Sample SNPs

Generated by Doxygen 1.8.20

Chapter 1

Overview

This distribution consists of three parts. One, a program (*sampleSNPs*) that creates and saves ordered random samples of SNPs from a variety of formats. Two, a program (*sampleLD*) that calculates linkage disequilibrium (LD) among randomly chosen (without replacement) pairs of SNPs from a `binary variant format file`. Three, a library (*libsampFiles.a*) that allows users to build similar applications taking advantage of the fast sampling algorithms used in the two programs mentioned above.

1.1 Requirements

The software is build for Unix-like systems. Neither compilation nor running was checked under Windows and there are reasons to believe it will not compile on that OS. There are no dependencies other than a compiler that understands the C++11 standard. Random number generation employs the `RDRAND` CPU instruction if supported by the processor, otherwise an implementation of the 64-bit Mersenne Twister [1] is substituted. Intel Ivy Bridge or later support `RDRAND`. With AMD it is not completely clear. Opteron definitely does not support it. Zen architectures (Ryzen) claim to support it, but I did not have access to one so I cannot personally vouch for it. The RNG choice is made automatically at run time.

1.2 Installation

The simplest way to install everything is to run

```
make all
sudo make install
```

in the directory with the source code. This will install the executables (*sampleSNPs* and *sampleLD*) in _/usr/local/bin/_ and the library in the appropriate folders in _/usr/local/_. The included Makefile can be modified to change where things go. The headers *varfiles.hpp*, *populations.hpp*, and *random.hpp* have to be included in your code as necessary.

1.3 Testing

The *tests/* directory contains example .bed, .tped, .vcf, and .hmp.txt files to try running the programs on. To keep sizes manageable for distribution, the .bed file has 50,000 loci, while the text files have only 5,000. Make sure your samples do not exceed these values. Uncompress the directory and run, for example,

```
./sampleSNPs -i tests/sample -t BED -s 5000
```

This should sample 5,000 SNPs from the included *sample_ALL.bed* and *sample_ALL.bim* files and save the results into files with the *sample_ALL_s5000* prefix.

Note that *sampleLD* supports only the .bed format.

Running *sampleSNPs* and *sampleLD* without flags will cause these programs to print flag descriptions and exit.

1.4 Linkage disequilibrium estimates

`sampleLD` calculates LD between sampled pairs of loci. The paper cited below gives the details. The flag `-s` controls the number of locus pairs picked for estimates, and as this number increases, the average distance between pairs gets smaller. This leads to an increased precision of LD estimates between SNPs close to each other, at the expense of undersamling and hence diminished accuracy of disequilibrium calculations between distant loci. The user should keep this trade-off in mind when selecting the sample size. A further increase in LD estimate precision between distant variants can be achieved if one collects several sparse samples. However, care must be taken to eliminate redundant locus pairs because sampling without replacement is not guaranteed in this case.

An optional population index file can be provided after the `-p` flag. This file should contain space-delimited integer values that relate each individual in the .fam file to a population. An example file is in the archived *tests* directory.

1.5 Timing

Expanding the *timingTrials.tar.gz* archive will generate a directory with separate software, depending only on random.cpp and .hpp, that performs analyses of execution time using Vitter's Method D and Method S. The README.md file included there explains how to compile and run these analyses.

1.6 Citing this work

The paper describing this work is available, and can be referenced, from `bioRxiv` and `arXiv`.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

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 documented files with brief descriptions:

Chapter 5

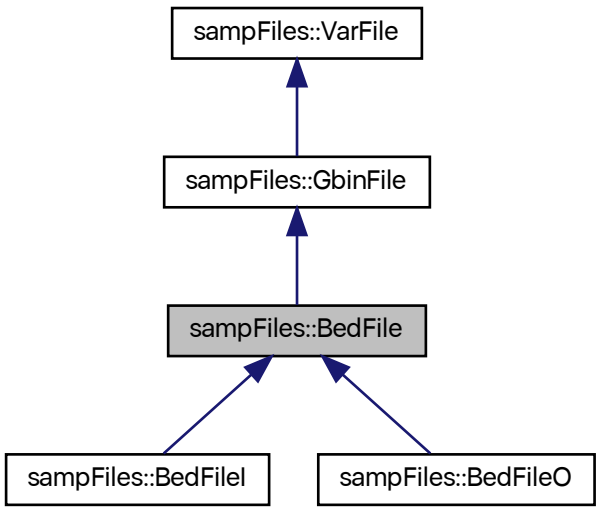
Class Documentation

5.1 sampFiles::BedFile Class Reference

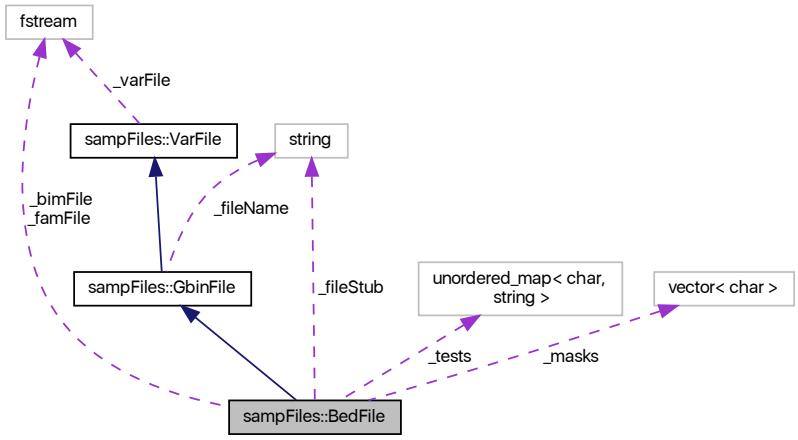
BED file base class.

`#include <varfiles.hpp>`

Inheritance diagram for sampFiles::BedFile:



Collaboration diagram for sampFiles::BedFile:



Public Member Functions

- BedFile ()

  Default constructor.
- BedFile (const string &stubName)

  File name constructor.
- BedFile (const BedFile &in)=default

  Copy constructor.
- BedFile & operator= (const BedFile &in)=default

  Copy assignment.
- BedFile (BedFile &&in)=default

  Move constructor.
- BedFile & operator= (BedFile &&in)=default

  Move assignment.
- ~BedFile ()

  Destructor.
- virtual void open ()

  Open stream (does nothing)
- void close ()

  Close stream.

Protected Attributes

- fstream _famFile

  Corresponding .fam file stream.
- fstream _bimFile

  Corresponding .bim file stream.
- string _fileStub

  File name stub (minus the extension)

Static Protected Attributes

- static const vector< char > _masks = {static_cast<char>(0x03), static_cast<char>(0x0C), static_cast<char>(0x30), static_cast<char>(0xC0)}

  Genotype bit masks.
- static const unordered_map< char, string > _tests

  Genotype bit tests.

Additional Inherited Members

5.1.1 Detailed Description

BED file base class.

Sets up streams for the auxiliary files.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 BedFile()

```
BedFile::BedFile (
            const string & stubName )
```

File name constructor.

Parameters

in	*stubName*	file name minus the extension

5.1.3 Member Data Documentation

5.1.3.1 _masks

```
const vector< char > BedFile::_masks = {static_cast<char>(0x03), static_cast<char>(0x0C), static_cast<char>(0x30), static_cast<char>(0xC0)}  [static], [protected]
```

Genotype bit masks.

Used to isolate each of the four genotypes (moving from the last bit pair) from the .bed two-bit genotype coding.

5.1.3.2 _tests

```
const unordered_map< char, string > BedFile::_tests  [static], [protected]
```

Initial value:

```
= {
    {'M', {static_cast<char>(0x01), static_cast<char>(0x04), static_cast<char>(0x10), static_cast<char>(0x40)}},
    {'H', {static_cast<char>(0x02), static_cast<char>(0x08), static_cast<char>(0x20), static_cast<char>(0x80)}}
}
```

Genotype bit tests.

Used to test each of the four genotypes (moving from the last bit pair) from the .bed two-bit genotype coding for the possible states.

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

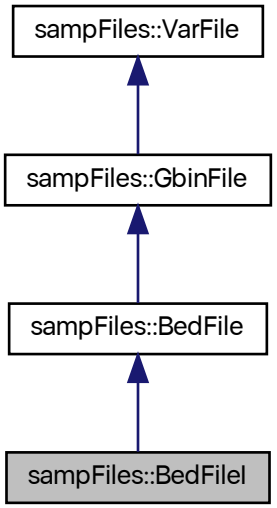
- varfiles.hpp
- varfiles.cpp

5.2 sampFiles::BedFileI Class Reference

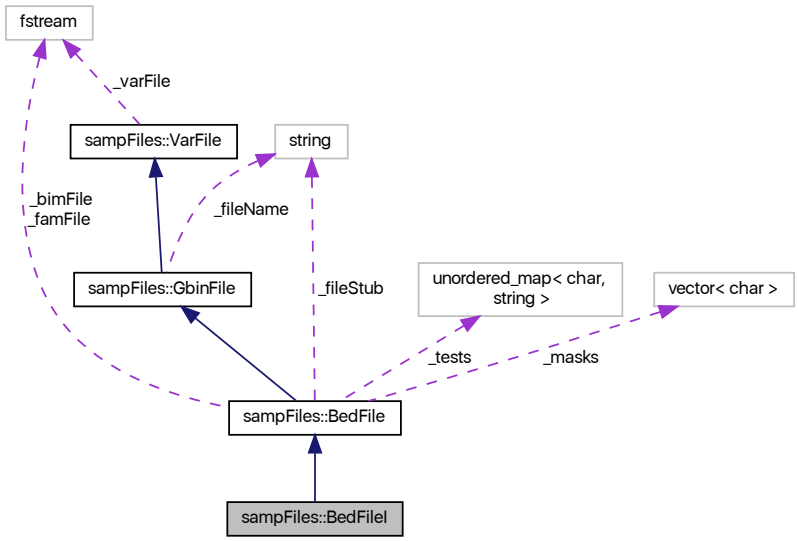
BED file input class.

```
#include <varfiles.hpp>
```

Inheritance diagram for sampFiles::BedFileI:



Collaboration diagram for sampFiles::BedFileI:



Public Member Functions

- BedFileI ()

  Default constructor.
- BedFileI (const string &stubName)

  File name constructor.
- BedFileI (const BedFileI &in)=default

  Copy constructor.
- BedFileI & operator= (const BedFileI &in)=default

  Copy assignment.
- BedFileI (BedFileI &&in)=default

  Move constructor.
- BedFileI & operator= (BedFileI &&in)=default

  Move assignment.
- ~BedFileI ()

  Destructor.
- void open ()

  Open stream to read.
- void sample (BedFileO &out, const uint64_t &n)

  Sample SNPs and save to BED file.
- void sampleLD (const uint64_t &n)

  Linkage disequilibrium among sampled sites.
- void sampleLD (const PopIndex &popID, const uint64_t &n)

  LD among sampled sites within populations.

- uint64_t nsnp ()

  Number of SNPs in the object.
- uint64_t nindiv ()

  Number of individuals in the object.

Protected Member Functions

- uint64_t _numLines ()

  Get number of lines in the `_bimFile`
- uint64_t _famLines ()

  Get number of lines in the `_famFile`
- uint64_t _famLines (fstream &fam)

  Copy the .fam file and count number of lines.
- void _ld (const char *snp1, const char *snp2, const size_t &N, const unsigned short &pad, double &rSq, double &Dprime, double &dcnt1, double &dcnt2)

  Between-SNP linkage disequilibrium (LD)
- void _ld (const char *snp1, const char *snp2, const PopIndex &popID, vector< double > &rSq, vector< double > &Dprime, vector< double > &dcnt1, vector< double > &dcnt2)

  Between-SNP LD within populations.

Additional Inherited Members

5.2.1 Detailed Description

BED file input class.

Reads BED files and the auxiliary files that come with them (.fam and .bim) as necessary. Only the SNP-major version is supported.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 BedFileI()

```
sampFiles::BedFileI::BedFileI (
            const string & stubName )  [inline]
```

File name constructor.

Parameters

in	*stubName*	file name minus the extension

5.2.3 Member Function Documentation

5.2.3.1 _famLines() [1/2]

```
uint64_t BedFileI::_famLines ( )  [protected]
```

Get number of lines in the _famFile

Assumes Unix-like line endings. The result is equal to the number of individuals.

Returns

number of lines in _famFile

5.2.3.2 _famLines() [2/2]

```
uint64_t BedFileI::_famLines (
            fstream & fam )  [protected]
```

Copy the .fam file and count number of lines.

Assumes Unix-like line endings. The result is equal to the number of individuals. The current object's .fam file is copied to the provided file stream, which should be open for raading. If not, the function throws a *string* object ``Output .fam filestream not open''.

