# vash

Generated by Doxygen 1.9.6

# Chapter 1

# Overview

A C++14 library and software to efficiently summarize genetic polymorphism statistics from binary variant files. Linkage disequilibrium (LD) between pairs of loci is estimated using Locality-Sensitive Hashing ( LSH), in particular a variant of the one-permutation hash Mai *et al.*, 2020. These hashes can be used to efficiently group loci by similarity in hash tables. Jaccard similarity is used as the LD statistic because this is the measure approximated by LSH collision probability.

The gist of the method is that each locus (or individual, although the latter functionality is not yet available) data are converted to one-bit encoding by setting the major allele (as well as missing genotype values) to 0, minor allele to 1, and heterozygotes to 1 with 50% probability. The resulting binary data are hashed, the hashes banded (as described, e.g., in the Mining massive data sets book, Chapter 3.4), and loci assigned to buckets in an unordered hash table. Loci end up in the same bucket if at least one hash band is identical between them. LD is then estimated within buckets only. Thus, if high-LD pairs are rare in a data set, many fewer than $N(N-1)/2$ pairwise LD values are estimated.

## 1.1 Dependencies

Building the library and binaries requires a C++14 compiler. The build process requires `cmake` version 3.11 or later, but everything can be compiled by hand if desired. Current implementation requires `x86_64` processors and has only been tested on Linux. I plan to add 64-bit ARM (e.g., Apple M processor) support.

## 1.2 Download and install

The repository comes with a submodule, so to clone use

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

Next, create a build directory

```
cd vash
mkdir build
```

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

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

Installation may require root privileges.

## 1.3 Use ldblocks

`lblocks` is a command line tool that estimates LD among loci in a `plink .bed` file. Running it without command line flags prints the flags and their possible values. Most flags are self-explanatory, but setting values to some of them requires special consideration.

```
--hash-size       - bigger values result in better estimates of LD, but at the expense of speed.
    Based on some experimentation, minimal recommended value is 40, and 100 seems sufficient.
    Setting this flag to 0 leads to full (not hash-based) Jaccard estimates among all pairs.

--n-rows-per-band - is the banding parameter that controls sparsity.
    Setting it high (1/5 the hash size or more) leads to lower probability of
    inclusion of low to moderately similar pairs.
    Setting this flag 0 leads to all pairwise estates to be calculated.
    Beware, since this can result in huge output files.
    The software does its best to not use up RAM, but this has only been tested on Linux.

--only-groups     - no LD calculations are performed and only groups
    and the loci they contain is saved to a file.
--add-locus-names - locus names from the corresponding `.bim` file are outputinstead
    instead the default (base-1) locus indexes. This may result in larger output files.
```

Output files are tab-delimited and include group IDs, locus pair indexes, and Jaccard similarity estimates. If full Jaccard estimates are produced, group IDs are not included, but $r^2$ as well as Jaccard similarity estimates are output.

Running the software on whole genomes with millions of loci should not tax RAM (the software keeps track of free memory and only uses about half available RAM), but can still tax disk space. In addition, some pairs can be assigned to more than one group. Removal of these duplicates requires in-memory operations, so if partial results are written to disk some of these duplicates are retained. I recommend using the `--only-groups` flags in preliminary runs to get a sense of the number of locus pairs that will result given a set of parameters. Analyzing a single chromosome at a time also speeds up the analyses.

Library interface documentation can be found here.

# 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 BayesicSpace::GenoTableBin Class Reference

Class to store binary compressed genotype tables.

`#include <gvarHash.hpp>`

### Public Member Functions

- **GenoTableBin** ()

  *Default constructor.*
- GenoTableBin (const std::string &inputFileName, const size_t &nIndividuals, std::string logFileName)

  *Constructor with input file name.*
- GenoTableBin (const std::string &inputFileName, const size_t &nIndividuals, std::string logFileName, const size_t &nThreads)

  *Constructor with input file name and thread count.*
- GenoTableBin (const std::vector< int > &maCounts, const size_t &nIndividuals, std::string logFileName)

  *Constructor with count vector.*
- GenoTableBin (const std::vector< int > &maCounts, const size_t &nIndividuals, std::string logFileName, const size_t &nThreads)

  *Constructor with count vector and thread count.*
- **GenoTableBin** (const GenoTableBin &toCopy)=delete

  *Copy constructor (deleted)*
- GenoTableBin **operator=** (const GenoTableBin &toCopy)=delete

  *Copy assignment operator (deleted)*
- GenoTableBin (GenoTableBin &&toMove) noexcept

  *Move constructor.*
- GenoTableBin & operator= (GenoTableBin &&toMove) noexcept

  *Move assignment operator.*
- ~**GenoTableBin** ()=default

  *Destructor.*
- void saveGenoBinary (const std::string &outFileName) const

*Save the binary genotype file.*

- void allJaccardLD (const std::string &ldFileName) const

  *All by all Jaccard similarity LD.*

- void allJaccardLD (const std::string &bimFileName, const std::string &ldFileName) const

  *All by all Jaccard similarity LD with locus names.*

- void saveLogFile () const

  *Save the log to a file.*

### 4.1.1 Detailed Description

Class to store binary compressed genotype tables.

Converts genotype data to a lossy compressed binary code. Genotypes are stored in memory in a one-bit format: bit set for the minor allele, unset for the major. Bits corresponding to missing data are unset (this is the same as mean imputation), heterozygotes are set with a 50% probability.

### 4.1.2 Constructor & Destructor Documentation

#### 4.1.2.1 GenoTableBin() [1/5]

```
BayesicSpace::GenoTableBin::GenoTableBin (
            const std::string & inputFileName,
            const size_t & nIndividuals,
            std::string logFileName )  [inline]
```

Constructor with input file name.

The file should be in the `plink .bed format`. Heterozygotes are assigned the major or minor allele at random, missing genotypes are assigned the major allele. If necessary, alleles are re-coded so that the set bit is always the minor allele.

**Parameters**

| | | |
|---|---|---|
| in | *inputFileName* | input file name |
| in | *nIndividuals* | number of genotyped individuals |
| in | *logFileName* | name of the log file |

#### 4.1.2.2 GenoTableBin() [2/5]

```
GenoTableBin::GenoTableBin (
            const std::string & inputFileName,
```

```
            const size_t & nIndividuals,
            std::string logFileName,
            const size_t & nThreads )
```

Constructor with input file name and thread count.

The file should be in the `plink .bed format`. Heterozygotes are assigned the major or minor allele at random, missing genotypes are assigned the major allele. If necessary, alleles are re-coded so that the set bit is always the minor allele. The number of threads requested is maximum to be used, depending on available system resources.

**Parameters**

| | | |
|---|---|---|
| in | *inputFileName* | input file name |
| in | *nIndividuals* | number of genotyped individuals |
| in | *logFileName* | name of the log file |
| in | *nThreads* | maximal number of threads to use |

#### 4.1.2.3 GenoTableBin() [3/5]

```
BayesicSpace::GenoTableBin::GenoTableBin (
            const std::vector< int > & maCounts,
            const size_t & nIndividuals,
            std::string logFileName )  [inline]
```

Constructor with count vector.

