list paths to each FASTA file to be processed. An example data set is included to illustrate the necessary features of the data. One of the individuals must be marked as the reference, and it is also assumed that this reference is the outgroup. The reference genotypes are not included in the output. BED format files require the SNPs to be biallelic, so SNPs that do not meet this criterion are not included as output. However, if a SNP is biallelic within the population (non-reference) sample, but both alleles are different from the outgroup, it is included. Names of such SNPs are marked with "d" at the end (in the accompanying .bim file) to enable downstream filtering. In addition, SNPs with outgroup missing are included but their names marked with "m" at the end. SNP names are "s" plus position, then "m" or "d" if applicable, then underscore ("_"), then chromosome arm name. Note that the ancestral nucleotide (if available) is listed last in the .bim file.

While *align2bed* is tailored for the *Drosophila* Genome Nexus data, there are three ways it can be extended to similar data sets from other species. Data can be arranged to mimic the *Drosophila* set by treating chromosomes in groups of five. Alignment length can vary indefinitely. Furthermore, I wrote the program using a class that has wider applicability. Taking the *align2bed* source code as an exmaple, and reading the provided interface documentation, someone with even very limited experience in C++ can write software that applies to different data sets and hardware configurations. Finally, anyone who would like to extend functionality even further is welcome to modify the class implementation to suit their needs.

Neither *align2bed* itself nor the class used to implement it has any dependencies outside of the C++ STL. Only a compiler capable of recognizing the C++11 standard is required (I successfully compiled with LLVM and GCC). The implementation is multithreaded, with chromosome arms processed in parallel. Each thread allocates a 2Gb buffer to read the FASTA files. The *align2bed* source can be easily modified to change the threading and memory allocation parameters (see included class documentation for details).

To compile, make sure you are in the directory with the source code files and run

```
g++ align2bed.cpp sequence.cpp -o align2bed -lpthread -O3 -march=native -std=c++11
```

then copy the binary where you need it. On FreeBSD, replace `g++` with `c++`. Run by typing `./align2bed` in the directory with the binary and the data, or move into an appropriate /bin folder for global access.

The example data set includes control files for each autosome and 20 kb of alignments extracted from 284 *Drosophila* lines (283 *D. melanogaster* and a *D. simulans* outgroup). Each chromosome needs a separate control file, which simply lists the paths to FASTA files (which can include directories), one file per line. The file containing the outgroup sequence should be marked with "r:".

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 SFparse Class Reference

Sequence file parsing class.

```
#include <sequence.hpp>
```

Public Member Functions

- SFparse ()

    Default constructor.
- SFparse (const vector< string > &inFlNam, const vector< string > &lineNames, const string &refFlNam, const string &outFlNam, const string &chrNam, const unsigned short &chrNum, const string &inFlType, const string &outFlType, const unsigned long &alloc=2000000000UL)

    Constructor with vectors of names.
- SFparse (const string &fileList, const string &outFlNam, const string &chrNam, const unsigned short &chrNum, const string &inFlType, const string &outFlType, const unsigned long &alloc=2000000000UL)

    Constructor with file list and explicit types.
- SFparse (const string &fileList, const string &outFlNam, const unsigned long &alloc=2000000000UL)

    Constructor with file list.
- ∼SFparse ()

    Destructor.
- SFparse (const SFparse &inObj)

    Copy constructor.
- SFparse & operator= (const SFparse &inObj)

    Copy assignement operator.
- SFparse (SFparse &&inObj)

    Move constructor.
- SFparse & operator= (SFparse &&inObj)

    Move assignement operator.
- void changeOutType (const string &newType)

    Modify output type.

- void operator() ()

  Input file parsing.
- SFparse ()

  Default constructor.
- SFparse (const vector< string > &inFlNam, const vector< string > &lineNames, const string &refFlNam, const string &outFlNam, const string &chrNam, const unsigned short &chrNum, const string &inFlType, const string &outFlType, const unsigned long &alloc=2000000000UL)

  Constructor with vectors of names.
- SFparse (const string &fileList, const string &outFlNam, const string &chrNam, const unsigned short &chrNum, const string &inFlType, const string &outFlType, const unsigned long &alloc=2000000000UL)

  Constructor with file list and explicit types.
- SFparse (const string &fileList, const string &outFlNam, const unsigned long &alloc=2000000000UL)

  Constructor with file list.
- ∼SFparse ()

  Destructor.
- SFparse (const SFparse &inObj)

  Copy constructor.
- SFparse & operator= (const SFparse &inObj)

  Copy assignement operator.
- SFparse (SFparse &&inObj)

  Move constructor.
- SFparse & operator= (SFparse &&inObj)

  Move assignement operator.
- void changeOutType (const string &newType)

  Modify output type.
- void operator() ()

  Input file parsing.