Parameters

in	*fam*	.fam file stream

Returns

number of lines in _famFile

5.2.3.3 _ld() [1/2]

```
void BedFileI::_ld (
            const char * snp1,
            const char * snp2,
```

```
            const PopIndex & popID,
            vector< double > & rSq,
            vector< double > & Dprime,
            vector< double > & dcnt1,
            vector< double > & dcnt2 )  [protected]
```

Between-SNP LD within populations.

Calculates two LD statistics (r^2 and D') between two SNPs from a BED file. Missing values are ignored. If there are fewer than three haplotypes with data present at both loci, the return values are -9. This value is also returned if one of the loci is monomorphic after taking out missing data at the other SNP. Minor (not necessarily derived) allele counts are also reported to enable downstream filtering. Note that the populations are assumed diploid and the counts are of haploid chromosomes (i.e. one homozygote yields count of 2). The values are calculted within each population as indicated by the PopIndex object. The results are returned in the supplied vectors, which are assumed to be of correct size. Since this is an internal function unexposed to the user, this is not chaecked to save on compuation steps. Care must be taken that the char arrays passed to the function have lengths compatible with the number of individuals indexed by PopIndex. This is not checked.

Parameters

in	*snp1*	first SNP
in	*snp2*	second SNP
in	*popID*	population index
out	*rSq*	vector of r^2 estimates
out	*Dprime*	vector of D' estimates
out	*dcnt1*	vector of minor allele counts at locus 1
out	*dcnt2*	vector of minor allele counts at locus 2

5.2.3.4 _ld() [2/2]

```
void BedFileI::_ld (
            const char * snp1,
            const char * snp2,
            const size_t & N,
            const unsigned short & pad,
            double & rSq,
            double & Dprime,
            double & dcnt1,
            double & dcnt2 )  [protected]
```

Between-SNP linkage disequilibrium (LD)

Calculates two LD statistics (r^2 and D') between two SNPs from a BED file. Missing values are ignored. If there are fewer than three haplotypes with data present at both loci, the return values are -9. This value is also returned if one of the loci is monomorphic after taking out missing data at the other SNP. Minor (not necessarily derived) allele counts are also reported to enable downstream filtering. Note that the populations are assumed diploid and the counts are of haploid chromosomes (i.e. one homozygote yields count of 2).

Parameters

in	*snp1*	first SNP
in	*snp2*	second SNP
in	*N*	length of the genotype vector in bytes (four genotypes per byte)
in	*pad*	number of bit pairs of padding in the last byte
out	*rSq*	the r^2 estimate
out	*Dprime*	the D' estimate
out	*dcnt1*	minor allele count at locus 1
out	*dcnt2*	minor allele count at locus 2

5.2.3.5 _numLines()

```
uint64_t BedFileI::_numLines ( )  [protected]
```

Get number of lines in the `_bimFile`

Assumes Unix-like line endings. The result is equal to the number of SNPs.

Returns

number of lines in `_bimFile`

5.2.3.6 sample()

```
void BedFileI::sample (
            BedFileO & out,
            const uint64_t & n )
```

Sample SNPs and save to BED file.

Sample n SNPs without replacement from the file represented by the current object and save to the `out` object. Uses Vitter's [2] method. Number of samples has to be smaller that the number of SNPs in the file.

Parameters

in	*out*	output object
in	*n*	number of SNPs to sample

5.2.3.7 sampleLD() [1/2]

```
void BedFileI::sampleLD (
            const PopIndex & popID,
            const uint64_t & n )
```

LD among sampled sites within populations.

Samples sequential pairs of SNPs and calculates two LD measures (r^2 and D') within populations indicated by PopIndex. Saves to a file with the same name as the one preceding the .bed etc extensions, but adds _LD.tsv at the end. Each line is tab-delimited with the chromosome number (from the .bim file), between-SNP distance, non-reference allele count for each SNP, r^2, and D'. Missing data are ignored (only pairwise-complete observations are included). If one of the SNPs is monomorphic or if the total number of pairwise present genotypes is fewer than three (exclusive), the LD measures are returned as -9 to indicate missing values.

Parameters

in	*popID*	population index
in	*n*	number of SNP pairs to sample

5.2.3.8 sampleLD() [2/2]

```
void BedFileI::sampleLD (
            const uint64_t & n )
```

Linkage disequilibrium among sampled sites.

Samples sequential pairs of SNPs and calculates two LD measures (r^2 and D'). Saves to a file with the same name as the one preceding the .bed etc extensions, but adds _LD.tsv at the end. Each line is tab-delimited with the chromosome number (from the .bim file), between-SNP distance, non-reference allele count for each SNP, r^2, and D'. Missing data are ignored (only pairwise-complete observations are included). If one of the SNPs is monomorphic or if the total number of pairwise present genotypes is fewer than three (exclusive), the LD measures are returned as -9 to indicate missing values.

Parameters

in	*n*	number of SNP pairs to sample

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

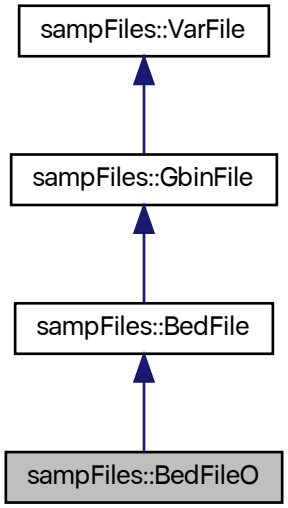
- varfiles.hpp
- varfiles.cpp

5.3 sampFiles::BedFileO Class Reference

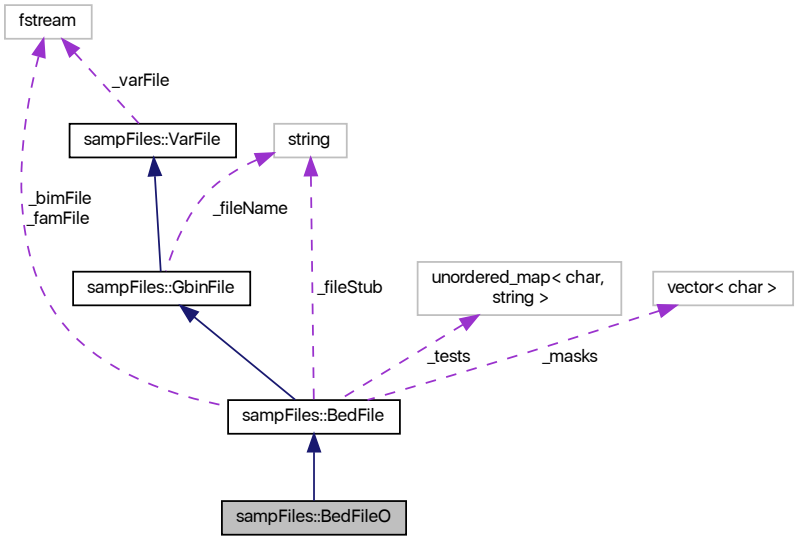
BED file output class.

```
#include <varfiles.hpp>
```

Inheritance diagram for sampFiles::BedFileO:



Collaboration diagram for sampFiles::BedFileO:



Public Member Functions

- BedFileO ()
  Default constructor.
- BedFileO (const string &stubName)
  File name constructor.
- BedFileO (const BedFileO &in)=default
  Copy constructor.
- BedFileO & operator= (const BedFileO &in)=default
  Copy assignment.
- BedFileO (BedFileO &&in)=default
  Move constructor.
- BedFileO & operator= (BedFileO &&in)=default
  Move assignment.
- ~BedFileO ()
  Destructor.
- void open ()
  Open stream to write.

Friends

- class **BedFileI**

Additional Inherited Members

5.3.1 Detailed Description

BED file output class.

Writes to BED files and the auxiliary files that come with them (.fam and .bim) as necessary. Data are written in the SNP-major format.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 BedFileO()

```
sampFiles::BedFileO::BedFileO (
            const string & stubName )  [inline]
```

File name constructor.

Parameters

in	*stubName*	file name minus the extension

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

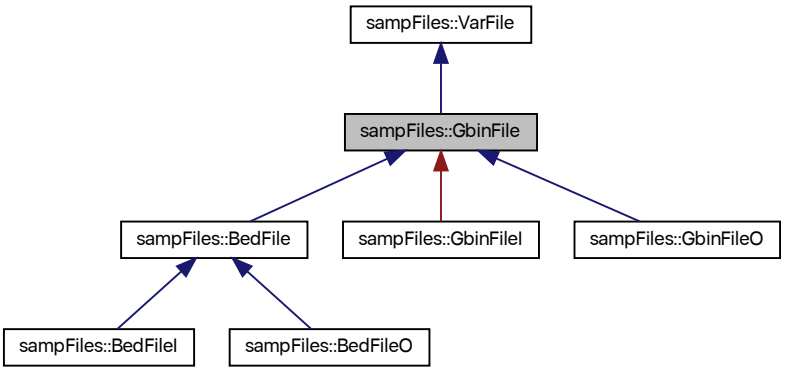
- varfiles.hpp
- varfiles.cpp

5.4 sampFiles::GbinFile Class Reference

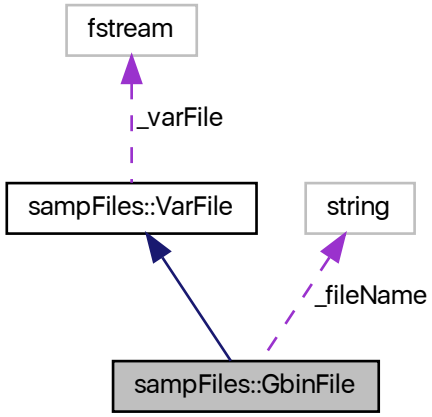
Generic binary file base class.

```
#include <varfiles.hpp>
```

Inheritance diagram for sampFiles::GbinFile:



Collaboration diagram for sampFiles::GbinFile:



Public Member Functions

- GbinFile ()

  Default constructor.
- GbinFile (const string &fileName, const size_t &nCols, const size_t &elemSize)

  Constructor with file name.
- GbinFile (const GbinFile &in)=default

  Copy constructor.
- GbinFile & operator= (const GbinFile &in)=default

  Copy assignment.
- GbinFile (GbinFile &&in)=default

  Move constructor.
- GbinFile & operator= (GbinFile &&in)=default

  Move assignment.
- ~GbinFile ()

  Destructor.
- virtual void open ()

  Open stream (does nothing)
- virtual void close ()

  Close stream.

Protected Attributes

- string _fileName

  File name.
- size_t _nCols

  Number of elements in a row.
- size_t _elemSize

  Size of each element in bytes.

Additional Inherited Members

5.4.1 Detailed Description

Generic binary file base class.

Sets up streams for binary files. No support for header lines.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 GbinFile()

```
sampFiles::GbinFile::GbinFile (
            const string & fileName,
            const size_t & nCols,
            const size_t & elemSize )  [inline]
```

Constructor with file name.

Throws ``Number of elements not divisible by the number of columns'' if the total number of elements is not divisible by the number of elements in a column.

Parameters

in	*fileName*	file name
in	*nCols*	number of columns, or elements in a row
in	*elemSize*	size of each element in bytes

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

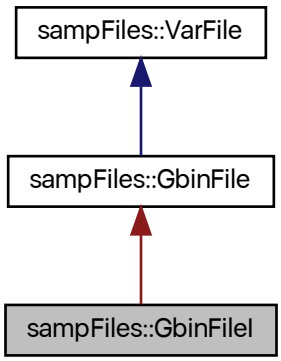
- varfiles.hpp
- varfiles.cpp

5.5 sampFiles::GbinFileI Class Reference

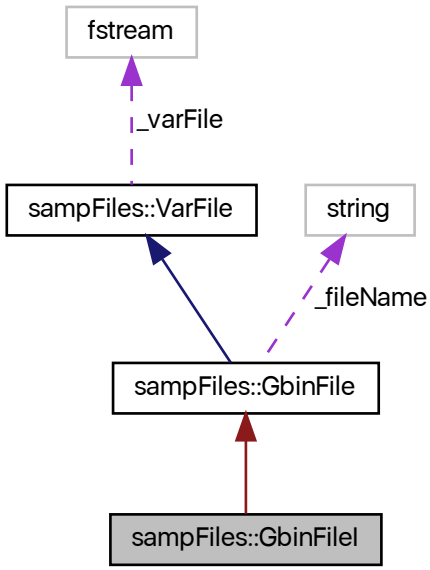
Binary file input class.

`#include <varfiles.hpp>`

Inheritance diagram for sampFiles::GbinFileI:



Collaboration diagram for sampFiles::GbinFileI:



Public Member Functions

- GbinFileI ()
  Default constructor.
- GbinFileI (const string &fileName, const size_t &nCols, const size_t &elemSize)
  File name constructor.
- GbinFileI (const GbinFileI &in)=default
  Copy constructor.
- GbinFileI & operator= (const GbinFileI &in)=default
  Copy assignment.
- GbinFileI (GbinFileI &&in)=default
  Move constructor.
- GbinFileI & operator= (GbinFileI &&in)=default
  Move assignment.
- ~GbinFileI ()
  Destructor.
- void open ()
  Open stream to read.
- void sample (GbinFileO &out, const uint64_t &n)
  Sample rows and save to a binary file.
- uint64_t nlines ()
  Number of rows in the object.