Input is a vector of minor allele counts (0, 1, or 2) or -9 for missing data. Heterozygotes are assigned the major or minor allele at random, missing genotypes are assigned the major allele. The counts are checked and re-coded if necessary so that set bits represent the minor allele. This function should run faster if the 0 is the major allele homozygote. While the above values are the norm, any negative number will be interpreted as missing, any odd number as 1, and any (non-0) even number as 2. The input is a vectorized matrix of genotypes. The original matrix has individuals on rows, and is vectorized by row.

**Parameters**

| | | |
|---|---|---|
| in | *maCounts* | vector of minor allele numbers |
| in | *nIndividuals* | number of genotyped individuals |
| in | *logFileName* | name of the log file |

#### 4.1.2.4 GenoTableBin() [4/5]

```
GenoTableBin::GenoTableBin (
            const std::vector< int > & maCounts,
```

```
            const size_t & nIndividuals,
            std::string logFileName,
            const size_t & nThreads )
```

Constructor with count vector and thread count.

Input is a vector of minor allele counts (0, 1, or 2) or -9 for missing data. Heterozygotes are assigned the major or minor allele at random, missing genotypes are assigned the major allele. The counts are checked and re-coded if necessary so that set bits represent the minor allele. This function should run faster if the 0 is the major allele homozygote. While the above values are the norm, any negative number will be interpreted as missing, any odd number as 1, and any (non-0) even number as 2. The input is a vectorized matrix of genotypes. The original matrix has individuals on rows, and is vectorized by row. The number of threads requested is maximum to be used, depending on available system resources.

**Parameters**

| | | |
|---|---|---|
| in | *maCounts* | vector of minor allele numbers |
| in | *nIndividuals* | number of genotyped individuals |
| in | *logFileName* | name of the log file |
| in | *nThreads* | maximal number of threads to use |

#### 4.1.2.5 GenoTableBin() [5/5]

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

Move constructor.

**Parameters**

| | | |
|---|---|---|
| in | *toMove* | object to move |

### 4.1.3 Member Function Documentation

#### 4.1.3.1 allJaccardLD() [1/2]

```
void GenoTableBin::allJaccardLD (
            const std::string & bimFileName,
            const std::string & ldFileName ) const
```

All by all Jaccard similarity LD with locus names.

Calculates linkage disequilibrium among all loci using Jaccard similarity as the statistic. Result is a vectorized lower triangle of the symmetric $N \times N$ similarity matrix, where $N$ is the number of loci. All values belong to the same group. Row and column locus names are also included in the tab-delimited output file. The lower triangle is vectorized by column (i.e. all correlations of the first locus, then all remaining correlations of the second, etc.).

**Parameters**

| | | |
|---|---|---|
| in | *bimFileName* | name of the _.bim_ file that has locus names |
| in | *ldFileName* | name of the output file |

#### 4.1.3.2 allJaccardLD() [2/2]

```
void GenoTableBin::allJaccardLD (
            const std::string & ldFileName ) const
```

All by all Jaccard similarity LD.

Calculates linkage disequilibrium among all loci using Jaccard similarity as the statistic. Result is a vectorized lower triangle of the symmetric $N \times N$ similarity matrix, where $N$ is the number of loci. All values belong to the same group. Row and column (1-base) indexes of the similarity matrix are also included in the tab-delimited output file. The lower triangle is vectorized by column (i.e. all correlations of the first locus, then all remaining correlations of the second, etc.).

**Parameters**

| | | |
|---|---|---|
| in | *ldFileName* | name of the output file |

**Returns**

lower triangle of the LD matrix

#### 4.1.3.3 operator=()

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

Move assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *toMove* | object to be moved |

**Returns**

GenoTableBin object

#### 4.1.3.4 saveGenoBinary()

```
void GenoTableBin::saveGenoBinary (
            const std::string & outFileName ) const
```

Save the binary genotype file.

Saves the binary approximate genotype data to a binary file.

**Parameters**

| | | |
|---|---|---|
| in | *outFileName* | output file name |

#### 4.1.3.5 saveLogFile()

```
void GenoTableBin::saveLogFile ( ) const
```

Save the log to a file.

Log file name provided at construction.

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

- include/gvarHash.hpp
- src/gvarHash.cpp

## 4.2 BayesicSpace::GenoTableHash Class Reference

Class to store compressed genotype tables.

```
#include <gvarHash.hpp>
```

## Public Member Functions

- **GenoTableHash** ()

  *Default constructor.*
- GenoTableHash (const std::string &inputFileName, const size_t &nIndividuals, const size_t &kSketches, const size_t &nThreads, std::string logFileName)

  *Constructor with input file name and thread number.*
- GenoTableHash (const std::string &inputFileName, const size_t &nIndividuals, const size_t &kSketches, std↩ ::string logFileName)

  *Constructor with input file name.*
- GenoTableHash (const std::vector< int > &maCounts, const size_t &nIndividuals, const size_t &kSketches, const size_t &nThreads, std::string logFileName)

  *Constructor with count vector and thread number.*
- GenoTableHash (const std::vector< int > &maCounts, const size_t &nIndividuals, const size_t &kSketches, std↩ ::string logFileName)

  *Constructor with count vector.*
- **GenoTableHash** (const GenoTableHash &toCopy)=delete

  *Copy constructor (deleted)*
- GenoTableHash **operator=** (const GenoTableHash &toCopy)=delete

  *Copy assignment operator (deleted)*
- GenoTableHash (GenoTableHash &&toMove) noexcept

  *Move constructor.*
- GenoTableHash & operator= (GenoTableHash &&toMove) noexcept

  *Move assignment operator.*
- ~**GenoTableHash** ()=default

  *Destructor.*
- void allHashLD (const std::string &ldFileName) const

  *All by all LD from hashes.*
- void allHashLD (const std::string &bimFileName, const std::string &ldFileName) const

  *All by all LD from hashes with locus names.*
- std::vector< std::vector< uint32_t > > makeLDgroups (const size_t &nRowsPerBand) const

  *Assign groups by linkage disequilibrium (LD)*
- void makeLDgroups (const size_t &nRowsPerBand, const std::string &outFileName) const

  *Assign groups by LD and save to a file.*
- void makeLDgroups (const size_t &nRowsPerBand, const std::string &bimFileName, const std::string &outFile↩ Name) const

  *Assign groups by LD and save to a file with locus names.*
- void ldInGroups (const size_t &nRowsPerBand, const std::string &outFileName) const

  *LD in groups.*
- void ldInGroups (const size_t &nRowsPerBand, const std::string &bimFileName, const std::string &outFileName) const

  *LD in groups with locus names.*
- void saveLogFile () const

  *Save the log to a file.*

### 4.2.1 Detailed Description

Class to store compressed genotype tables.

Provides facilities to store and manipulate compressed genotype tables. Genotypes are stored in a one-bit format: bit set for the minor allele, unset for the major. Bits corresponding to missing data are unset (this is the same as mean imputation), heterozygotes are set with a 50% probability.

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 GenoTableHash() [1/5]

```
GenoTableHash::GenoTableHash (
            const std::string & inputFileName,
            const size_t & nIndividuals,
            const size_t & kSketches,
            const size_t & nThreads,
            std::string logFileName )
```

Constructor with input file name and thread number.

