Utilities

Generated by Doxygen 1.9.1

Chapter 1

Overview

A collection of numerical methods and data structures I most often use in my projects. The `utilities` module contains various basic functions and algorithms. The `random` module implements random number generation, using hardware random numbers if available for the CPU, and sampling from various distributions. The `index` module is a class that can be used to relate elements to groups they belong to, similar to the factor in R. I have been using these utilities in several projects, so they are fairly well tested.

1.1 Including in your project

If you want to try these out, you can include them in your project by running

```
git submodule add https://github.com/tonymugen/bayesicUtilities [optional local name]
```

Interface documentation is `available here`.

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

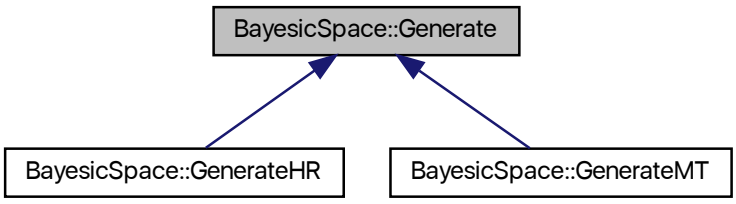
Class Documentation

5.1 BayesicSpace::Generate Class Reference

Abstract base random number class.

`#include <random.hpp>`

Inheritance diagram for BayesicSpace::Generate:



Public Member Functions

- virtual ~Generate ()

  Destructor.
- virtual uint64_t ranInt () const =0

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

Protected Member Functions

- Generate ()

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

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

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

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

  Protected move assignment.

5.1.1 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.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Generate() [1/2]

```
BayesicSpace::Generate::Generate (
            const Generate & old )  [protected], [default]
```

Protected copy constructor.

Parameters

in	*old*	object to copy

5.1.2.2 Generate() [2/2]

```
BayesicSpace::Generate::Generate (
            Generate && old )  [protected], [default]
```

Protected move constructor.

Parameters

in	*old*	object to move

5.1.3 Member Function Documentation

5.1.3.1 operator=() [1/2]

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

Protected copy assignment operator.

Parameters

in	*old*	object to copy

5.1.3.2 operator=() [2/2]

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

Protected move assignment.

Parameters

in	*old*	object to move

5.1.3.3 ranInt()

```
virtual uint64_t BayesicSpace::Generate::ranInt ( ) const  [pure virtual]
```

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

Returns

random or pseudo-random 64-bit unsigned integer

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

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

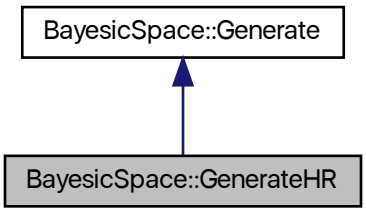
- random.hpp

5.2 BayesicSpace::GenerateHR Class Reference

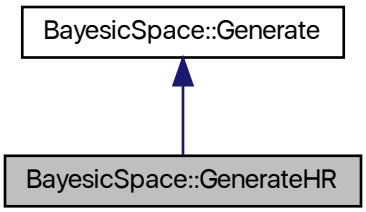
Hardware random number generating class.

`#include <random.hpp>`

Inheritance diagram for BayesicSpace::GenerateHR:



Collaboration diagram for BayesicSpace::GenerateHR:



Public Member Functions

- GenerateHR ()

  Default constructor.
- ~GenerateHR ()

  Destructor.
- GenerateHR (const GenerateHR &old)=default

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

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

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

  Move assignment.
- uint64_t ranInt () const override

  Generate a random 64-bit unsigned integer.

Additional Inherited Members

5.2.1 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`.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 GenerateHR() [1/2]

```
BayesicSpace::GenerateHR::GenerateHR (
            const GenerateHR & old )  [default]
```

Copy constructor.

Parameters

in	*old*	object to copy

5.2.2.2 GenerateHR() [2/2]

```
BayesicSpace::GenerateHR::GenerateHR (
            GenerateHR && old )  [default]
```

Move constructor.

Parameters

in	*old*	object to move

5.2.3 Member Function Documentation

5.2.3.1 operator=() [1/2]

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

Copy assignment operator.

Parameters

in	*old*	object to copy

5.2.3.2 operator=() [2/2]

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

Move assignment.