Protected Member Functions

- virtual uint64_t _numLines ()
  Get number of rows in the binary file.

5.5.1 Detailed Description

Binary file input class.

Reads binary files.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 GbinFileI()

```
sampFiles::GbinFileI::GbinFileI (
            const string & fileName,
            const size_t & nCols,
            const size_t & elemSize )  [inline]
```

File name constructor.

Parameters

in	*fileName*	file name including extension
in	*nCols*	number of columns, or elements in a row
in	*elemSize*	size of each element in bytes

5.5.3 Member Function Documentation

5.5.3.1 _numLines()

```
uint64_t GbinFileI::_numLines ( )  [protected], [virtual]
```

Get number of rows in the binary file.

Requires knowledge of the number of elements in a row and their size in bytes. Throws a *string* object ``Number of elements not divisible by row size'' if the total number of elements in the file is not divisible by the product of the number of columns and element size.

Returns

number of rows

5.5.3.2 sample()

```
void GbinFileI::sample (
            GbinFileO & out,
            const uint64_t & n )
```

Sample rows and save to a binary file.

Sample n lines without replacement from the file represented by the current object and save to the `out` object. Uses Vitter's [2] method. Number of samples has to be smaller that the number of rows in the file.

Parameters

in	*out*	output object
in	*n*	number of rows to sample

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

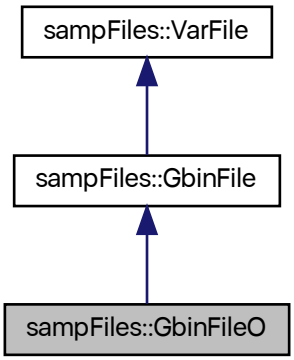
- varfiles.hpp
- varfiles.cpp

5.6 sampFiles::GbinFileO Class Reference

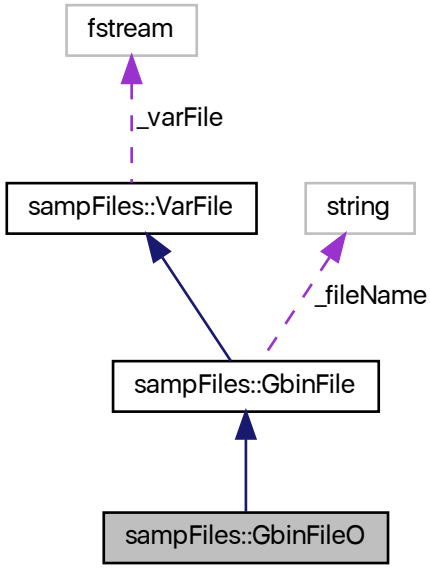
Generic binary file output class.

```
#include <varfiles.hpp>
```

Inheritance diagram for sampFiles::GbinFileO:



Collaboration diagram for sampFiles::GbinFileO:



Public Member Functions

- GbinFileO ()
  Default constructor.
- GbinFileO (const string &fileName, const size_t &nCols, const size_t &elemSize)
  File name constructor.
- GbinFileO (const GbinFileO &in)=default
  Copy constructor.
- GbinFileO & operator= (const GbinFileO &in)=default
  Copy assignment.
- GbinFileO (GbinFileO &&in)=default
  Move constructor.
- GbinFileO & operator= (GbinFileO &&in)=default
  Move assignment.
- ~GbinFileO ()
  Destructor.
- void open ()
  Open stream to write.

Friends

- class **GbinFileI**

Additional Inherited Members

5.6.1 Detailed Description

Generic binary file output class.

Writes binary files.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 GbinFileO()

```
sampFiles::GbinFileO::GbinFileO (
            const string & fileName,
            const size_t & nCols,
            const size_t & elemSize )  [inline]
```

File name constructor.

Parameters

in	*fileName*	file name including extension
in	*nCols*	number of columns, or elements in a row
in	*elemSize*	size of each element in bytes

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

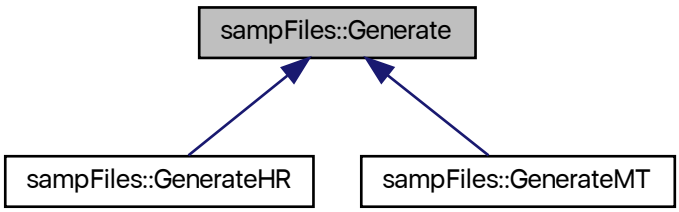
- varfiles.hpp
- varfiles.cpp

5.7 sampFiles::Generate Class Reference

Abstract base random number class.

```
#include <random.hpp>
```

Inheritance diagram for sampFiles::Generate:



Public Member Functions

- virtual ~Generate ()

  Protected destructor.
- virtual volatile uint64_t ranInt ()=0

  Generate a (pseudo-)random 64-bit unsigned integer.

Protected Member Functions

- Generate ()

  Protected default constructor.
- Generate (const Generate &old)

  Protected copy constructor.
- Generate (Generate &&old)

  Protected move constructor.
- Generate & operator= (const Generate &old)=default

  Protected copy assignment operator.
- Generate & operator= (Generate &&old)=default

  Protected move assignment.

5.7.1 Detailed Description

Abstract base random number class.

Provides the interface for random or pseudorandom (depending on derived class) generation. For internal use by the `RanDraw` interface class.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Generate() [1/2]

```
sampFiles::Generate::Generate (
            const Generate & old )  [inline], [protected]
```

Protected copy constructor.

Parameters

in	*old*	object to copy

5.7.2.2 Generate() [2/2]

```
sampFiles::Generate::Generate (
            Generate && old )  [inline], [protected]
```

Protected move constructor.

Parameters

in	*old*	object to move

5.7.3 Member Function Documentation

5.7.3.1 operator=() [1/2]

```
Generate& sampFiles::Generate::operator= (
            const Generate & old )  [protected], [default]
```

Protected copy assignment operator.

Parameters

in	*old*	object to copy

5.7.3.2 operator=() [2/2]

```
Generate& sampFiles::Generate::operator= (
            Generate && old )  [protected], [default]
```

Protected move assignment.

Parameters

in	*old*	object to move

5.7.3.3 ranInt()

```
virtual volatile uint64_t sampFiles::Generate::ranInt ( )  [pure virtual]
```

Generate a (pseudo-)random 64-bit unsigned integer.

Returns

random or pseudo-random 64-bit unsigned integer

Implemented in sampFiles::GenerateMT, and sampFiles::GenerateHR.

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

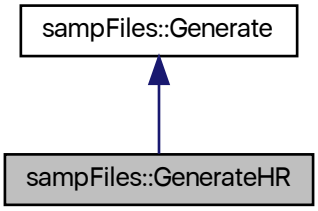
- random.hpp

5.8 sampFiles::GenerateHR Class Reference

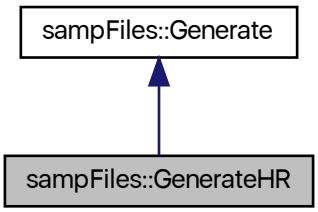
Hardware random number generating class.

`#include <random.hpp>`

Inheritance diagram for sampFiles::GenerateHR:



Collaboration diagram for sampFiles::GenerateHR:



Public Member Functions

- GenerateHR ()
  Default constructor.
- ~GenerateHR ()
  Destructor.
- GenerateHR (const GenerateHR &old)
  Copy constructor.
- GenerateHR (GenerateHR &&old)
  Move constructor.
- GenerateHR & operator= (const GenerateHR &old)=default
  Copy assignment operator.
- GenerateHR & operator= (GenerateHR &&old)=default
  Move assignment.
- volatile uint64_t ranInt ()
  Generate a random 64-bit unsigned integer.

Additional Inherited Members

5.8.1 Detailed Description

Hardware random number generating class.

Generates random deviates from a number of distributions, using hardware random numbers (*RDRAND* processor instruction). Health of the RDRAND generator is tested every time a new number is required. Throws a `string` object "RDRAND_failed" if the test fails. The implementation of random 64-bit integer generation follows `Intel's suggestions`.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 GenerateHR() [1/2]

```
sampFiles::GenerateHR::GenerateHR (
            const GenerateHR & old )  [inline]
```

Copy constructor.

Parameters

in	*old*	object to copy

5.8.2.2 GenerateHR() [2/2]

```
sampFiles::GenerateHR::GenerateHR (
            GenerateHR && old )  [inline]
```

Move constructor.

Parameters

in	*old*	object to move

5.8.3 Member Function Documentation

5.8.3.1 operator=() [1/2]

```
GenerateHR& sampFiles::GenerateHR::operator= (
            const GenerateHR & old )  [default]
```

Copy assignment operator.

Parameters

in	*old*	object to copy

5.8.3.2 operator=() [2/2]

```
GenerateHR& sampFiles::GenerateHR::operator= (
            GenerateHR && old )  [default]
```

Move assignment.

Parameters

in	*old*	object to move

5.8.3.3 ranInt()

```
volatile uint64_t GenerateHR::ranInt ( )  [virtual]
```

Generate a random 64-bit unsigned integer.

Monitors the health of the CPU random number generator and throws a `string` object "RDRAND_failed" if a failure is detected after ten tries.

Returns

digital random 64-bit unsigned integer

Implements sampFiles::Generate.

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

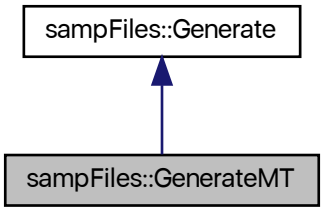
- random.hpp
- random.cpp

5.9 sampFiles::GenerateMT Class Reference

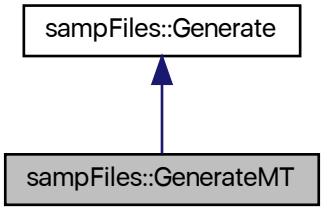
Pseudo-random number generator.

```
#include <random.hpp>
```

Inheritance diagram for sampFiles::GenerateMT:



Collaboration diagram for sampFiles::GenerateMT:



Public Member Functions

- GenerateMT ()

  Default constructor.
- ~GenerateMT ()

  Protected destructor.
- GenerateMT (const GenerateMT &old)=default

  Copy constructor.
- GenerateMT (GenerateMT &&old)=default

  Move constructor.
- GenerateMT & operator= (const GenerateMT &old)=default

  Copy assignment operator.
- GenerateMT & operator= (GenerateMT &&old)=default

  Move assignment.
- volatile uint64_t ranInt ()

  Generate a pseudo-random 64-bit unsigned integer.

Protected Attributes

- uint64_t _mt [312]

  Generator state array.
- size_t _mti

  State of the array index.
- uint64_t _x

  Current state.

Static Protected Attributes

- static const unsigned short _n = 312

  Degree of recurrence.
- static const unsigned short _m = 156

  Middle word.
- static const uint64_t _um = static_cast<uint64_t>(0x7FFFFFFF)

  Most significant 33 bits.
- static const uint64_t _lm = static_cast<uint64_t>(0xFFFFFFFF80000000)

  Least significant 31 bits.
- static const uint64_t _b = static_cast<uint64_t>(0x71D67FFFEDA60000)

  Tempering bitmask.
- static const uint64_t _c = static_cast<uint64_t>(0xFFF7EEE000000000)

  Tempering bitmask.
- static const uint64_t _d = static_cast<uint64_t>(0x5555555555555555)

  Tempering bitmask.
- static const unsigned int _l = 43

  Tempering shift.
- static const unsigned int _s = 17

  Tempering shift.
- static const unsigned int _t = 37

  Tempering shift.
- static const unsigned int _u = 29

  Tempering shift.
- static const uint64_t _alt [2] = {static_cast<uint64_t>(0), static_cast<uint64_t>(0xB5026F5AA96619E9)}

  Array of alternative values for the twist.

Additional Inherited Members

5.9.1 Detailed Description

Pseudo-random number generator.

An implementaiton of the 64-bit MT19937 ("Mersenne Twister") [1] pseudo-random number generator (PRNG). The constructor automatically seeds the PRNG. The implementation was guided by the reference code posted by the authors.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 GenerateMT() [1/3]

```
GenerateMT::GenerateMT ( )
```

Default constructor.

Seeds the PRNG with a call to the *RDTSC* instruction.

5.9.2.2 GenerateMT() [2/3]

```
sampFiles::GenerateMT::GenerateMT (
            const GenerateMT & old )  [default]
```

Copy constructor.

Parameters

in	*old*	object to copy

5.9.2.3 GenerateMT() [3/3]

```
sampFiles::GenerateMT::GenerateMT (
            GenerateMT && old )  [default]
```

Move constructor.

Parameters

in	*old*	object to move

5.9.3 Member Function Documentation

5.9.3.1 operator=() [1/2]

```
GenerateMT& sampFiles::GenerateMT::operator= (
            const GenerateMT & old )  [default]
```

Copy assignment operator.