The file should be in the `plink .bed format` format. Heterozygotes are assigned the major or minor allele at random, missing genotypes are assigned the major allele. If necessary, alleles are re-coded so that the set bit is always the minor allele. The binary stream is then hashed using a one-permutation hash (OPH; one sketch per locus). Bits are permuted using the Fisher-Yates-Durstenfeld algorithm. Filling in empty bins using the Mai *et al.* (2020) algorithm. The number of threads specified is the maximal that will be used. Actual number depends on system resources.

**Parameters**

| | | |
|---|---|---|
| in | *inputFileName* | input file name |
| in | *nIndividuals* | number of genotyped individuals |
| in | *kSketches* | number of sketches per locus |
| in | *nThreds* | maximal number of threads to use |
| in | *logFileName* | name of the log file |

#### 4.2.2.2 GenoTableHash() [2/5]

```
BayesicSpace::GenoTableHash::GenoTableHash (
            const std::string & inputFileName,
            const size_t & nIndividuals,
            const size_t & kSketches,
            std::string logFileName )  [inline]
```

Constructor with input file name.

The file should be in the `plink .bed format` format. Heterozygotes are assigned the major or minor allele at random, missing genotypes are assigned the major allele. If necessary, alleles are re-coded so that the set bit is always the minor allele. The input is a vectorized matrix of genotypes. The original matrix has individuals on rows, and is vectorized by row. The binary stream is then hashed using a one-permutation hash (OPH; one sketch per locus). Bits are permuted using the Fisher-Yates-Durstenfeld algorithm. Filling in empty bins using the Mai *et al.* (2020) algorithm.

**Parameters**

| | | |
|---|---|---|
| in | *inputFileName* | input file name |
| in | *nIndividuals* | number of genotyped individuals |
| in | *kSketches* | the number of sketches per locus |
| in | *logFileName* | name of the log file |

#### 4.2.2.3 GenoTableHash() [3/5]

```
GenoTableHash::GenoTableHash (
            const std::vector< int > & maCounts,
            const size_t & nIndividuals,
            const size_t & kSketches,
            const size_t & nThreads,
            std::string logFileName )
```

Constructor with count vector and thread number.

Input is a vector of minor allele counts (0, 1, or 2) or -9 for missing data. Heterozygotes are assigned the major or minor allele at random, missing genotypes are assigned the major allele. The counts are checked and re-coded if necessary so that set bits represent the minor allele. This function should run faster if the 0 is the major allele homozygote. While the above values are the norm, any negative number will be interpreted as missing, any odd number as 1, and any (non-0) even number as 2. The input is a vectorized matrix of genotypes. The original matrix has individuals on rows, and is vectorized by row. The binary stream is then hashed using a one-permutation hash (OPH; one sketch per locus). Bits are permuted using the Fisher-Yates-Durstenfeld algorithm. Filling in empty bins using the Mai *et al.* (2020) algorithm. The number of threads specified is the maximal that will be used. Actual number depends on system resources.

**Parameters**

| | | |
|---|---|---|
| in | *maCounts* | vector of minor allele numbers |
| in | *nIndividuals* | number of genotyped individuals |
| in | *kSketches* | the number of sketches per locus |
| in | *nThreds* | maximal number of threads to use |
| in | *logFileName* | name of the log file |

#### 4.2.2.4 GenoTableHash() [4/5]

```
BayesicSpace::GenoTableHash::GenoTableHash (
            const std::vector< int > & maCounts,
            const size_t & nIndividuals,
            const size_t & kSketches,
            std::string logFileName )  [inline]
```

Constructor with count vector.

Input is a vector of minor allele counts (0, 1, or 2) or -9 for missing data. Heterozygotes are assigned the major or minor allele at random, missing genotypes are assigned the major allele. The counts are checked and re-coded if necessary so that set bits represent the minor allele. This function should run faster if the 0 is the major allele homozygote. While the above values are the norm, any negative number will be interpreted as missing, any odd number as 1, and any (non-0) even number as 2. The binary stream is then hashed using a one-permutation hash (OPH; one sketch per locus). Bits are permuted using the Fisher-Yates-Durstenfeld algorithm. Filling in empty bins using the Mai *et al.* (2020) algorithm.

**Parameters**

| | | |
|---|---|---|
| in | *maCounts* | vector of minor allele numbers |
| in | *nIndividuals* | number of genotyped individuals |
| in | *kSketches* | the number of sketches per locus |
| in | *logFileName* | name of the log file |

#### 4.2.2.5 GenoTableHash() [5/5]

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

Move constructor.

**Parameters**

| | | |
|---|---|---|
| in | *toMove* | object to move |

### 4.2.3 Member Function Documentation

#### 4.2.3.1 allHashLD() [1/2]

```
void GenoTableHash::allHashLD (
            const std::string & bimFileName,
            const std::string & ldFileName ) const
```

All by all LD from hashes with locus names.

Calculates linkage disequilibrium among all loci using a modified OPH. Result is a vectorized lower triangle of the symmetric $N \times N$ similarity matrix, where $N$ is the number of loci. All values belong to the same group. Row and column locus names are also included in the tab-delimited output file. The lower triangle is vectorized by column (i.e. all correlations of the first locus, then all remaining correlations of the second, etc.).

**Parameters**

| | | |
|---|---|---|
| in | *bimFileName* | name of the _.bim_ file with locus names |
| in | *ldFileName* | name of file to save the results |

#### 4.2.3.2 allHashLD() [2/2]

```
void GenoTableHash::allHashLD (
            const std::string & ldFileName ) const
```

All by all LD from hashes.

Calculates linkage disequilibrium among all loci using a modified OPH. Result is a vectorized lower triangle of the symmetric $N \times N$ similarity matrix, where $N$ is the number of loci. All values belong to the same group. Row and column indexes (1-base) of the similarity matrix are also included in the tab-delimited output file. The lower triangle is vectorized by column (i.e. all correlations of the first locus, then all remaining correlations of the second, etc.).

**Parameters**

| | | |
|---|---|---|
| in | *ldFileName* | name of file to save the results |

#### 4.2.3.3 ldInGroups() [1/2]

```
void GenoTableHash::ldInGroups (
            const size_t & nRowsPerBand,
            const std::string & bimFileName,
            const std::string & outFileName ) const
```

LD in groups with locus names.

Group loci according to LD using the algorithm for `makeLDgroups` and calculate similarity within groups. Output LD (Jaccard similarity) estimates with group IDs and locus names.

**Parameters**

| | | |
|---|---|---|
| in | *nRowsPerBand* | number of rows per sketch matrix band |
| in | *bimFileName* | _.bim_ file name |
| in | *outFileName* | name of the output file |

#### 4.2.3.4 ldInGroups() [2/2]

```
void GenoTableHash::ldInGroups (
            const size_t & nRowsPerBand,
            const std::string & outFileName ) const
```

LD in groups.

Group loci according to LD using the algorithm for `makeLDgroups` and calculate similarity within groups. Output LD (Jaccard similarity) estimates with group IDs and locus indexes.