Parameters

in	*old*	object to move

5.2.3.3 ranInt()

```
uint64_t GenerateHR::ranInt ( ) const  [override], [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 BayesicSpace::Generate.

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

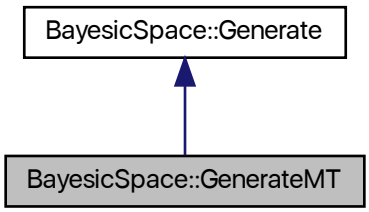
- random.hpp
- random.cpp

5.3 BayesicSpace::GenerateMT Class Reference

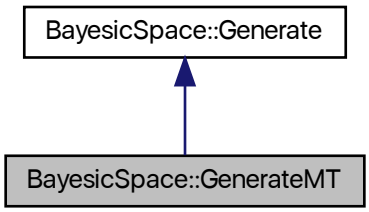
Pseudo-random number generator.

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

Inheritance diagram for BayesicSpace::GenerateMT:



Collaboration diagram for BayesicSpace::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.
- uint64_t ranInt () const override

  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 uint16_t n_ = 312
  Degree of recurrence.
- static const uint16_t 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 uint32_t l_ = 43
  Tempering shift.
- static const uint32_t s_ = 17
  Tempering shift.
- static const uint32_t t_ = 37
  Tempering shift.
- static const uint32_t 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

5.3.1 Detailed Description

Pseudo-random number generator.

An implementaiton of the 64-bit MT19937 ("Mersenne Twister") **[matsumoto98a]** pseudo-random number generator (PRNG). The constructor automatically seeds the PRNG. The implementation was guided by the reference code posted by the authors.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 GenerateMT() [1/3]

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

Default constructor.

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

5.3.2.2 GenerateMT() [2/3]

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

Copy constructor.

Parameters

in	*old*	object to copy

5.3.2.3 GenerateMT() [3/3]

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

Move constructor.

Parameters

in	*old*	object to move

5.3.3 Member Function Documentation

5.3.3.1 operator=() [1/2]

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

Copy assignment operator.

Parameters

in	*old*	object to copy

5.3.3.2 operator=() [2/2]

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

Move assignment.

Parameters

in	*old*	object to move

5.3.3.3 ranInt()

```
uint64_t GenerateMT::ranInt ( ) const  [override], [virtual]
```

Generate a pseudo-random 64-bit unsigned integer.

Returns

pseudo-random 64-bit unsigned integer

Implements BayesicSpace::Generate.

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

- random.hpp
- random.cpp

5.4 BayesicSpace::Index Class Reference

Group index.

```
#include <index.hpp>
```

Public Member Functions

- Index ()
  Default constructor.
- Index (const size_t &Ngroups)
  Group constructor.
- Index (const size_t *arr, const size_t &N)
  Array constructor.
- Index (const vector< size_t > &vec)
  Vector constructor.
- Index (const string &inFileName)
  File read constructor.
- Index (const Index &in)
  Copy constructor.
- Index & operator= (const Index &in)
  Copy assignment operator.
- Index (Index &&in)
  Move constructor.
- Index & operator= (Index &&in)
  Move assignment operator.
- ~Index ()
  Destructor.
- const vector< size_t > & operator[ ] (const size_t &i) const
  Vector subscript operator.
- size_t groupSize (const size_t &i) const
  Group size.
- size_t size () const
  Total sample size.
- size_t groupNumber () const
  Number of groups.
- size_t neGroupNumber () const
  Number of non-empty groups.
- size_t groupID (const size_t &ind) const
  Group ID.
- void update (const vector< size_t > &newVec)
  Update the index.

5.4.1 Detailed Description

Group index.

For each group, contains indexes of the lines that belong to it. Can also identify the group a given element belongs to. Group numbers need not be consecutive. Although group IDs are assumed to be base-0, everything should work even if they are not.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 Index() [1/6]

```
Index::Index (
            const size_t & Ngroups )
```

Group constructor.

Sets up empty groups.

Parameters

in	*Ngroups*	number of groups to set up

5.4.2.2 Index() [2/6]

```
Index::Index (
            const size_t * arr,
            const size_t & N )
```

Array constructor.

The input array has an element for each line, and the value of that element is the base-0 group ID (i.e., if line *n* is in the first group, then `arr[n] == 0`).

Parameters

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

5.4.2.3 Index() [3/6]

```
Index::Index (
            const vector< size_t > & vec )
```

Vector constructor.

The input vector has an element for each line, and the value of that element is the base-0 group ID (i.e., if line *n* is in the first group, then `vec[n] == 0`).

Parameters

in	*vec*	array of group IDs

5.4.2.4 Index() [4/6]

```
Index::Index (
            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 base-0 group ID. If the file cannot be opened, throws "Cannot open file file_name". If a negative group value is detected, thorws "Negative group ID".