Parameters

in	*old*	object to copy

5.9.3.2 operator=() [2/2]

```
GenerateMT& sampFiles::GenerateMT::operator= (
            GenerateMT && old )  [default]
```

Move assignment.

Parameters

in	*old*	object to move

5.9.3.3 ranInt()

```
volatile uint64_t GenerateMT::ranInt ( )  [virtual]
```

Generate a pseudo-random 64-bit unsigned integer.

Returns

pseudo-random 64-bit unsigned integer

Implements sampFiles::Generate.

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

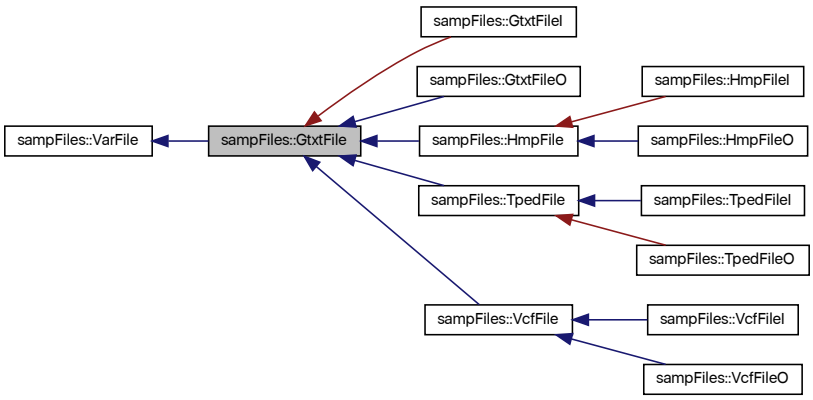
- random.hpp
- random.cpp

5.10 sampFiles::GtxtFile Class Reference

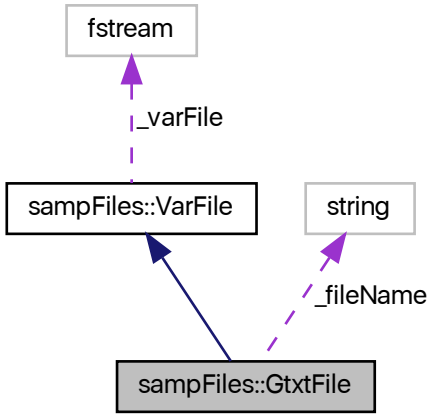
Generic text file base class.

`#include <varfiles.hpp>`

Inheritance diagram for sampFiles::GtxtFile:



Collaboration diagram for sampFiles::GtxtFile:



Public Member Functions

- GtxtFile ()
  Default constructor.
- GtxtFile (const string &fileName)
  Constructor with file name.
- GtxtFile (const string &fileName, const bool &head)
  Constructor with file name and header indicator.
- GtxtFile (const GtxtFile &in)=default
  Copy constructor.
- GtxtFile & operator= (const GtxtFile &in)=default
  Copy assignment.
- GtxtFile (GtxtFile &&in)=default
  Move constructor.
- GtxtFile & operator= (GtxtFile &&in)=default
  Move assignment.
- ~GtxtFile ()
  Destructor.
- virtual void open ()
  Open stream (does nothing)
- virtual void close ()
  Close stream.

Protected Attributes

- string _fileName
  File name.
- bool _head
  Is there a header?

Additional Inherited Members

5.10.1 Detailed Description

Generic text file base class.

Sets up streams for text files. If specified, the first line can be considered a header and treated separately.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 GtxtFile() [1/2]

```
sampFiles::GtxtFile::GtxtFile (
            const string & fileName )  [inline]
```

Constructor with file name.

Parameters

in	*fileName*	file name

5.10.2.2 GtxtFile() [2/2]

```
sampFiles::GtxtFile::GtxtFile (
            const string & fileName,
            const bool & head )  [inline]
```

Constructor with file name and header indicator.

Parameters

in	*fileName*	file name
in	*head*	header presence

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

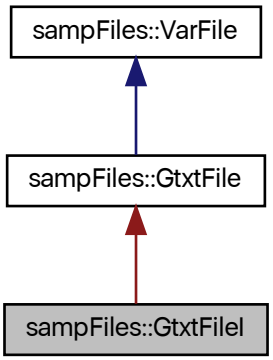
- varfiles.hpp
- varfiles.cpp

5.11 sampFiles::GtxtFileI Class Reference

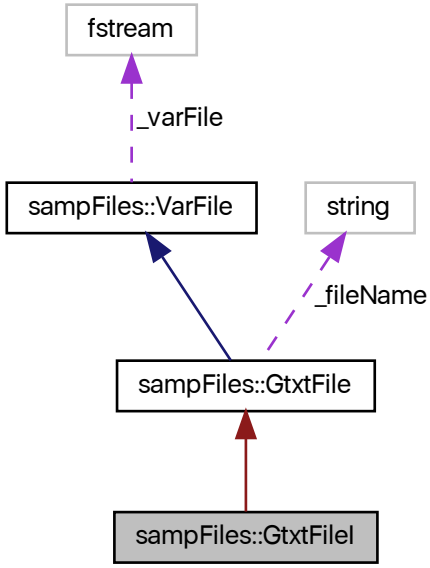
Text file input class.

`#include <varfiles.hpp>`

Inheritance diagram for sampFiles::GtxtFileI:



Collaboration diagram for sampFiles::GtxtFileI:



Public Member Functions

- GtxtFileI ()
  Default constructor.
- GtxtFileI (const string &fileName)
  File name constructor with header specification.
- GtxtFileI (const string &fileName, const bool &head)
  File name constructor with header specification.
- GtxtFileI (const GtxtFileI &in)=default
  Copy constructor.
- GtxtFileI & operator= (const GtxtFileI &in)=default
  Copy assignment.
- GtxtFileI (GtxtFileI &&in)=default
  Move constructor.
- GtxtFileI & operator= (GtxtFileI &&in)=default
  Move assignment.
- ~GtxtFileI ()
  Destructor.
- void open ()
  Open stream to read.
- void sample (GtxtFileO &out, const uint64_t &n, const bool &headSkip)

Sample rows and save to a text file.

- void sample (const uint64_t &n, const bool &headSkip, const char &delim, vector< string > &out)

  Sample rows and save export to a vector of strings.

- uint64_t nlines ()

  Number of SNPs in the object.

Protected Member Functions

- virtual uint64_t _numLines ()

  Get number of rows in the text file.

5.11.1 Detailed Description

Text file input class.

Reads text files, skipping or copying the header as necessary.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 GtxtFileI() [1/2]

```
sampFiles::GtxtFileI::GtxtFileI (
            const string & fileName )  [inline]
```

File name constructor with header specification.

Parameters

in	*fileName*	file name including extension

5.11.2.2 GtxtFileI() [2/2]

```
sampFiles::GtxtFileI::GtxtFileI (
            const string & fileName,
            const bool & head )  [inline]
```

File name constructor with header specification.

Parameters

in	*fileName*	file name including extension
in	*head*	header presence

5.11.3 Member Function Documentation

5.11.3.1 _numLines()

```
uint64_t GtxtFileI::_numLines ( )  [protected], [virtual]
```

Get number of rows in the text file.

Assumes Unix-like line endings. Header, if present, is not counted. Is overriden in some, but not all, derived classes.

Returns

number of rows

5.11.3.2 sample() [1/2]

```
void GtxtFileI::sample (
            const uint64_t & n,
            const bool & headSkip,
            const char & delim,
            vector< string > & out )
```

Sample rows and save export to a vector of strings.

Sample n rows without replacement from the file represented by the current object and output a vector of strings. Each field separated by the specified delimiter is stored as an element of the vector. Uses Vitter's [2] method. Number of samples has to be smaller that the number of rows in the file. The output vector is erased if it is not empty.

Parameters

in	*n*	number of SNPs to sample
in	*headSkip*	skip header? Ignored if there is no header
in	*delim*	field delimiter
out	*out*	output vector

5.11.3.3 sample() [2/2]

```
void GtxtFileI::sample (
            GtxtFileO & out,
            const uint64_t & n,
            const bool & headSkip )
```

Sample rows and save to a text file.

Sample n lines without replacement from the file represented by the current object and save to the `out` object. Uses Vitter's [2] method. Number of samples has to be smaller that the number of rows in the file.

Parameters

in	*out*	output object
in	*n*	number of rows to sample
in	*headSkip*	skip header? Ignored if there is no header

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

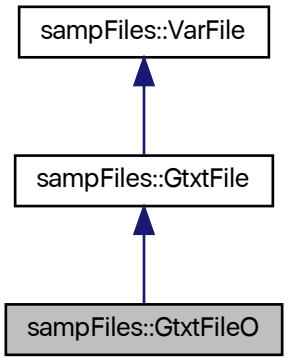
- varfiles.hpp
- varfiles.cpp

5.12 sampFiles::GtxtFileO Class Reference

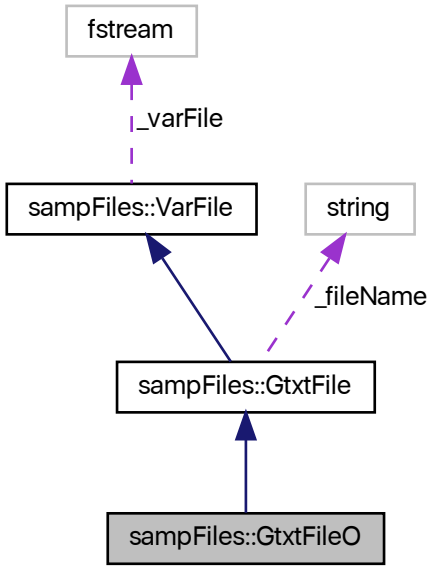
Generic text file output class.

`#include <varfiles.hpp>`

Inheritance diagram for sampFiles::GtxtFileO:



Collaboration diagram for sampFiles::GtxtFileO:



Public Member Functions

- GtxtFileO ()

  Default constructor.
- GtxtFileO (const string &fileName)

  File name constructor.
- GtxtFileO (const string &fileName, const bool &head)

  File name constructor with header specification.
- GtxtFileO (const GtxtFileO &in)=default

  Copy constructor.
- GtxtFileO & operator= (const GtxtFileO &in)=default

  Copy assignment.
- GtxtFileO (GtxtFileO &&in)=default

  Move constructor.
- GtxtFileO & operator= (GtxtFileO &&in)=default

  Move assignment.
- ~GtxtFileO ()

  Destructor.
- void open ()

  Open stream to write.

Friends

- class **GtxtFileI**

Additional Inherited Members

5.12.1 Detailed Description

Generic text file output class.

Writes text files.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 GtxtFileO() [1/2]

```
sampFiles::GtxtFileO::GtxtFileO (
            const string & fileName )  [inline]
```

File name constructor.

Parameters

in	*fileName*	file name including the extension

5.12.2.2 GtxtFileO() [2/2]

```
sampFiles::GtxtFileO::GtxtFileO (
            const string & fileName,
            const bool & head )  [inline]
```

File name constructor with header specification.

Parameters

in	*fileName*	file name including the extension
in	*head*	header presence

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

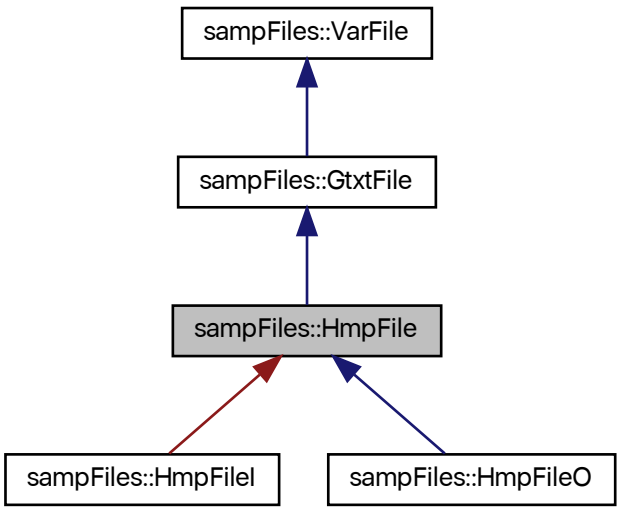
- varfiles.hpp
- varfiles.cpp

5.13 sampFiles::HmpFile Class Reference

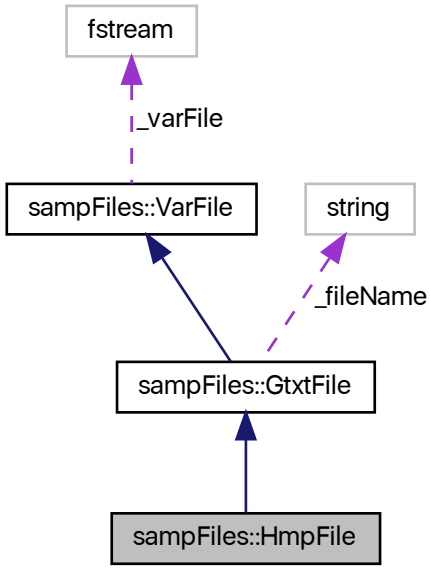
Hapmap (HMP) file base class.

```
#include <varfiles.hpp>
```

Inheritance diagram for sampFiles::HmpFile:



Collaboration diagram for sampFiles::HmpFile:



Public Member Functions

- HmpFile ()

  Default constructor.
- HmpFile (const string &fileName)

  Constructor with file name.
- HmpFile (const HmpFile &in)=default

  Copy constructor.
- HmpFile & operator= (const HmpFile &in)=default

  Copy assignment.
- HmpFile (HmpFile &&in)=default

  Move constructor.
- HmpFile & operator= (HmpFile &&in)=default

  Move assignment.
- ~HmpFile ()

  Destructor.
- virtual void open ()

  Open stream (does nothing)
- virtual void close ()

  Close stream.