**Parameters**

| | | |
|---|---|---|
| in | *nRowsPerBand* | number of rows per sketch matrix band |
| in | *outFileName* | name of the output file |

#### 4.2.3.5 makeLDgroups() [1/3]

```
std::vector< std::vector< uint32_t > > GenoTableHash::makeLDgroups (
            const size_t & nRowsPerBand ) const
```

Assign groups by linkage disequilibrium (LD)

The sketch matrix is divided into bands, `nRowsPerBand` rows per band (must be 1 or greater). Locus pairs are included in the pair hash table if all rows in at least one band match. The resulting hash table has groups with at least two loci per group (indexed by a hash of the index vector in the group). Locus indexes are in increasing order within each group. Groups are sorted by first and second locus indexes. Some locus pairs may end up in more than one group, but no groups are completely identical in locus composition.

**Parameters**

| | | |
|---|---|---|
| in | *nRowsPerBand* | number of rows per sketch matrix band |

**Returns**

locus index hash table

#### 4.2.3.6 makeLDgroups() [2/3]

```
void GenoTableHash::makeLDgroups (
            const size_t & nRowsPerBand,
```

```
            const std::string & bimFileName,
            const std::string & outFileName ) const
```

Assign groups by LD and save to a file with locus names.

Assign groups as above and save locus names with their group IDs to a file.

**Parameters**

| | | |
|---|---|---|
| in | *nRowsPerBand* | number of rows per sketch matrix band |
| in | *bimFileName* | _.bim_ file name |
| in | *outFileName* | output file name |

#### 4.2.3.7 makeLDgroups() [3/3]

```
void GenoTableHash::makeLDgroups (
            const size_t & nRowsPerBand,
            const std::string & outFileName ) const
```

Assign groups by LD and save to a file.

Assign groups as above and save locus indexes with their group IDs to a file.

**Parameters**

| | | |
|---|---|---|
| in | *nRowsPerBand* | number of rows per sketch matrix band |
| in | *outFileName* | output file name |

#### 4.2.3.8 operator=()

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

Move assignment operator.

**Parameters**

| | | |
|---|---|---|
| in | *toMove* | object to be moved |

**Returns**

GenoTableHash object

#### 4.2.3.9 saveLogFile()

```
void GenoTableHash::saveLogFile ( ) const
```

Save the log to a file.

Log file name provided at construction.

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

- include/gvarHash.hpp
- src/gvarHash.cpp

## 4.3 BayesicSpace::IndexedPairLD Struct Reference

LD value with indexes.

```
#include <gvarHash.hpp>
```

### Public Attributes

- float **jaccard**
- float **rSq**
- uint32_t **element1ind**
- uint32_t **element2ind**

### 4.3.1 Detailed Description

LD value with indexes.

Groups a Jaccard similarity value and the $r^2$ linkage disequilibrium statistic of two loci with their indexes.

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

- include/gvarHash.hpp

## 4.4 BayesicSpace::IndexedPairSimilarity Struct Reference

Jaccard value with indexes.

`#include <gvarHash.hpp>`

### Public Attributes

- float **similarityValue**
- uint32_t **element1ind**
- uint32_t **element2ind**
- uint32_t **groupID**

### 4.4.1 Detailed Description

Jaccard value with indexes.

Groups a Jaccard similarity value of two elements (e.g., loci or individuals) with their indexes and the ID of the group they belong to.

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

- include/gvarHash.hpp

# Chapter 5

# File Documentation

## 5.1 apps/ldblocks.cpp File Reference

Build local linkage disequilibrium blocks.

```
#include <cstdlib>
#include <iostream>
#include <string>
#include <vector>
#include <array>
#include <unordered_map>
#include <cmath>
#include <stdexcept>
#include <chrono>
#include "gvarHash.hpp"
#include "vashFunctions.hpp"
```

Include dependency graph for ldblocks.cpp:

### Functions

- void BayesicSpace::fullJaccard (const std::unordered_map< std::string, std::string > &stringVariables, const std::unordered_map< std::string, int > &intVariables, const std::string &bimFileName)

  *Full Jaccard estimates.*
- void BayesicSpace::hashJaccard (const std::unordered_map< std::string, std::string > &stringVariables, const std::unordered_map< std::string, int > &intVariables, const std::string &bimFileName)

  *Hash-based Jaccard estimates.*
- int **main** (int argc, char *argv[ ])

### 5.1.1 Detailed Description

Build local linkage disequilibrium blocks.

**Author**

Anthony J. Greenberg

**Copyright**



**Version**

0.5

Uses the variant hashing library to build local LD blocks from *plink* .bed files.

### 5.1.2 Function Documentation

#### 5.1.2.1 fullJaccard()

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

Full Jaccard estimates.

Run the full Jaccard estimates after binary conversion.

**Parameters**

| | | |
|---|---|---|
| in | *stringVariables* | string-valued input flag variables |
| in | *intVariables* | integer-valued input flag variables |
| in | *bimFileName* | .bim file name |

#### 5.1.2.2 hashJaccard()

```
void BayesicSpace::hashJaccard (
```

```
const std::unordered_map< std::string, std::string > & stringVariables,
const std::unordered_map< std::string, int > & intVariables,
const std::string & bimFileName )
```

Hash-based Jaccard estimates.

Run the hash-based Jaccard estimates after binary conversion, possibly in groups.

**Parameters**

| | | |
|---|---|---|
| in | *stringVariables* | string-valued input flag variables |
| in | *intVariables* | integer-valued input flag variables |
| in | *bimFileName* | .bim file name |

## 5.2 include/gvarHash.hpp File Reference

Summarize variant tables by hashing.

```
#include <cstddef>
#include <fstream>
#include <vector>
#include <array>
#include <unordered_map>
#include <utility>
#include <string>
#include <thread>
#include <mutex>
#include "random.hpp"
```

Include dependency graph for gvarHash.hpp: This graph shows which files directly or indirectly include this file:

### Classes

- struct BayesicSpace::IndexedPairSimilarity

  *Jaccard value with indexes.*
- struct BayesicSpace::IndexedPairLD

  *LD value with indexes.*
- class BayesicSpace::GenoTableBin

  *Class to store binary compressed genotype tables.*
- class BayesicSpace::GenoTableHash

  *Class to store compressed genotype tables.*

### 5.2.1 Detailed Description

Summarize variant tables by hashing.

**Author**

Anthony J. Greenberg

**Copyright**

Copyright (c) 2021 Anthony J. Greenberg

**Version**

0.5

Definitions and interface documentation for classes that take binary variant files and generate lossy summaries with hashing.