Parameters

in	*inFileName*	input file name

5.4.2.5 Index() [5/6]

```
BayesicSpace::Index::Index (
            const Index & in )  [inline]
```

Copy constructor.

Parameters

in	*in*	Index to be copied

Returns

Index object

5.4.2.6 Index() [6/6]

```
BayesicSpace::Index::Index (
            Index && in )  [inline]
```

Move constructor.

Parameters

in	*in*	Index object to be moved

Returns

Index object

5.4.3 Member Function Documentation

5.4.3.1 groupID()

```
size_t BayesicSpace::Index::groupID (
            const size_t & ind ) const  [inline]
```

Group ID.

Returns the group ID for a given individual.

Parameters

in	*ind*	index of an individual

Returns

group ID

5.4.3.2 groupNumber()

```
size_t BayesicSpace::Index::groupNumber ( ) const  [inline]
```

Number of groups.

Returns

number of groups

5.4.3.3 groupSize()

```
size_t BayesicSpace::Index::groupSize (
            const size_t & i ) const  [inline]
```

Group size.

Parameters

in	*i*	group index

Returns

size of the _i_th group

5.4.3.4 neGroupNumber()

```
size_t Index::neGroupNumber ( ) const
```

Number of non-empty groups.

Returns

number of non-empty groups

5.4.3.5 operator=() [1/2]

```
Index & Index::operator= (
            const Index & in )
```

Copy assignment operator.

Parameters

in	*in*	object to be copied

Returns

an Index object

5.4.3.6 operator=() [2/2]

```
Index & Index::operator= (
            Index && in )
```

Move assignment operator.

Parameters

in	*in*	object to be moved

Returns

an Index object

5.4.3.7 operator[]()

```
const vector<size_t>& BayesicSpace::Index::operator[] (
            const size_t & i ) const  [inline]
```

Vector subscript operator.

Returns the index of group *i*.

Parameters

in	*i*	group index

Returns

index of line IDs

5.4.3.8 size()

```
size_t BayesicSpace::Index::size ( ) const  [inline]
```

Total sample size.

Returns

total sample size

5.4.3.9 update()

```
void Index::update (
            const vector< size_t > & newVec )
```

Update the index.

Updates the groups with a new index. If a group is not present in the new vector, it is left empty but still exists.

Parameters

in	*newVec*	new vector of group IDs

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

- index.hpp
- index.cpp

5.5 BayesicSpace::NumerUtil Class Reference

Numerical utilities collection.

`#include <utilities.hpp>`

Public Member Functions

- void swapXOR (size_t &i, size_t &j) const
  Swap two `size_t` values.
- double logit (const double &p) const
  Logit function.
- double logistic (const double &x) const
  Logistic function.
- double lnGamma (const double &x) const
  Logarithm of the Gamma function.
- double digamma (const double &x) const
  Digamma function.
- double dotProd (const vector< double > &v) const
  Vector self-dot-product.
- double dotProd (const vector< double > &v1, const vector< double > &v2) const
  Dot-product of two vectors.
- void updateWeightedMean (const double &xn, const double &wn, double &mu, double &w) const
  Weighted mean update.
- double mean (const double arr[ ], const size_t &len)
  Mean of an array.

5.5.1 Detailed Description

Numerical utilities collection.

Implements numerical functions for use throughout the project.

5.5.2 Member Function Documentation

5.5.2.1 digamma()

```
double NumerUtil::digamma (
            const double & x ) const
```

Digamma function.

Defined only for $x > 0$, will return *NaN* otherwise. Adopted from the `dpsifn` function in R.

Parameters

in	*x*	function argument (must be positive)

Returns

value of the digamma function

5.5.2.2 dotProd() [1/2]

```
double NumerUtil::dotProd (
            const vector< double > & v ) const
```

Vector self-dot-product.