Additional Inherited Members

5.13.1 Detailed Description

Hapmap (HMP) file base class.

Sets up streams for HMP files.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 HmpFile()

```
sampFiles::HmpFile::HmpFile (
            const string & fileName )  [inline]
```

Constructor with file name.

Parameters

in	*fileName*	file name

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

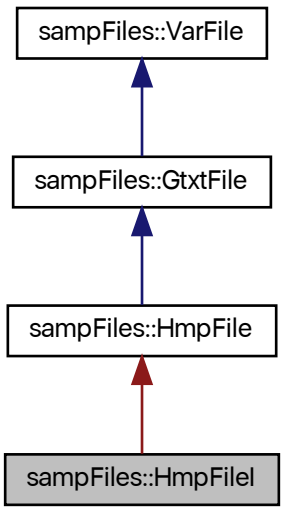
- varfiles.hpp
- varfiles.cpp

5.14 sampFiles::HmpFileI Class Reference

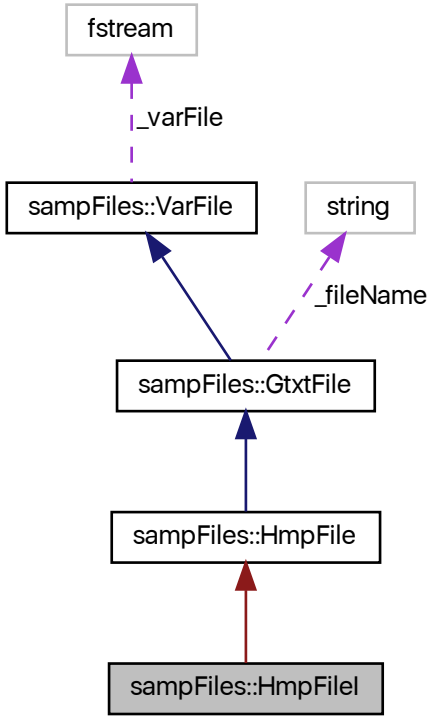
HMP file input class.

```
#include <varfiles.hpp>
```

Inheritance diagram for sampFiles::HmpFileI:



Collaboration diagram for sampFiles::HmpFileI:



Public Member Functions

- HmpFileI ()
  Default constructor.
- HmpFileI (const string &fileName)
  File name constructor.
- HmpFileI (const HmpFileI &in)=default
  Copy constructor.
- HmpFileI & operator= (const HmpFileI &in)=default
  Copy assignment.
- HmpFileI (HmpFileI &&in)=default
  Move constructor.
- HmpFileI & operator= (HmpFileI &&in)=default
  Move assignment.
- ~HmpFileI ()
  Destructor.
- void open ()

Open stream to read.

- void sample (HmpFileO &out, const uint64_t &n)

  Sample SNPs and save to HMP file.

- uint64_t nsnp ()

  Number of SNPs in the object.

Protected Member Functions

- uint64_t _numLines ()

  Get number of SNPs in the HMP file.

5.14.1 Detailed Description

HMP file input class.

Reads HMP files, skipping or copying the header as necessary.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 HmpFileI()

```
HmpFileI::HmpFileI (
            const string & fileName )
```

File name constructor.

Parameters

in	*fileName*	file name including extension

5.14.3 Member Function Documentation

5.14.3.1 _numLines()

```
uint64_t HmpFileI::_numLines ( )  [protected]
```

Get number of SNPs in the HMP file.

Assumes Unix-like line endings. Header is not counted.

Returns

number of SNPs

5.14.3.2 sample()

```
void HmpFileI::sample (
            HmpFileO & out,
            const uint64_t & n )
```

Sample SNPs and save to HMP file.

Sample n SNPs without replacement from the file represented by the current object and save to the `out` object. Uses Vitter's [2] method. Number of samples has to be smaller that the number of SNPs in the file.

Parameters

in	*out*	output object
in	*n*	number of SNPs to sample

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

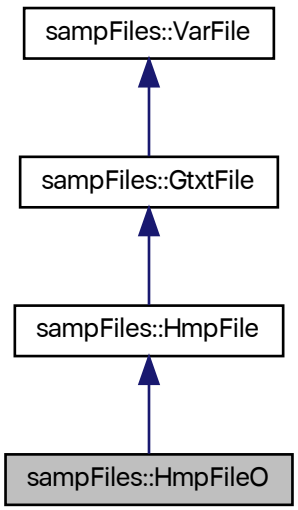
- varfiles.hpp
- varfiles.cpp

5.15 sampFiles::HmpFileO Class Reference

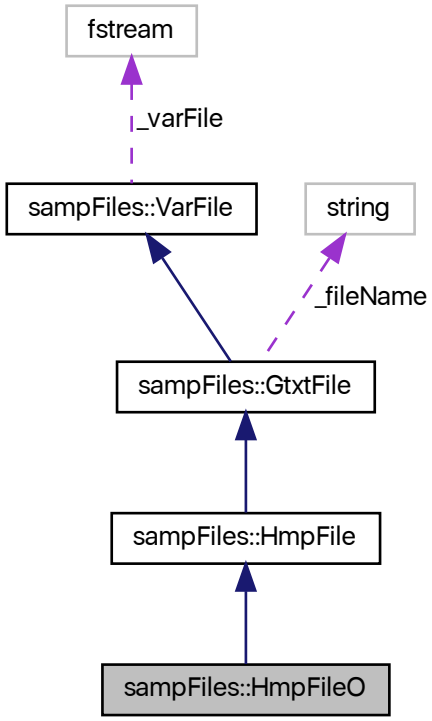
HMP file output class.

```
#include <varfiles.hpp>
```

Inheritance diagram for sampFiles::HmpFileO:



Collaboration diagram for sampFiles::HmpFileO:



Public Member Functions

- HmpFileO ()
  Default constructor.
- HmpFileO (const string &fileName)
  File name constructor.
- HmpFileO (const HmpFileO &in)=default
  Copy constructor.
- HmpFileO & operator= (const HmpFileO &in)=default
  Copy assignment.
- HmpFileO (HmpFileO &&in)=default
  Move constructor.
- HmpFileO & operator= (HmpFileO &&in)=default
  Move assignment.
- ~HmpFileO ()
  Destructor.
- void open ()
  Open stream to write.

Friends

- class **HmpFileI**

Additional Inherited Members

5.15.1 Detailed Description

HMP file output class.

Writes HMP files.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 HmpFileO()

```
sampFiles::HmpFileO::HmpFileO (
            const string & fileName )  [inline]
```

File name constructor.

Parameters

in	*fileName*	file name including the extension

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

- varfiles.hpp
- varfiles.cpp

5.16 sampFiles::PopIndex Class Reference

Population index.

```
#include <populations.hpp>
```

Public Member Functions

- PopIndex ()
  Default constructor.
- PopIndex (const int *arr, const size_t &N)
  Array constructor.
- PopIndex (const string &inFileName)
  File read constructor.
- vector< size_t > & operator[ ] (const size_t &i)
  Vector subscript operator.
- const vector< size_t > & operator[ ] (const size_t &i) const
  `const` *vector subscript operator*
- size_t popSize (const size_t &i)
  Population size.
- size_t popSize (const size_t &i) const
  `const` *population size*
- size_t size ()
  Total sample size.
- size_t size () const
  `const` *total sample size*
- size_t popNumber ()
  Number of populations.
- size_t popNumber () const
  `const` *number of populations*

5.16.1 Detailed Description

Population index.

For each population, contains indexes of the lines that belong to it.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 PopIndex() [1/2]

```
PopIndex::PopIndex (
            const int * arr,
            const size_t & N )
```

Array constructor.

The input array has an element for each line, and the value of that element is the population ID in the form of an *int* that is base-1 (i.e., if line N is in the first population, then `arr[N] == 1`).

Parameters

in	*arr*	array of population IDs
in	*N*	array length

5.16.2.2 PopIndex() [2/2]

```
PopIndex::PopIndex (
            const string & inFileName )
```

File read constructor.

The input file has an entry for each line (separated by white space), and the value of that entry is the population ID in the form of an *int* that is base-1 (i.e., if line N is in the first population, then `arr[N] == 1`).

Parameters

in	*inFileName*	input file name

5.16.3 Member Function Documentation

5.16.3.1 operator[]() [1/2]

```
vector<size_t>& sampFiles::PopIndex::operator[] (
            const size_t & i )  [inline]
```

Vector subscript operator.

Returns the index of population *i*.

Parameters

in	*i*	population index

Returns

index of line IDs

5.16.3.2 operator[]() [2/2]

```
const vector<size_t>& sampFiles::PopIndex::operator[] (
            const size_t & i ) const  [inline]
```

const vector subscript operator

Returns the index of population *i*.

Parameters

in	*i*	population index

Returns

index of line IDs

5.16.3.3 popNumber() [1/2]

```
size_t sampFiles::PopIndex::popNumber ( )  [inline]
```

Number of populations.

Returns

number of populations

5.16.3.4 popNumber() [2/2]

```
size_t sampFiles::PopIndex::popNumber ( ) const  [inline]
```

const number of populations

Returns

number of populations

5.16.3.5 popSize() [1/2]

```
size_t sampFiles::PopIndex::popSize (
            const size_t & i )  [inline]
```

Population size.

Parameters

in	*i*	population index

Returns

size of the _i_th population

5.16.3.6 popSize() [2/2]

```
size_t sampFiles::PopIndex::popSize (
            const size_t & i ) const  [inline]
```

`const` population size

Parameters

in	*i*	population index

Returns

size of the _i_th population

5.16.3.7 size() [1/2]

```
size_t sampFiles::PopIndex::size ( )  [inline]
```

Total sample size.

Returns

total sample size

5.16.3.8 size() [2/2]

```
size_t sampFiles::PopIndex::size ( ) const  [inline]
```

`const` total sample size

Returns

total sample size

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

- populations.hpp
- populations.cpp

5.17 sampFiles::RanDraw Class Reference

Random number generating class.

`#include <random.hpp>`

Public Member Functions

- RanDraw ()
  Default constructor.
- ~RanDraw ()
  Destructor.
- RanDraw (const RanDraw &old)=default
  Copy constructor.
- RanDraw (RanDraw &&old)=default
  Move constructor.
- RanDraw & operator= (const RanDraw &old)=default
  Copy assignment.
- RanDraw & operator= (RanDraw &&old)=default
  Move assignment.
- volatile uint64_t ranInt ()
  Generate random integer.
- volatile double runif ()
  Generate a uniform deviate.
- volatile double runifnz ()
  Generate a non-zero uniform deviate.
- volatile uint64_t vitterA (const double &n, const double &N)
  Sample from Vitter's distribution, method A.
- volatile uint64_t vitter (const double &n, const double &N)
  Sample from Vitter's distribution, method D.

5.17.1 Detailed Description

Random number generating class.

Generates (pseudo-)random deviates from a number of distributions. If hardware random numbers are supported, uses them. Otherwise, falls back to 64-bit MT19937 ("Mersenne Twister").

5.17.2 Constructor & Destructor Documentation

5.17.2.1 RanDraw() [1/3]

```
RanDraw::RanDraw ( )
```

Default constructor.

Checks if the processor provides hardware random number support. Seeds the Mersenne Twister if not. Throws "CPU_unsupported" string object if the CPU is not AMD or Intel.

5.17.2.2 RanDraw() [2/3]

```
sampFiles::RanDraw::RanDraw (
            const RanDraw & old )  [default]
```

Copy constructor.

Parameters

in	*old*	pbject to be copied

5.17.2.3 RanDraw() [3/3]

```
sampFiles::RanDraw::RanDraw (
            RanDraw && old )  [default]
```

Move constructor.

Parameters

in	*old*	pbject to be moved

5.17.3 Member Function Documentation

5.17.3.1 operator=() [1/2]

```
RanDraw& sampFiles::RanDraw::operator= (
            const RanDraw & old )  [default]
```

Copy assignment.