## 5.3 gvarHash.hpp

Go to the documentation of this file.

```
00001 /*
00002  * Copyright (c) 2021 Anthony J. Greenberg
00003  *
00004  * Redistribution and use in source and binary forms, with or without modification, are permitted provided
      that the following conditions are met:
00005  *
00006  * 1. Redistributions of source code must retain the above copyright notice, this list of conditions and
      the following disclaimer.
00007  *
00008  * 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions
      and the following disclaimer in the documentation and/or other materials provided with the distribution.
00009  *
00010  * 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or
      promote products derived from this software without specific prior written permission.
00011  *
00012  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED
      WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
00013  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
      EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
00014  * BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
      (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
00015  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
      AND ON ANY THEORY OF LIABILITY, WHETHER
00016  * IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
      THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
00017  * THE POSSIBILITY OF SUCH DAMAGE.
00018  */
00019
00021
00030 #pragma once
00031
00032 #include <cstddef>
00033 #include <fstream>
00034 #include <vector>
00035 #include <array>
00036 #include <unordered_map>
00037 #include <utility>        // for std::pair
00038 #include <string>
00039 #include <thread>
00040 #include <mutex>
```

```
00041
00042 #include "random.hpp"
00043
00044 namespace BayesicSpace {
00045     struct IndexedPairSimilarity;
00046     class GenoTableBin;
00047     class GenoTableHash;
00048
00053     struct IndexedPairSimilarity {
00054         float similarityValue;
00055         uint32_t element1ind;
00056         uint32_t element2ind;
00057         uint32_t groupID;
00058     };
00059
00064     struct IndexedPairLD {
00065         float jaccard;
00066         float rSq;
00067         uint32_t element1ind;
00068         uint32_t element2ind;
00069     };
00070
00077     class GenoTableBin {
00078     public:
00080         GenoTableBin() : nIndividuals_{0}, nLoci_{0}, binLocusSize_{0}, nThreads_{1} {};
00091         GenoTableBin(const std::string &inputFileName, const size_t &nIndividuals, std::string
      logFileName) : GenoTableBin( inputFileName, nIndividuals, std::move(logFileName),
      std::thread::hardware_concurrency() ) {};
00104         GenoTableBin(const std::string &inputFileName, const size_t &nIndividuals, std::string
      logFileName, const size_t &nThreads);
00117         GenoTableBin(const std::vector<int> &maCounts, const size_t &nIndividuals, std::string
      logFileName) : GenoTableBin( maCounts, nIndividuals, std::move(logFileName),
      std::thread::hardware_concurrency() ) {};
00132         GenoTableBin(const std::vector<int> &maCounts, const size_t &nIndividuals, std::string
      logFileName, const size_t &nThreads);
00133
00135         GenoTableBin(const GenoTableBin &toCopy) = delete;
00137         GenoTableBin operator=(const GenoTableBin &toCopy) = delete;
00142         GenoTableBin(GenoTableBin &&toMove) noexcept;
00148         GenoTableBin& operator=(GenoTableBin &&toMove) noexcept;
00150         ~GenoTableBin() = default;
00151
00158         void saveGenoBinary(const std::string &outFileName) const;
00169         void allJaccardLD(const std::string &ldFileName) const;
00180         void allJaccardLD(const std::string &bimFileName, const std::string &ldFileName) const;
00185         void saveLogFile() const;
00186     private:
00191         std::vector<uint8_t> binGenotypes_;
00193         size_t nIndividuals_;
00195         size_t nLoci_;
00197         size_t binLocusSize_;
00199         size_t nThreads_;
00201         RanDraw rng_;
00203         mutable std::mutex mtx_;
00205         static const size_t nMagicBytes_;
00207         static const uint8_t oneBit_;
00209         static const uint8_t byteSize_;
00211         static const uint8_t bedGenoPerByte_;
00213         static const uint8_t llWordSize_;
00215         static const size_t maxNlocusPairs_;
00217         mutable std::string logMessages_;
00219         std::string logFileName_;
00229         void bed2binBlk_(const std::vector<char> &bedData, const std::pair<size_t, size_t>
      &bedLocusIndRange, const size_t &firstLocusInd, const size_t &bedLocusLength);
00240         size_t bed2binThreaded_(const std::vector<char> &bedData, const std::vector< std::pair<size_t,
      size_t> > &threadRanges, const size_t &firstLocusInd, const size_t &bedLocusLength);
00249         void mac2binBlk_(const std::vector<int> &macData, const std::pair<size_t, size_t> &locusIndRange,
      const size_t &randVecLen);
00256         void jaccardBlock_(const std::pair<size_t, size_t> &blockVecRange, const size_t &blockStartAll,
      std::vector<IndexedPairLD> &ldVec) const;
00264         size_t jaccardThreaded_(const std::vector< std::pair<size_t, size_t> > &pairIndRanges, const
      size_t &blockStartAll, std::vector<IndexedPairLD> &ldVec) const;
00265     };
00272     class GenoTableHash {
00273     public:
00275         GenoTableHash() : nIndividuals_{0}, kSketches_{0}, fSketches_{0.0}, sketchSize_{0}, nLoci_{0},
      locusSize_{0}, nFullWordBytes_{0}, nThreads_{1} {};
00292         GenoTableHash(const std::string &inputFileName, const size_t &nIndividuals, const size_t
      &kSketches, const size_t &nThreads, std::string logFileName);
00308         GenoTableHash(const std::string &inputFileName, const size_t &nIndividuals, const size_t
      &kSketches, std::string logFileName) :
```

```
00309                           GenoTableHash( inputFileName, nIndividuals, kSketches,
    std::thread::hardware_concurrency(), std::move(logFileName) ) {};
00328         GenoTableHash(const std::vector<int> &maCounts, const size_t &nIndividuals, const size_t
    &kSketches, const size_t &nThreads, std::string logFileName);
00344         GenoTableHash(const std::vector<int> &maCounts, const size_t &nIndividuals, const size_t
    &kSketches, std::string logFileName) :
00345                 GenoTableHash( maCounts, nIndividuals, kSketches, std::thread::hardware_concurrency(),
    std::move(logFileName) ) {};
00346
00348         GenoTableHash(const GenoTableHash &toCopy) = delete;
00350         GenoTableHash operator=(const GenoTableHash &toCopy) = delete;
00355         GenoTableHash(GenoTableHash &&toMove) noexcept;
00361         GenoTableHash& operator=(GenoTableHash &&toMove) noexcept;
00363         ~GenoTableHash() = default;
00364
00374         void allHashLD(const std::string &ldFileName) const;
00385         void allHashLD(const std::string &bimFileName,  const std::string &ldFileName) const;
00397         std::vector< std::vector<uint32_t> > makeLDgroups(const size_t &nRowsPerBand) const;
00405         void makeLDgroups(const size_t &nRowsPerBand, const std::string &outFileName) const;
00414         void makeLDgroups(const size_t &nRowsPerBand, const std::string &bimFileName, const std::string
    &outFileName) const;
00423         void ldInGroups(const size_t &nRowsPerBand, const std::string &outFileName) const;
00433         void ldInGroups(const size_t &nRowsPerBand, const std::string &bimFileName, const std::string
    &outFileName) const;
00438         void saveLogFile() const;
00439     private:
00444         std::vector<uint16_t> sketches_;
00446         size_t nIndividuals_;
00448         size_t kSketches_;
00450         float fSketches_;
00452         size_t sketchSize_;
00454         size_t nLoci_;
00456         size_t locusSize_;
00458         size_t nFullWordBytes_;
00460         size_t nThreads_;
00462         mutable RanDraw rng_;
00464         mutable std::mutex mtx_;
00466         mutable std::string logMessages_;
00468         std::string logFileName_;
00470         static const size_t maxPairs_;
00472         static const size_t nMagicBytes_;
00474         static const uint8_t oneBit_;
00476         static const uint8_t byteSize_;
00478         static const uint8_t bedGenoPerByte_;
00480         static const uint8_t llWordSize_;
00482         static const uint64_t roundMask_;
00484         static const uint64_t allBitsSet_;
00486         static const size_t wordSizeInBits_;
00488         static const uint16_t emptyBinToken_;
00498         void permuteBits_(const std::vector<size_t> &permutationIdx, std::vector<uint8_t> &binLocus)
    const;
00509         void locusOPH_(const size_t &locusInd, const std::vector<size_t> &permutation,
    std::vector<uint32_t> &seeds, std::vector<uint8_t> &binLocus);
00522         void bed2ophBlk_(const std::vector<char> &bedData, const std::pair<size_t, size_t>
    &bedLocusIndRange, const size_t &firstLocusInd,
00523                          const size_t &bedLocusLength, const std::vector<size_t> &permutation,
    std::vector< std::pair<size_t, size_t> > &padIndiv, std::vector<uint32_t> &seeds);
00537         size_t bed2ophThreaded_(const std::vector<char> &bedData, const std::vector< std::pair<size_t,
    size_t> > &threadRanges, const size_t &firstLocusInd,
00538                          const size_t &bedLocusLength, const std::vector<size_t> &permutation,
    std::vector< std::pair<size_t, size_t> > &padIndiv, std::vector<uint32_t> &seeds);
00551         void mac2ophBlk_(const std::vector<int> &macData, const size_t &startLocusInd, const size_t
    &endLocusInd, const size_t &randVecLen, const std::vector<size_t> &permutation, std::vector<uint32_t>
    &seeds);
00561         void hashJacBlock_(const std::pair<size_t, size_t> &blockRange, const size_t &blockStartAll,
    std::vector<IndexedPairSimilarity> &hashJacVec) const;
00572         void hashJacBlock_(const size_t &blockStartVec, const size_t &blockEndVec,
    std::vector<IndexedPairSimilarity> &indexedJacc) const;
00582         void hashJacBlock_(const std::vector<IndexedPairSimilarity>::iterator blockStart, const
    std::vector<IndexedPairSimilarity>::iterator blockEnd) const;
00593         size_t hashJacThreaded_(const std::vector< std::pair<size_t, size_t> > &threadRanges, const size_t
    &blockStartAll, std::vector<IndexedPairSimilarity> &hashJacVec) const;
00603         void hashJacThreaded_(const std::vector< std::pair<size_t, size_t> > &threadRanges,
    std::vector<IndexedPairSimilarity> &hashJacVec) const;
00604     };
00605 }
00606
```