4.1.1 Detailed Description

Sequence file parsing class.

Takes a list of files in one format and outputs one or more files in a different format, depending on settings. The data are presumed to come from a single chromosome.

Note

Formats will be added as the need arises.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 SFparse() [1/10]

```
SFparse::SFparse (
            const vector< string > & inFlNam,
            const vector< string > & lineNames,
            const string & refFlNam,
            const string & outFlNam,
            const string & chrNam,
            const unsigned short & chrNum,
            const string & inFlType,
            const string & outFlType,
            const unsigned long & alloc = 2000000000UL )
```

Constructor with vectors of names.

Takes vectors of input and output file names. Note that the number of lines cannot be bigger than maximum of *unsigned int*. This is not checked. Also, the *lineNames* vector must have one fewer elements than the *inFlNam* vector.

Parameters

in	*inFlNam*	vector of input file names
in	*lineNames*	vector of line names
in	*refFlNam*	reference file name
in	*outFlNam*	output file name
in	*chrNam*	chromosome name
in	*chrNum*	chromosome number
in	*inFlType*	input file type
in	*outFlType*	outout file type
in	*alloc*	buffer allocation in bytes

4.1.2.2 SFparse() [2/10]

```
SFparse::SFparse (
            const string & fileList,
            const string & outFlNam,
            const string & chrNam,
            const unsigned short & chrNum,
            const string & inFlType,
            const string & outFlType,
            const unsigned long & alloc = 2000000000UL )
```

Constructor with file list and explicit types.

Takes a name of a file with the list of input files. The files should be listed one per line. Types to convert from and to are specified. The reference should be marked *r:* in the list of files. Number of lines is checked and warning issued if it is larger than the *unsigned int* maximum. Line names are derived from the non-reference files which should have the names before the underscore or extension (i.e., lineName_XXX.ext or lineName.ext). Directory names will be stripped out.

Parameters

in	*fileList*	name of the control file
in	*outFlNam*	output file name
in	*chrNam*	chromosome name
in	*chrNum*	chromosome number
in	*inFlType*	input file type
in	*outFlType*	outout file type
in	*alloc*	buffer allocation in bytes

4.1.2.3 SFparse() [3/10]

```
SFparse::SFparse (
            const string & fileList,
            const string & outFlNam,
            const unsigned long & alloc = 2000000000UL )
```

Constructor with file list.

Takes a name of a file with the list of input files. The files should be listed one per line. Types are identified from file extensions. Only the first file listed for input is used (and checked) to determine the input type. The reference should be marked *r:* in the list of files. Chromosome name is derived from the output file name which should immediately precede the extension and may be preceded by an underscore (i.e., XXX_chrName.ext). Number of lines is checked and warning issued if it is larger than the *unsigned int* maximum. Line names are derived from the non-reference files which should have the names before the underscore or extension (i.e., lineName_XXX.ext or lineName.ext). Directory names will be stripped out.

Parameters

in	*fileList*	name of the control file
in	*outFlNam*	output file name
in	*alloc*	buffer allocation in bytes

4.1.2.4 SFparse() [4/10]

```
SFparse::SFparse (
            const SFparse & inObj )  [inline]
```

Copy constructor.

Parameters

in	*inObj*	object to be copied

4.1.2.5 SFparse() [5/10]

```
SFparse::SFparse (
            SFparse && inObj )  [inline]
```

Move constructor.

Parameters

in	*inObj*	object to be moved

4.1.2.6 SFparse() [6/10]

```
SFparse::SFparse (
            const vector< string > & inFlNam,
            const vector< string > & lineNames,
            const string & refFlNam,
            const string & outFlNam,
            const string & chrNam,
            const unsigned short & chrNum,
            const string & inFlType,
            const string & outFlType,
            const unsigned long & alloc = 2000000000UL )
```

Constructor with vectors of names.