Parameters

in	*old*	pbject to be copied

5.17.3.2 operator=() [2/2]

```
RanDraw& sampFiles::RanDraw::operator= (
            RanDraw && old )  [default]
```

Move assignment.

Parameters

in	*old*	pbject to be moved

5.17.3.3 ranInt()

```
volatile uint64_t sampFiles::RanDraw::ranInt ( )  [inline]
```

Generate random integer.

Returns

An unsigned random 64-bit integer

5.17.3.4 runif()

```
volatile double sampFiles::RanDraw::runif ( )  [inline]
```

Generate a uniform deviate.

Returns

A double-precision value from the $U[0, 1]$ distribution

5.17.3.5 runifnz()

```
volatile double RanDraw::runifnz ( )
```

Generate a non-zero uniform deviate.

Returns

A double-precision value from the $U(0, 1]$ distribution

5.17.3.6 vitter()

```
volatile uint64_t RanDraw::vitter (
            const double & n,
            const double & N )
```

Sample from Vitter's distribution, method D.

Given the number of remaining records in a file N and the number of records n remaining to be selected, sample the number of records to skip over. This function implements Vitter's [3] [2] method D. It is useful for online one-pass sampling of records from a file. While the inputs are integer, we pass them in as *double* because that is more efficient for calculations.

Parameters

in	*n*	number of records remaining to be picked
in	*N*	number of remaining records in the file

Returns

the number of records to skip

5.17.3.7 vitterA()

```
volatile uint64_t RanDraw::vitterA (
            const double & n,
            const double & N )
```

Sample from Vitter's distribution, method A.

Given the number of remaining records in a file N and the number of records n remaining to be selected, sample the number of records to skip over. This function implements Vitter's [3] [2] method A. It is useful for online one-pass sampling of records from a file. While the inputs are integer, we pass them in as *double* because that is more efficient for calculations.

Parameters

in	*n*	number of records remaining to be picked
in	*N*	number of remaining records in the file

Returns

the number of records to skip

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

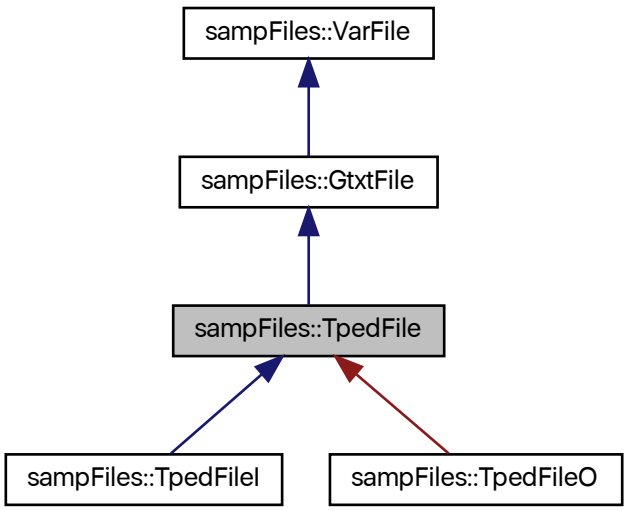
- random.hpp
- random.cpp

5.18 sampFiles::TpedFile Class Reference

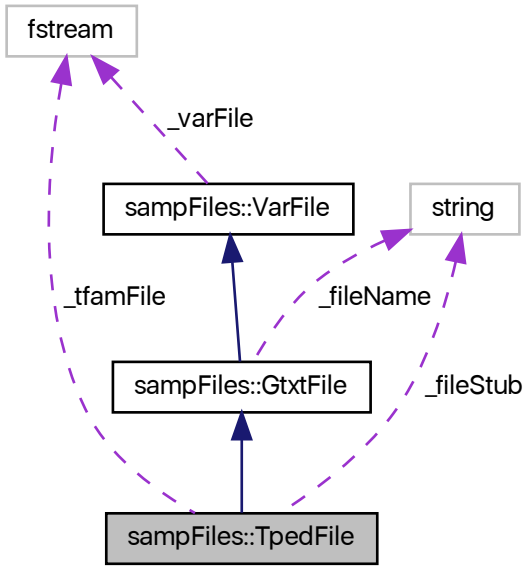
TPED file base class.

`#include <varfiles.hpp>`

Inheritance diagram for sampFiles::TpedFile:



Collaboration diagram for sampFiles::TpedFile:



Public Member Functions

- TpedFile ()

  Default constructor.
- TpedFile (const string &stubName)

  File name constructor.
- TpedFile (const TpedFile &in)=default

  Copy constructor.
- TpedFile & operator= (const TpedFile &in)=default

  Copy assignment.
- TpedFile (TpedFile &&in)=default

  Move constructor.
- TpedFile & operator= (TpedFile &&in)=default

  Move assignment.
- ~TpedFile ()

  Destructor.
- virtual void open ()

  Open stream (does nothing)
- void close ()

  Close stream.

Protected Attributes

- fstream _tfamFile

  Corresponding .tfam file stream.
- string _fileStub

  File name stub (minus the extension)

Additional Inherited Members

5.18.1 Detailed Description

TPED file base class.

Sets up the stream for the corresponding .tfam file.

5.18.2 Constructor & Destructor Documentation

5.18.2.1 TpedFile()

```
sampFiles::TpedFile::TpedFile (
            const string & stubName )  [inline]
```

File name constructor.

Parameters

in	*stubName*	file name minus the extension

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

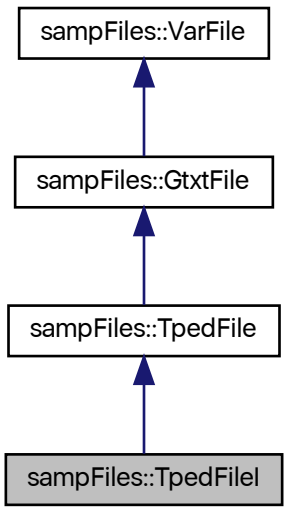
- varfiles.hpp
- varfiles.cpp

5.19 sampFiles::TpedFilel Class Reference

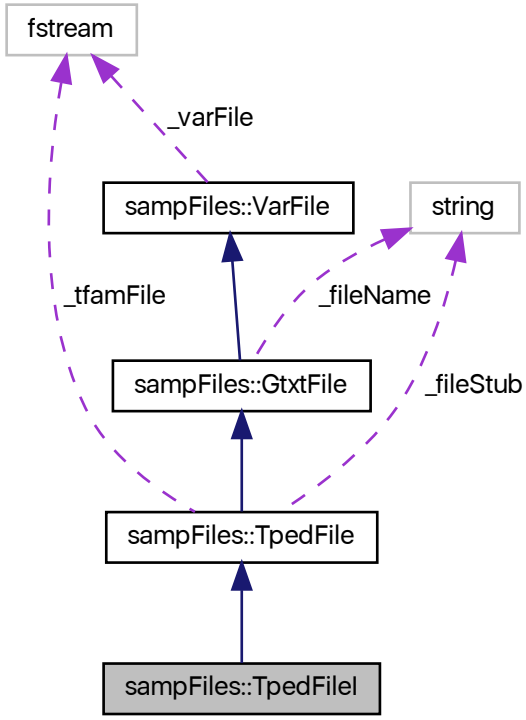
TPED file input class.

```
#include <varfiles.hpp>
```

Inheritance diagram for sampFiles::TpedFileI:



Collaboration diagram for sampFiles::TpedFileI:



Public Member Functions

- TpedFileI ()
  Default constructor.
- TpedFileI (const string &stubName)
  File name constructor.
- TpedFileI (const TpedFileI &in)=default
  Copy constructor.
- TpedFileI & operator= (const TpedFileI &in)=default
  Copy assignment.
- TpedFileI (TpedFileI &&in)=default
  Move constructor.
- TpedFileI & operator= (TpedFileI &&in)=default
  Move assignment.
- ~TpedFileI ()
  Destructor.
- void open ()

Open stream to read.

- void sample (TpedFileO &out, const uint64_t &n)

  Sample SNPs and save to BED file.

- uint64_t nsnp ()

  Number of SNPs in the object.

- uint64_t nindiv ()

  Number of individuals in the object.

Protected Member Functions

- uint64_t _famLines ()

  Get number of lines in the `_tfamFile`

- uint64_t _famLines (fstream &fam)

  Copy the .tfam file and count number of lines.

- void _famCopy (fstream &fam)

  Copy the .tfam file.

- uint64_t _numLines ()

  Get number of rows in the text file.

Additional Inherited Members

5.19.1 Detailed Description

TPED file input class.

Reads TPED files and the corresponding .tfam files as necessary.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 TpedFileI()

```
sampFiles::TpedFileI::TpedFileI (
            const string & stubName )  [inline]
```

File name constructor.

Parameters

in	*stubName*	file name minus the extension

5.19.3 Member Function Documentation

5.19.3.1 _famCopy()

```
void TpedFileI::_famCopy (
            fstream & fam )  [protected]
```

Copy the .tfam file.

The current object's .tfam file is copied to the provided file stream, which should already be open for writing. If not, the function throws a *string* object ``Output .fam filestream not open''.

Parameters

in	*fam*	.tfam file stream

5.19.3.2 _famLines() [1/2]

```
uint64_t TpedFileI::_famLines ( )  [protected]
```

Get number of lines in the `_tfamFile`

Assumes Unix-like line endings. The result is equal to the number of individuals. The `_tfamFile` should already be open for reading.

Returns

number of lines in `_tfamFile`

5.19.3.3 _famLines() [2/2]

```
uint64_t TpedFileI::_famLines (
            fstream & fam )  [protected]
```

Copy the .tfam file and count number of lines.

Assumes Unix-like line endings. The result is equal to the number of individuals. The current object's .tfam file is copied to the provided file stream, which should already be open for writing. If not, the function throws a *string* object ``Output .fam filestream not open''.

Parameters

in	*fam*	.tfam file stream

Returns

number of lines in `_tfamFile`

5.19.3.4 _numLines()

```
uint64_t TpedFileI::_numLines ( )  [protected]
```

Get number of rows in the text file.

Assumes Unix-like line endings. Header, if present, is not counted. Is overriden in some, but not all, derived classes.

Returns

number of rows

5.19.3.5 sample()

```
void TpedFileI::sample (
            TpedFileO & out,
            const uint64_t & n )
```

Sample SNPs and save to BED file.

Sample n SNPs without replacement from the file represented by the current object and save to the `out` object. Uses Vitter's [2] method. Number of samples has to be smaller that the number of SNPs in the file.

Parameters

in	*out*	output object
in	*n*	number of SNPs to sample

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

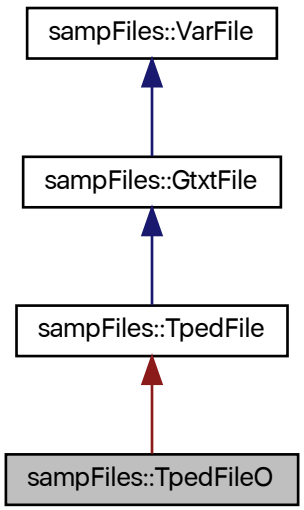
- varfiles.hpp
- varfiles.cpp

5.20 sampFiles::TpedFileO Class Reference

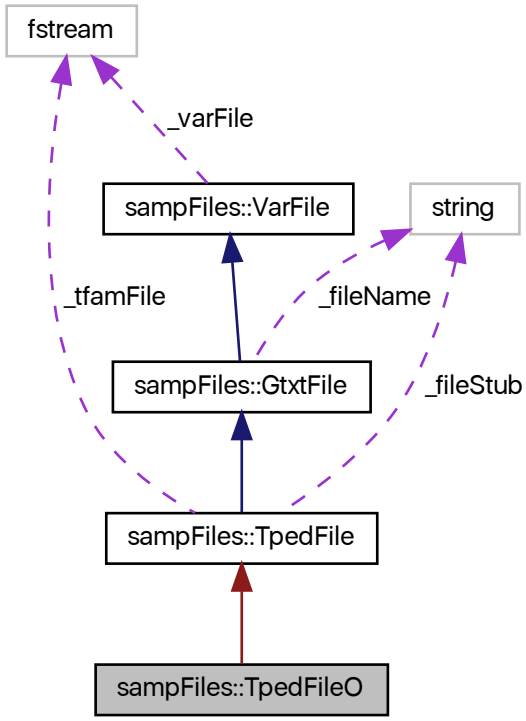
TPED file output class.

```
#include <varfiles.hpp>
```

Inheritance diagram for sampFiles::TpedFileO:



Collaboration diagram for sampFiles::TpedFileO:



Public Member Functions

- TpedFileO ()
  Default constructor.
- TpedFileO (const string &stubName)
  File name constructor.
- TpedFileO (const TpedFileO &in)=default
  Copy constructor.
- TpedFileO & operator= (const TpedFileO &in)=default
  Copy assignment.
- TpedFileO (TpedFileO &&in)=default
  Move constructor.
- TpedFileO & operator= (TpedFileO &&in)=default
  Move assignment.
- ~TpedFileO ()
  Destructor.
- void open ()
  Open stream to write.