## 5.4 include/vashFunctions.hpp File Reference

Auxiliary functions for variant hashing.

```
#include <cstdint>
#include <cstddef>
#include <vector>
#include <array>
#include <unordered_map>
#include <fstream>
#include "random.hpp"
#include "gvarHash.hpp"
```

Include dependency graph for vashFunctions.hpp: This graph shows which files directly or indirectly include this file:

### Functions

- uint16_t BayesicSpace::countSetBits (uint16_t inVal)

  *Count set bits in a 16-bit word.*
- uint64_t BayesicSpace::countSetBits (const std::vector< uint8_t > &inVec)

  *Count set bits in a vector.*
- uint64_t BayesicSpace::countSetBits (const std::vector< uint8_t > &inVec, const size_t &start, const size_t &length)

  *Count set bits in a range within a vector.*
- size_t BayesicSpace::getAvailableRAM ()

  *Get available RAM.*
- uint32_t BayesicSpace::murMurHashMixer (const size_t &key, const uint32_t &seed)

  *MurMurHash mixer module of an index value.*
- uint32_t BayesicSpace::murMurHashFinalizer (const uint32_t &inputHash)

  *MurMurHash finalizer.*
- uint32_t BayesicSpace::murMurHash (const size_t &key, const uint32_t &seed)

  *MurMurHash of an index value.*
- uint32_t BayesicSpace::murMurHash (const std::vector< size_t > &key, const uint32_t &seed)

  *MurMurHash of a vector of indexes.*
- uint32_t BayesicSpace::murMurHash (const size_t &start, const size_t &length, const std::vector< uint16_t > &key, const uint32_t &seed)

  *MurMurHash of a vector of indexes.*
- void BayesicSpace::testBedMagicBytes (std::array< char, 3 > &bytesToTest)

  *Test .bed magic bytes.*
- std::vector< std::pair< size_t, size_t > > BayesicSpace::makeThreadRanges (const size_t &nThreads, const size_t &nElementsPerThread)

  *Build thread ranges.*
- void BayesicSpace::binarizeBedLocus (const size_t &bedIdx, const size_t &bedLocusSize, const std::vector< char > &bedLocus, const size_t &nIndividuals, RanDraw &prng, const size_t &binIdx, const size_t &binLocusSize, std::vector< uint8_t > &binLocus)

  *Convert a locus from _.bed_ to binary format.*
- std::vector< IndexedPairSimilarity > BayesicSpace::vectorizeGroups (const uint32_t &firstGrpIdx, const std::vector< std::vector< uint32_t > >::const_iterator grpBlockStart, const std::vector< std::vector< uint32_t > >::const_iterator grpBlockEnd)

*Create pair vector from groups.*

- void BayesicSpace::saveValues (const std::vector< float > &inVec, std::fstream &outputStream)

  *Save values.*
- void BayesicSpace::saveValues (const std::vector< IndexedPairSimilarity > &inVec, std::fstream &outputStream)

  *Save indexed values.*
- void BayesicSpace::saveValues (const std::vector< IndexedPairSimilarity > &inVec, const std::vector< std::string > &locusNames, std::fstream &outputStream)

  *Save named values.*
- void BayesicSpace::saveValues (const std::vector< IndexedPairLD > &inVec, std::fstream &outputStream)

  *Save indexed LD values.*
- void BayesicSpace::saveValues (const std::vector< IndexedPairLD > &inVec, const std::vector< std::string > &locusNames, std::fstream &outputStream)

  *Save named LD values.*
- std::vector< std::string > BayesicSpace::getLocusNames (const std::string &bimFileName)

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

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

  *Extract parameters from parsed command line interface flags.*

### 5.4.1 Detailed Description

Auxiliary functions for variant hashing.

**Author**

Anthony J. Greenberg

**Copyright**

Copyright (c) 2023 Anthony J. Greenberg

**Version**

0.5

Definitions of class-external functions needed by hashing classes.

### 5.4.2 Function Documentation

#### 5.4.2.1 binarizeBedLocus()

```
void BayesicSpace::binarizeBedLocus (
            const size_t & bedIdx,
            const size_t & bedLocusSize,
            const std::vector< char > & bedLocus,
            const size_t & nIndividuals,
            RanDraw & prng,
            const size_t & binIdx,
            const size_t & binLocusSize,
            std::vector< uint8_t > & binLocus )
```

Convert a locus from _.bed_ to binary format.

Convert the _.bed_ two-bit format to one-bit binary.