Takes vectors of input and output file names. Note that the number of lines cannot be bigger than maximum of *unsigned int*. This is not checked. Also, the *lineNames* vector must have one fewer elements than the *inFlNam* vector.

Parameters

in	*inFlNam*	vector of input file names
in	*lineNames*	vector of line names
in	*refFlNam*	reference file name
in	*outFlNam*	output file name
in	*chrNam*	chromosome name
in	*chrNum*	chromosome number
in	*inFlType*	input file type
in	*outFlType*	outout file type
in	*alloc*	buffer allocation in bytes

4.1.2.7 SFparse() `[7/10]`

```
SFparse::SFparse (
             const string & fileList,
             const string & outFlNam,
             const string & chrNam,
             const unsigned short & chrNum,
             const string & inFlType,
             const string & outFlType,
             const unsigned long & alloc = 2000000000UL )
```

Constructor with file list and explicit types.

Takes a name of a file with the list of input files. The files should be listed one per line. Types to convert from and to are specified. The reference should be marked *r:* in the list of files. Number of lines is checked and warning issued if it is larger than the *unsigned int* maximum. Line names are derived from the non-reference files which should have the names before the underscore or extension (i.e., lineName_XXX.ext or lineName.ext). Directory names will be stripped out.

Parameters

in	*fileList*	name of the control file
in	*outFlNam*	output file name
in	*chrNam*	chromosome name
in	*chrNum*	chromosome number
in	*inFlType*	input file type
in	*outFlType*	outout file type
in	*alloc*	buffer allocation in bytes

4.1.2.8 SFparse() `[8/10]`

```
SFparse::SFparse (
             const string & fileList,
             const string & outFlNam,
             const unsigned long & alloc = 2000000000UL )
```

Constructor with file list.

Takes a name of a file with the list of input files. The files should be listed one per line. Types are identified from file extensions. Only the first file listed for input is used (and checked) to determine the input type. The reference should be marked *r:* in the list of files. Chromosome name is derived from the output file name which should immediately precede the extension and may be preceded by an underscore (i.e., XXX_chrName.ext). Number of lines is checked and warning issued if it is larger than the *unsigned int* maximum. Line names are derived from the non-reference files which should have the names before the underscore or extension (i.e., lineName_XXX.ext or lineName.ext). Directory names will be stripped out.

Parameters

in	*fileList*	name of the control file
in	*outFlNam*	output file name
in	*alloc*	buffer allocation in bytes

4.1.2.9 SFparse() [9/10]

```
SFparse::SFparse (
            const SFparse & inObj )  [inline]
```

Copy constructor.

Parameters

in	*inObj*	object to be copied

4.1.2.10 SFparse() [10/10]

```
SFparse::SFparse (
            SFparse && inObj )  [inline]
```

Move constructor.

Parameters

in	*inObj*	object to be moved

4.1.3 Member Function Documentation

4.1.3.1 changeOutType() [1/2]

```
void SFparse::changeOutType (
            const string & newType )  [inline]
```

Modify output type.

Parameters

in	*newType*	new output format

4.1.3.2 changeOutType() [2/2]

```
void SFparse::changeOutType (
            const string & newType )  [inline]
```

Modify output type.

Parameters

in	*newType*	new output format

4.1.3.3 operator()() [1/2]

```
void SFparse::operator() ( )
```

Input file parsing.

Function operator performs parsing of input files. Saves the results to the output file(s).

4.1.3.4 operator()() [2/2]

```
void SFparse::operator() ( )
```

Input file parsing.

Function operator performs parsing of input files. Saves the results to the output file(s).

4.1.3.5 operator=() [1/4]

```
SFparse & SFparse::operator= (
            const SFparse & inObj )
```

Copy assignement operator.

Parameters

in	*inObj*	object to be copied

Returns

SFparse object

4.1.3.6 operator=() [2/4]

```
SFparse& SFparse::operator= (
             const SFparse & inObj )
```

Copy assignement operator.

Parameters

in	*inObj*	object to be copied

Returns

SFparse object

4.1.3.7 operator=() [3/4]

```
SFparse & SFparse::operator= (
             SFparse && inObj )
```

Move assignement operator.