Friends

- class **TpedFileI**

5.20.1 Detailed Description

TPED file output class.

Writes to TPED files and the corresponding .tfam files as necessary. Data are written in the SNP-major format.

5.20.2 Constructor & Destructor Documentation

5.20.2.1 TpedFileO()

```
sampFiles::TpedFileO::TpedFileO (
            const string & stubName )  [inline]
```

File name constructor.

Parameters

in	*stubName*	file name minus the extension

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

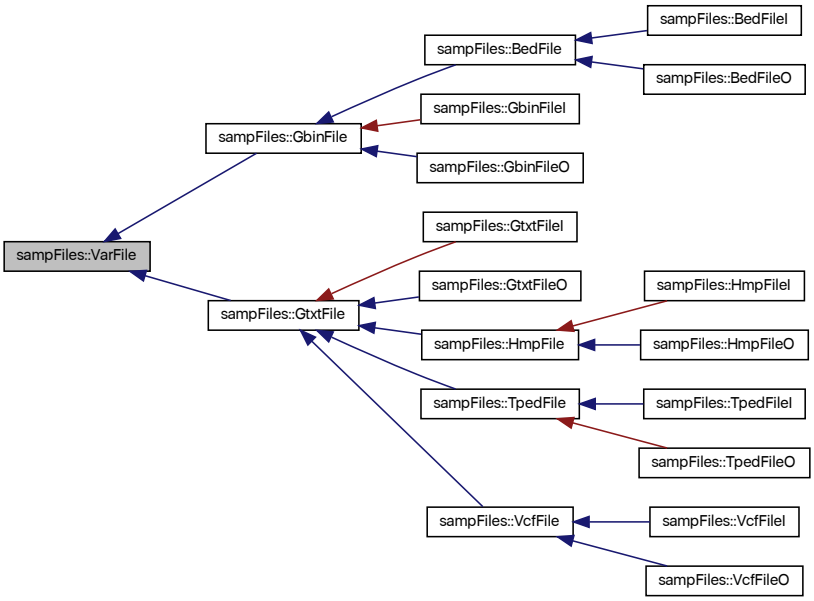
- varfiles.hpp
- varfiles.cpp

5.21 sampFiles::VarFile Class Reference

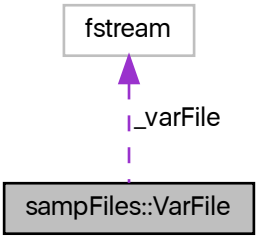
Base variant file class.

```
#include <varfiles.hpp>
```

Inheritance diagram for sampFiles::VarFile:



Collaboration diagram for sampFiles::VarFile:



Public Member Functions

- VarFile (const VarFile &in)=default

  Copy constructor.
- VarFile & operator= (const VarFile &in)=default

  Copy assignment.
- VarFile (VarFile &&in)=default

Move constructor.

- VarFile & operator= (VarFile &&in)=default

  Move assignment.

- ~VarFile ()

  Destructor.

- virtual void open ()=0

  Open stream.

- virtual void close ()=0

  Close stream.

Protected Member Functions

- VarFile ()

  Default constructor (protected)

Protected Attributes

- fstream _varFile

  Variant file stream.

5.21.1 Detailed Description

Base variant file class.

Abstract base class for all the input/output formats. Cannot be initialized directly.

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

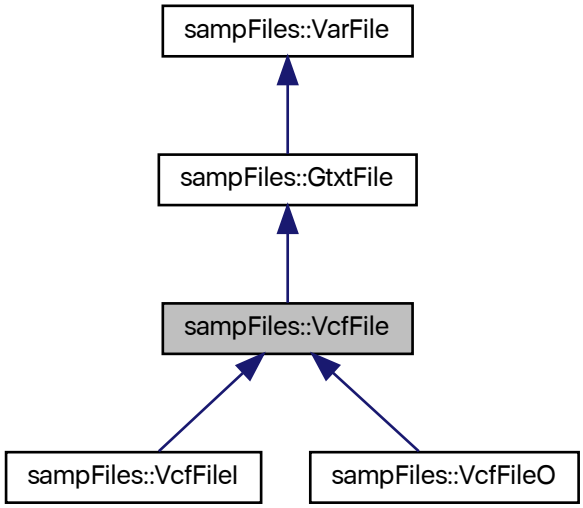
- varfiles.hpp

5.22 sampFiles::VcfFile Class Reference

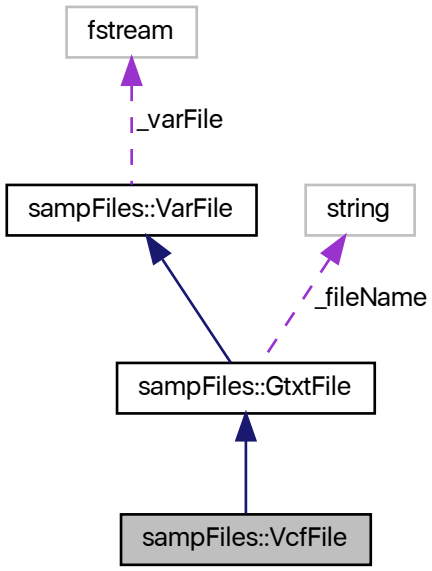
VCF file base class.

```
#include <varfiles.hpp>
```

Inheritance diagram for sampFiles::VcfFile:



Collaboration diagram for sampFiles::VcfFile:



Public Member Functions

- VcfFile ()
  Default constructor.
- VcfFile (const string &fileName)
  Constructor with file name.
- VcfFile (const VcfFile &in)=default
  Copy constructor.
- VcfFile & operator= (const VcfFile &in)=default
  Copy assignment.
- VcfFile (VcfFile &&in)=default
  Move constructor.
- VcfFile & operator= (VcfFile &&in)=default
  Move assignment.
- ~VcfFile ()
  Destructor.
- void open ()
  Open stream (does nothing)
- void close ()
  Close stream.

Additional Inherited Members

5.22.1 Detailed Description

VCF file base class.

Sets up streams for VCF files. Any accompanying .idx files are ignored.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 VcfFile()

```
sampFiles::VcfFile::VcfFile (
            const string & fileName )  [inline]
```

Constructor with file name.

Parameters

in	*fileName*	file name

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

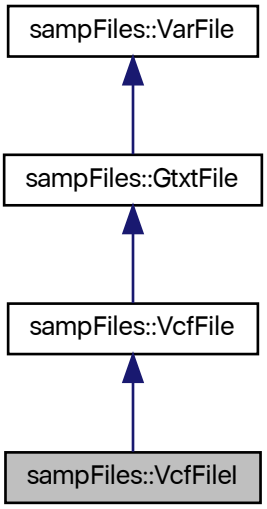
- varfiles.hpp
- varfiles.cpp

5.23 sampFiles::VcfFileI Class Reference

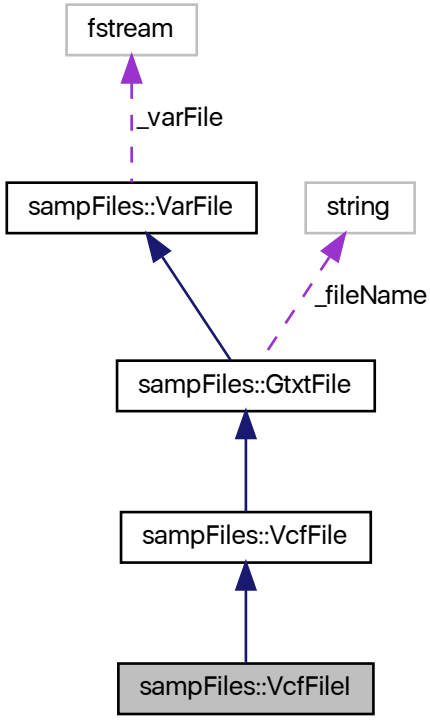
VCF file input class.

```
#include <varfiles.hpp>
```

Inheritance diagram for sampFiles::VcfFileI:



Collaboration diagram for sampFiles::VcfFileI:



Public Member Functions

- VcfFileI ()
  Default constructor.
- VcfFileI (const string &fileName)
  File name constructor.
- VcfFileI (const VcfFileI &in)=default
  Copy constructor.
- VcfFileI & operator= (const VcfFileI &in)=default
  Copy assignment.
- VcfFileI (VcfFileI &&in)=default
  Move constructor.
- VcfFileI & operator= (VcfFileI &&in)=default
  Move assignment.
- ~VcfFileI ()
  Destructor.
- void open ()

Open stream to read.

- void sample (VcfFileO &out, const uint64_t &n)

  Sample SNPs and save to VCF file.

- uint64_t nsnp ()

  Number of SNPs in the object.

Protected Member Functions

- uint64_t _numLines ()

  Get number of SNPs in the VCF file.

Additional Inherited Members

5.23.1 Detailed Description

VCF file input class.

Reads VCF files, skipping or copying the header as necessary; .idx files are ignored.

5.23.2 Constructor & Destructor Documentation

5.23.2.1 VcfFileI()

```
sampFiles::VcfFileI::VcfFileI (
            const string & fileName )  [inline]
```

File name constructor.

Parameters

in	*fileName*	file name including extension

5.23.3 Member Function Documentation

5.23.3.1 _numLines()

```
uint64_t VcfFileI::_numLines ( )  [protected]
```

Get number of SNPs in the VCF file.

Assumes Unix-like line endings. Header is not counted.

Returns

number of SNPs

5.23.3.2 sample()

```
void VcfFileI::sample (
            VcfFileO & out,
            const uint64_t & n )
```

Sample SNPs and save to VCF file.

Sample n SNPs without replacement from the file represented by the current object and save to the `out` object. Uses Vitter's [2] method. Number of samples has to be smaller that the number of SNPs in the file.

Parameters

in	*out*	output object
in	*n*	number of SNPs to sample

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

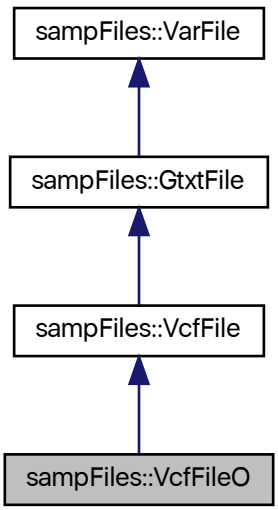
- varfiles.hpp
- varfiles.cpp

5.24 sampFiles::VcfFileO Class Reference

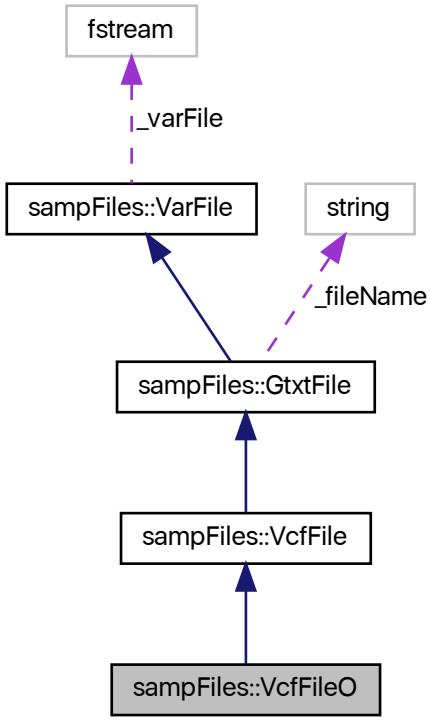
VCF file output class.

```
#include <varfiles.hpp>
```

Inheritance diagram for sampFiles::VcfFileO:



Collaboration diagram for sampFiles::VcfFileO:



Public Member Functions

- VcfFileO ()
  Default constructor.
- VcfFileO (const string &fileName)
  File name constructor.
- VcfFileO (const VcfFileO &in)=default
  Copy constructor.
- VcfFileO & operator= (const VcfFileO &in)=default
  Copy assignment.
- VcfFileO (VcfFileO &&in)=default
  Move constructor.
- VcfFileO & operator= (VcfFileO &&in)=default
  Move assignment.
- ~VcfFileO ()
  Destructor.
- void open ()
  Open stream to write.

Friends

- class **VcfFileI**

Additional Inherited Members

5.24.1 Detailed Description

VCF file output class.

Writes VCF files.

5.24.2 Constructor & Destructor Documentation

5.24.2.1 VcfFileO()

```
sampFiles::VcfFileO::VcfFileO (
            const string & fileName )  [inline]
```

File name constructor.