- 00 (homozygous first _.bim_ allele) becomes 0 if it is the major allele, 1 otherwise
- 11 (homozygous second _.bim_ allele) becomes 1 if it is the minor allele, 0 otherwise
- 01 (missing genotype) becomes 0
- 10 (heterozygous) becomes 1 with probability 0.5

If the number of individuals is not divisible by eight, the last binary byte is padded with 0s.

**Parameters**

| | | |
|---|---|---|
| in | *bedIdx* | index of the first byte in the `bedLocus` vector |
| in | *bedLocusSize* | number of bytes in the _.bed_ locus |
| in | *bedLocus* | vector of _.bed_ format bytes |
| in | *nIndividuals* | number of individuals |
| in,out | *prng* | (pseudo-)random number generator (for resolving heterozygotes) |
| in | *binIdx* | index of the first binary byte |
| in | *binLocusSize* | number of bytes in the binary locus |
| out | *binLocus* | vector of binary format bytes |

#### 5.4.2.2 countSetBits() [1/3]

```
uint64_t BayesicSpace::countSetBits (
            const std::vector< uint8_t > & inVec )
```

Count set bits in a vector.

Counting the set bits in a vector of bytes using Karnigan's method.

**Parameters**

| | | |
|---|---|---|
| in | *inVec* | input vector |

**Returns**

number of bits set

#### 5.4.2.3 countSetBits() [2/3]

```
uint64_t BayesicSpace::countSetBits (
            const std::vector< uint8_t > & inVec,
            const size_t & start,
            const size_t & length )
```

Count set bits in a range within a vector.

Counting the set bits in a range within a vector of bytes using Karnigan's method.

**Parameters**

| | | |
|---|---|---|
| in | *inVec* | input vector |
| in | *start* | staring index |
| in | *length* | number of bytes to process |

**Returns**

number of bits set

#### 5.4.2.4 countSetBits() [3/3]

```
uint16_t BayesicSpace::countSetBits (
            uint16_t inVal )
```

Count set bits in a 16-bit word.

Counting the set bits using Karnigan's method. Passing by value to modify the copy and also because the address is much bigger than 16 bits.

**Parameters**

| | | |
|---|---|---|
| in | *inVal* | input value |

**Returns**

number of bits set

#### 5.4.2.5 extractCLinfo()

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

Extract parameters from parsed command line interface flags.

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

**Parameters**

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

#### 5.4.2.6 getAvailableRAM()

```
size_t BayesicSpace::getAvailableRAM ( )
```

Get available RAM.

Estimates available RAM. If `procfs` is mounted, uses information from there. Otherwise, sets available RAM to 2 GiB.

**Returns**

estimated available RAM in bytes

#### 5.4.2.7 getLocusNames()

```
std::vector< std::string > BayesicSpace::getLocusNames (
            const std::string & bimFileName )
```

Extract locus names.

Extract locus names from a _.bim_ file.

**Parameters**

| | | |
|---|---|---|
| in | *bimFileName* | _.bim_ file name |

**Returns**

vector of locus names

#### 5.4.2.8 makeThreadRanges()

```
std::vector< std::pair< size_t, size_t > > BayesicSpace::makeThreadRanges (
            const size_t & nThreads,
            const size_t & nElementsPerThread )
```

Build thread ranges.

Build index ranges to use within each thread.

**Parameters**

| | | |
|---|---|---|
| in | *nThreads* | number of threads |
| in | *nElementsPerThread* | number of elements per thread |

**Returns**

vector of index ranges

#### 5.4.2.9 murMurHash() [1/3]

```
uint32_t BayesicSpace::murMurHash (
            const size_t & key,
            const uint32_t & seed )
```

MurMurHash of an index value.

Generates a 32-bit hash of an index value using the MurMurHash3 algorithm.

**Parameters**

| | | |
|---|---|---|
| in | *key* | the key to be hashed |
| in | *seed* | the seed |

**Returns**

the hash value

#### 5.4.2.10 murMurHash() [2/3]

```
uint32_t BayesicSpace::murMurHash (
            const size_t & start,
            const size_t & length,
            const std::vector< uint16_t > & key,
            const uint32_t & seed )
```

MurMurHash of a vector of indexes.

Generates a 32-bit hash of a vector of `uint16_t` values using the MurMurHash3 algorithm.

**Parameters**

| | | |
|---|---|---|
| in | *start* | start index |
| in | *length* | number of elements to hash |
| in | *key* | the vector to be hashed |
| in | *seed* | the hash seed |

**Returns**

hash value

#### 5.4.2.11 murMurHash() [3/3]

```
uint32_t BayesicSpace::murMurHash (
            const std::vector< size_t > & key,
            const uint32_t & seed )
```

MurMurHash of a vector of indexes.

Generates a 32-bit hash of an index value using the MurMurHash3 algorithm.

**Parameters**

| | | |
|---|---|---|
| in | *key* | the key vector to be hashed |
| in | *seed* | the seed |

**Returns**

the hash value

#### 5.4.2.12 murMurHashFinalizer()

```
uint32_t BayesicSpace::murMurHashFinalizer (
            const uint32_t & inputHash )
```

MurMurHash finalizer.

MurMurHash3 finalizer for a hash value.

**Parameters**

| | | |
|---|---|---|
| in | *inputHash* | input unfinlized hash value |

**Returns**

finalized hash value

#### 5.4.2.13 murMurHashMixer()

```
uint32_t BayesicSpace::murMurHashMixer (
            const size_t & key,
            const uint32_t & seed )
```

MurMurHash mixer module of an index value.

Generates a 32-bit an unfinalized hash of an index value using the MurMurHash3 algorithm.

**Parameters**

| | | |
|---|---|---|
| in | *key* | the key to be hashed |
| in | *seed* | the seed |

**Returns**

the hash value

#### 5.4.2.14 parseCL()

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

Command line parser.

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

**Parameters**

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

#### 5.4.2.15 saveValues() [1/5]

```
void BayesicSpace::saveValues (
            const std::vector< float > & inVec,
            std::fstream & outputStream )
```

Save values.

Saves each value from the vector to the provided `fstream` with space as the delimiter.

**Parameters**

| | | |
|---|---|---|
| in | *inVec* | vector of floats to save |
| in,out | *outputStream* | `fstream` to save to |

#### 5.4.2.16 saveValues() [2/5]

```
void BayesicSpace::saveValues (
            const std::vector< IndexedPairLD > & inVec,
            const std::vector< std::string > & locusNames,
            std::fstream & outputStream )
```

Save named LD values.

Saves LD estimates (Jaccard and $r^2$) with locus pair names, tab delimited.

**Parameters**

| | | |
|---|---|---|
| in | *inVec* | vector of indexed LD values to save |
| in | *locusNames* | vector of locus names |
| in,out | *outputStream* | fstream to save to |

#### 5.4.2.17 saveValues() [3/5]

```
void BayesicSpace::saveValues (
            const std::vector< IndexedPairLD > & inVec,
            std::fstream & outputStream )
```

Save indexed LD values.

Saves LD estimates (Jaccard and $r^2$) with locus pair indexes, tab delimited.