Parameters

in	*inObj*	object to be moved

Returns

SFparse object

4.1.3.8 operator=() [4/4]

```
SFparse& SFparse::operator= (
            SFparse && inObj )
```

Move assignement operator.

Parameters

in	*inObj*	object to be moved

Returns

SFparse object

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

- docs/sequence.hpp
- sequence.cpp

Chapter 5

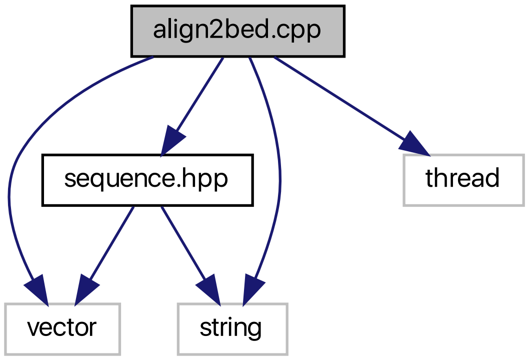
File Documentation

5.1 align2bed.cpp File Reference

Parsing DPGP.

```
#include "sequence.hpp"
#include <vector>
#include <string>
#include <thread>
```

Include dependency graph for align2bed.cpp:



Functions

- int **main** ()

5.1.1 Detailed Description

Parsing DPGP.

Author

Anthony J. Greenberg

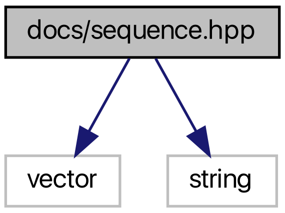
Extracting SNPs from the DPGP .seq files. The variant table will be in the *plink* BED format. Each chromosome is processed by its own thread in parallel.

5.2 docs/sequence.hpp File Reference

Sequence and SNP file parsing and conversion.

```
#include <vector>
#include <string>
```

Include dependency graph for sequence.hpp:



Classes

- class SFparse

  Sequence file parsing class.

5.2.1 Detailed Description

Sequence and SNP file parsing and conversion.

Author

Anthony J. Greenberg

Version

0.9

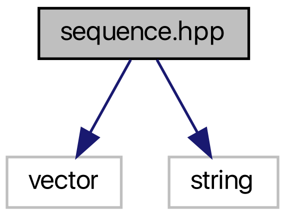
Class definitions and interface documentation for facilities that deal with common sequence and variant table file types.

5.3 sequence.hpp File Reference

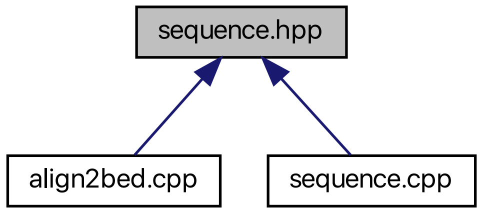
Sequence and SNP file parsing and conversion.

```
#include <vector>
#include <string>
```

Include dependency graph for sequence.hpp:



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



Classes

- class SFparse

    Sequence file parsing class.

5.3.1 Detailed Description

Sequence and SNP file parsing and conversion.

Author

Anthony J. Greenberg

Version

0.9

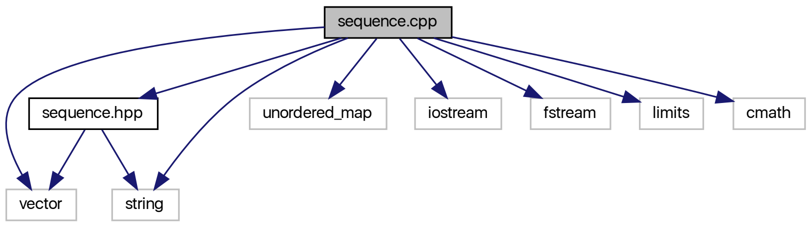
Class definitions and interface documentation for facilities that deal with common sequence and variant table file types.

5.4 sequence.cpp File Reference

Sequence and SNP file parsing and conversion.

```
#include "sequence.hpp"
#include <vector>
#include <unordered_map>
#include <string>
#include <iostream>
#include <fstream>
#include <limits>
#include <cmath>
```

Include dependency graph for sequence.cpp:



5.4.1 Detailed Description

Sequence and SNP file parsing and conversion.

Author

Anthony J. Greenberg

Version

0.9

Implementation of facilities that deal with common sequence and variant table file types.

Index