Parameters

in	*fileName*	file name including the extension

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

- varfiles.hpp
- varfiles.cpp

Chapter 6

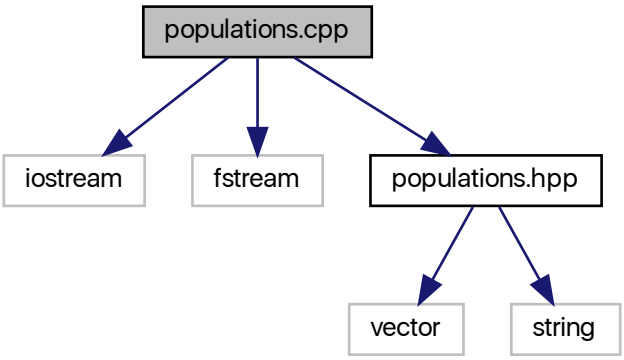
File Documentation

6.1 populations.cpp File Reference

Connect lines with populations.

```
#include <iostream>
#include <fstream>
#include "populations.hpp"
```

Include dependency graph for populations.cpp:



6.1.1 Detailed Description

Connect lines with populations.

Author

Anthony J. Greenberg

Copyright

Version

1.0

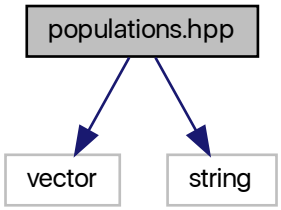
Implementation of the class that relates individual lines to populations they belong to.

6.2 populations.hpp File Reference

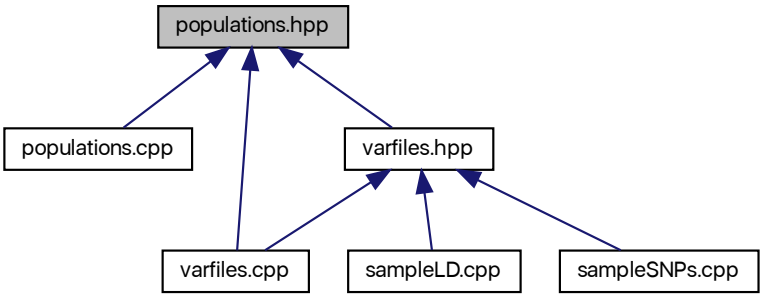
Connect lines with populations.

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

Include dependency graph for populations.hpp:



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



Classes

- class sampFiles::PopIndex

  Population index.

6.2.1 Detailed Description

Connect lines with populations.

Author

Anthony J. Greenberg

Copyright

Version

1.0

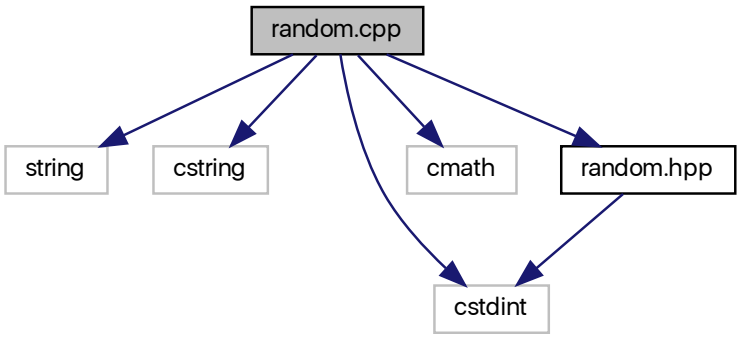
Definitions and interface documentation for the class that relates individual lines to populations they belong to.

6.3 random.cpp File Reference

Random number generation.

```
#include <string>
#include <cstring>
#include <cstdint>
#include <cmath>
#include "random.hpp"
```

Include dependency graph for random.cpp:



6.3.1 Detailed Description

Random number generation.

Author

Anthony J. Greenberg

Copyright

Copyright (c) 2017 Anthony J. Greenberg

Version

0.1

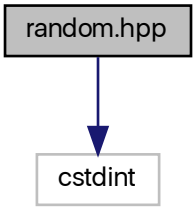
Class implementation for facilities that generate random draws from various distributions.

6.4 random.hpp File Reference

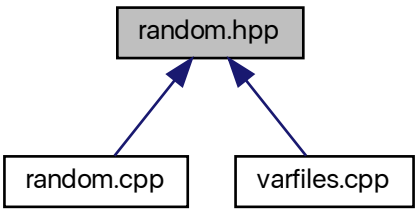
Random number generation.

```
#include <cstdint>
```

Include dependency graph for random.hpp:



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



Classes

- class sampFiles::Generate

  Abstract base random number class.
- class sampFiles::GenerateHR

  Hardware random number generating class.
- class sampFiles::GenerateMT

  Pseudo-random number generator.
- class sampFiles::RanDraw

  Random number generating class.

6.4.1 Detailed Description

Random number generation.

Author

Anthony J. Greenberg

Copyright

Copyright (c) 2017 Anthony J. Greenberg

Version

0.1

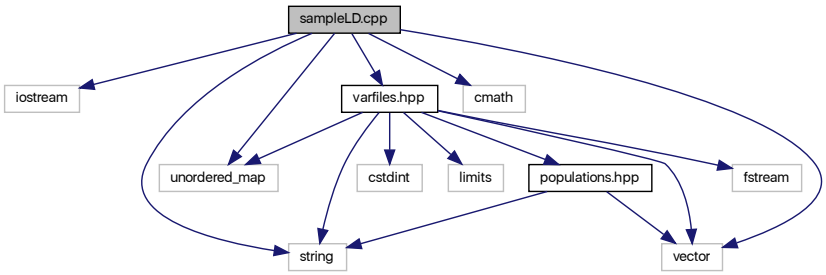
Class definition and interface documentation for facilities that generate random draws from various distributions.

6.5 sampleLD.cpp File Reference

Sample-based linkage disequilibrium.

```
#include <iostream>
#include <string>
#include <vector>
#include <unordered_map>
#include <cmath>
#include "varfiles.hpp"
```

Include dependency graph for sampleLD.cpp:



Functions

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

  Command line parser.
- int **main** (int argc, char *argv[ ])

6.5.1 Detailed Description

Sample-based linkage disequilibrium.

Author

Anthony J. Greenberg

Copyright

Copyright (c) 2017 Anthony J. Greenberg

Version

0.5

Using the *varfiles* library to sample SNPs calculate pairwise LD.

6.5.2 Function Documentation

6.5.2.1 parseCL()

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

Command line parser.

Maps flags to values. Flags assumed to be of the form -x.

Parameters

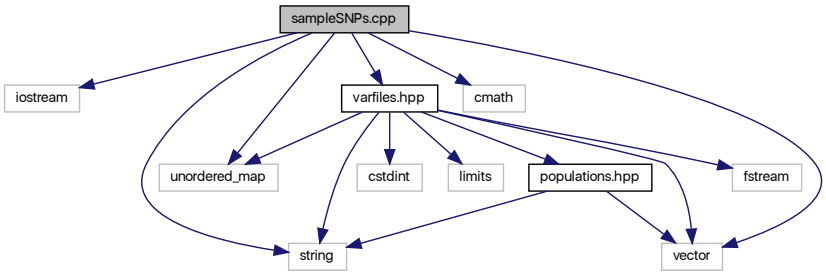
in	*argc*	size of the `argv` array
in	*argv*	command line input array
out	*cli*	map of tags to values

6.6 sampleSNPs.cpp File Reference

Sample SNPs.

```
#include <iostream>
#include <string>
#include <vector>
#include <unordered_map>
#include <cmath>
#include "varfiles.hpp"
```

Include dependency graph for sampleSNPs.cpp:



Functions

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

  Command line parser.
- int **main** (int argc, char *argv[ ])

6.6.1 Detailed Description

Sample SNPs.

Author

Anthony J. Greenberg

Copyright

Version

0.5

Using the *varfiles* library to sample SNPs from a variety of file formats.

6.6.2 Function Documentation

6.6.2.1 parseCL()

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

Command line parser.

Maps flags to values. Flags assumed to be of the form -x.

Parameters

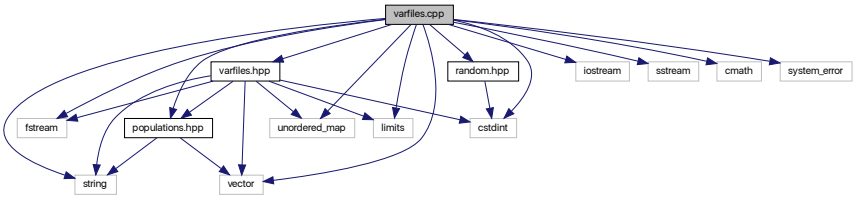
in	*argc*	size of the `argv` array
in	*argv*	command line input array
out	*cli*	map of tags to values

6.7 varfiles.cpp File Reference

Read and write genetic variant files.

```
#include "varfiles.hpp"
#include "random.hpp"
#include "populations.hpp"
#include <fstream>
#include <string>
#include <vector>
#include <unordered_map>
#include <iostream>
#include <sstream>
#include <cstdint>
#include <cmath>
#include <limits>
#include <system_error>
```

Include dependency graph for varfiles.cpp:



6.7.1 Detailed Description

Read and write genetic variant files.

Author

Anthony J. Greenberg

Copyright

Version

0.1

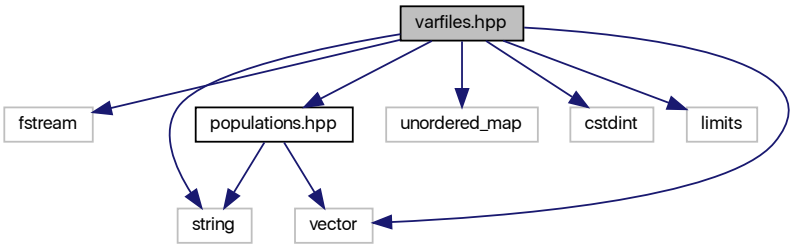
Implementation of classes that read and write various genetic variant file formats.

6.8 varfiles.hpp File Reference

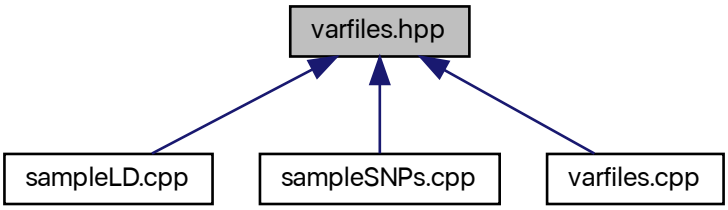
Read and write genetic variant files.

```
#include <fstream>
#include <string>
#include <vector>
#include <unordered_map>
#include <cstdint>
#include <limits>
#include "populations.hpp"
```

Include dependency graph for varfiles.hpp:



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



Classes

- class sampFiles::VarFile

  Base variant file class.
- class sampFiles::GbinFile

  Generic binary file base class.

- class sampFiles::GbinFileI

  Binary file input class.
- class sampFiles::GbinFileO

  Generic binary file output class.
- class sampFiles::BedFile

  BED file base class.
- class sampFiles::BedFileI

  BED file input class.
- class sampFiles::BedFileO

  BED file output class.
- class sampFiles::GtxtFile

  Generic text file base class.
- class sampFiles::GtxtFileI

  Text file input class.
- class sampFiles::GtxtFileO

  Generic text file output class.
- class sampFiles::TpedFile

  TPED file base class.
- class sampFiles::TpedFileI

  TPED file input class.
- class sampFiles::TpedFileO

  TPED file output class.
- class sampFiles::VcfFile

  VCF file base class.
- class sampFiles::VcfFileI

  VCF file input class.
- class sampFiles::VcfFileO

  VCF file output class.
- class sampFiles::HmpFile

  Hapmap (HMP) file base class.
- class sampFiles::HmpFileI

  HMP file input class.
- class sampFiles::HmpFileO

  HMP file output class.

Variables

- static const size_t sampFiles::BUF_SIZE = 10485760

  Buffer size.
- const double sampFiles::EPS = numeric_limits<double>::epsilon()

  Machine ϵ.
- const double sampFiles::PI = 3.14159265358979323846264338328

  pi

6.8.1 Detailed Description

Read and write genetic variant files.

Author

Anthony J. Greenberg

Copyright

Version

0.1

Definitions and interface documentation for classes that read and write various genetic variant file formats.

Currently supported formats:

- *plink* BED
- *plink* TPED
- VCF
- Hapmap (.hmp.txt)

6.8.2 Variable Documentation

6.8.2.1 BUF_SIZE

```
const size_t sampFiles::BUF_SIZE = 10485760  [static]
```

Buffer size.

Size of the buffer for reading files text files. I use it in functions that count the number of lines, for example. The buffer size (10M) is optimized for a MacBook Pro with an SSD. Other systems may perform better with a different value (e.g., if you have a spinning drive and more RAM you may want to experiemtn with increasing it).

Bibliography

[1] M Matsumoto and T Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM TOMACS*, 8(1):3–30, 1998. 1, 37

[2] Jeffrey S. Vitter. An efficient algorithm for sequential random sampling. *ACM Trans. Math. Softw.*, 13(1):58–67, March 1987. 17, 26, 45, 46, 55, 66, 74, 85

[3] Jeffrey Scott Vitter. Faster methods for random sampling. *Commun. ACM*, 27(7):703–718, July 1984. 66

Index