**Parameters**

| | | |
|---|---|---|
| in | *inVec* | vector of indexed LD values to save |
| in,out | *outputStream* | fstream to save to |

#### 5.4.2.18 saveValues() [4/5]

```
void BayesicSpace::saveValues (
            const std::vector< IndexedPairSimilarity > & inVec,
            const std::vector< std::string > & locusNames,
            std::fstream & outputStream )
```

Save named values.

Saves each value with group IDs and names of locus pairs, tab delimited.

**Parameters**

| | | |
|---|---|---|
| in | *inVec* | vector of indexed floats to save |
| in | *locusNames* | vector of locus names |
| in,out | *outputStream* | fstream to save to |

#### 5.4.2.19 saveValues() [5/5]

```
void BayesicSpace::saveValues (
            const std::vector< IndexedPairSimilarity > & inVec,
            std::fstream & outputStream )
```

Save indexed values.

Saves each value with group IDs and the pair of indexes, tab delimited.

**Parameters**

| | | |
|---|---|---|
| in | *inVec* | vector of indexed floats to save |
| in,out | *outputStream* | fstream to save to |

#### 5.4.2.20 testBedMagicBytes()

```
void BayesicSpace::testBedMagicBytes (
            std::array< char, 3 > & bytesToTest )
```

Test .bed magic bytes.

Throws if one of the input bytes does not match the three magic values in plink .bed files.

**Parameters**

| | | |
|---|---|---|
| in | *bytesToTest* | the byte set to test |

#### 5.4.2.21 vectorizeGroups()

```
std::vector< IndexedPairSimilarity > BayesicSpace::vectorizeGroups (
            const uint32_t & firstGrpIdx,
            const std::vector< std::vector< uint32_t > >::const_iterator grpBlockStart,
            const std::vector< std::vector< uint32_t > >::const_iterator grpBlockEnd )
```

Create pair vector from groups.

Create a vector of paired indexes within provided groups, in preparation for estimating Jaccard similarities.

**Parameters**

| | | |
|---|---|---|
| in | *firstGrpIdx* | index of the first group in the range |
| in | *grpBlockStart* | iterator of the first group in the block |
| in | *grpBlockEnd* | iterator of (one past the) end group in the block |

**Returns**

vector of pair indexes with group IDs

## 5.5 vashFunctions.hpp

Go to the documentation of this file.

```
00001 /*
00002  * Copyright (c) 2023 Anthony J. Greenberg
00003  *
00004  * Redistribution and use in source and binary forms, with or without modification, are permitted provided
      that the following conditions are met:
00005  *
00006  * 1. Redistributions of source code must retain the above copyright notice, this list of conditions and
      the following disclaimer.
00007  *
00008  * 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions
      and the following disclaimer in the documentation and/or other materials provided with the distribution.
00009  *
00010  * 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or
      promote products derived from this software without specific prior written permission.
00011  *
00012  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED
      WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
00013  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
      EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
00014  * BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
      (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
00015  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
      AND ON ANY THEORY OF LIABILITY, WHETHER
00016  * IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
      THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
00017  * THE POSSIBILITY OF SUCH DAMAGE.
00018  */
00019
00021
00030 #pragma once
00031
00032 #include <cstdint>
00033 #include <cstddef>
00034 #include <vector>
00035 #include <array>
00036 #include <unordered_map>
00037 #include <fstream>
00038
00039 #include "random.hpp"
00040 #include "gvarHash.hpp"
00041
00042 namespace BayesicSpace {
00050     uint16_t countSetBits(uint16_t inVal);
00058     uint64_t countSetBits(const std::vector<uint8_t> &inVec);
00068     uint64_t countSetBits(const std::vector<uint8_t> &inVec, const size_t &start, const size_t &length);
00075     size_t getAvailableRAM();
00085     uint32_t murMurHashMixer(const size_t &key, const uint32_t &seed);
00094     uint32_t murMurHashFinalizer(const uint32_t &inputHash);
00104     uint32_t murMurHash(const size_t &key, const uint32_t &seed);
00114     uint32_t murMurHash(const std::vector<size_t> &key, const uint32_t &seed);
00126     uint32_t murMurHash(const size_t &start, const size_t &length, const std::vector<uint16_t> &key, const
      uint32_t &seed);
00133     void testBedMagicBytes(std::array<char, 3> &bytesToTest);
00142     std::vector< std::pair<size_t, size_t> > makeThreadRanges(const size_t &nThreads, const size_t
      &nElementsPerThread);
00163     void binarizeBedLocus(const size_t &bedIdx, const size_t &bedLocusSize, const std::vector<char>
      &bedLocus, const size_t &nIndividuals,
00164                                                     RanDraw &prng, const size_t &binIdx, const size_t
      &binLocusSize, std::vector<uint8_t> &binLocus);
00174     std::vector<IndexedPairSimilarity> vectorizeGroups(const uint32_t &firstGrpIdx,
00175                 const std::vector< std::vector<uint32_t> >::const_iterator grpBlockStart, const
      std::vector< std::vector<uint32_t> >::const_iterator grpBlockEnd);
00183     void saveValues(const std::vector<float> &inVec, std::fstream &outputStream);
00191     void saveValues(const std::vector<IndexedPairSimilarity> &inVec, std::fstream &outputStream);
00200     void saveValues(const std::vector<IndexedPairSimilarity> &inVec, const std::vector<std::string>
      &locusNames, std::fstream &outputStream);
00208     void saveValues(const std::vector<IndexedPairLD> &inVec, std::fstream &outputStream);
00217     void saveValues(const std::vector<IndexedPairLD> &inVec, const std::vector<std::string> &locusNames,
      std::fstream &outputStream);
```

```
00225     std::vector<std::string> getLocusNames(const std::string &bimFileName);
00234     void parseCL(int &argc, char **argv, std::unordered_map<std::string, std::string> &cli);
00243     void extractCLinfo(const std::unordered_map<std::string, std::string> &parsedCLI,
      std::unordered_map<std::string, int> &intVariables, std::unordered_map<std::string, std::string>
      &stringVariables);
00244 }
```

## 5.6 src/gvarHash.cpp File Reference

Summarize variant tables by hashing.

```
#include <chrono>
#include <ctime>
#include <iomanip>
#include <cstring>
#include <cassert>
#include <string>
#include <sstream>
#include <fstream>
#include <vector>
#include <unordered_map>
#include <array>
#include <utility>
#include <iterator>
#include <algorithm>
#include <numeric>
#include <functional>
#include <limits>
#include <future>
#include <thread>
#include <immintrin.h>
#include "gvarHash.hpp"
#include "vashFunctions.hpp"
#include "random.hpp"
```

Include dependency graph for gvarHash.cpp:

## 5.7 src/vashFunctions.cpp File Reference

Auxiliary functions for variant hashing.

```
#include <cstring>
#include <cassert>
#include <string>
#include <sstream>
#include <fstream>
#include <vector>
#include <array>
#include <unordered_map>
#include <utility>
#include <limits>
#include <immintrin.h>
#include "gvarHash.hpp"
#include "vashFunctions.hpp"
```

Include dependency graph for vashFunctions.cpp:

### 5.7.1 Detailed Description

Auxiliary functions for variant hashing.

**Author**

Anthony J. Greenberg

**Copyright**



**Version**

0.5

Implementation of class-external functions needed by hashing classes.