Parameters

in	*v*	vector

Returns

dot-product value

5.5.2.3 dotProd() [2/2]

```
double NumerUtil::dotProd (
            const vector< double > & v1,
            const vector< double > & v2 ) const
```

Dot-product of two vectors.

Parameters

in	*v1*	vector 1
in	*v2*	vector 2

Returns

dot-product value

5.5.2.4 lnGamma()

```
double NumerUtil::lnGamma (
            const double & x ) const
```

Logarithm of the Gamma function.

The log of the $\Gamma(x)$ function. Implementing the Lanczos algorithm following Numerical Recipes in C++.

Parameters

in	*x*	value

Returns

$\log \Gamma(x)$

5.5.2.5 logistic()

```
double NumerUtil::logistic (
            const double & x ) const
```

Logistic function.

There is a guard against under- and overflow: the function returns 0.0 for $x \leq -35.0$ and 1.0 for $x \geq 35.0$.

Parameters

in	*x*	value to be projected to the (0, 1) interval

Returns

logistic transformation

5.5.2.6 logit()

```
double BayesicSpace::NumerUtil::logit (
            const double & p ) const  [inline]
```

Logit function.

Parameters

in	*p*	probability in the (0, 1) interval

Returns

logit transformation

5.5.2.7 mean()

```
double NumerUtil::mean (
            const double arr[],
            const size_t & len )
```

Mean of an array.

Uses the numerically stable recursive algorithm.

Parameters

in	*arr*	c-style array of values
in	*len*	array length

Returns

mean value

5.5.2.8 swapXOR()

```
void NumerUtil::swapXOR (
            size_t & i,
            size_t & j ) const
```

Swap two `size_t` values.

Uses the three XORs trick to swap two integers. Safe if the variables happen to refer to the same address.

Parameters

in,out	*i*	first integer
in,out	*j*	second integer

5.5.2.9 updateWeightedMean()

```
void NumerUtil::updateWeightedMean (
            const double & xn,
            const double & wn,
            double & mu,
            double & w ) const
```

Weighted mean update.

Takes the current weighted mean and updates using the new data point and weight. The formula is

$$\bar{\mu}_n = \frac{\bar{\mu}_{n-1} \sum_{i=1}^{n-1} w_i + w_n x_n}{\sum_{i=1}^{n-1} w_i + w_n}$$

Parameters

in	*xn*	new point x_n
in	*wn*	weight w_n
out	*mu*	new mean
out	*w*	new weight

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

- utilities.hpp
- utilities.cpp

5.6 BayesicSpace::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.
- string type () const

  Query RNG kind.
- uint64_t ranInt () const

  Generate random integer.
- uint64_t sampleInt (const uint64_t &max) const

  Sample and integer from the $[0, n)$ interval.
- uint64_t sampleInt (const uint64_t &min, const uint64_t &max) const

  Sample and integer from the $[m, n)$ interval.
- vector< uint64_t > shuffleUint (const uint64_t &N)

  Draw non-negative intergers in random order.
- double runif () const

  Generate a uniform deviate.
- double runifnz () const

  Generate a non-zero uniform deviate.
- double runifno () const

  Generate a non-one uniform deviate.
- double runifop () const

  Generate an open-interval uniform deviate.
- double rnorm () const

  A standard Gaussian deviate.
- double rnorm (const double &sigma) const

  A zero-mean Gaussian deviate.
- double rnorm (const double &mu, const double &sigma) const

  A Gaussian deviate.
- double rgamma (const double &alpha) const

  A standard Gamma deviate.
- double rgamma (const double &alpha, const double &beta) const

  A general Gamma deviate.
- void rdirichlet (const vector< double > &alpha, vector< double > &p) const

  A Dirichlet deviate.

- double rchisq (const double &nu) const

  A chi-square deviate.
- uint64_t vitterA (const double &n, const double &N) const

  Sample from Vitter's distribution, method A.
- uint64_t vitter (const double &n, const double &N) const

  Sample from Vitter's distribution, method D.

5.6.1 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").

5.6.2 Constructor & Destructor Documentation

5.6.2.1 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.

5.6.2.2 RanDraw() [2/3]

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

Copy constructor.

Parameters

in	*old*	pbject to be copied

5.6.2.3 RanDraw() [3/3]

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

Move constructor.

Parameters

in	*old*	pbject to be moved

5.6.3 Member Function Documentation

5.6.3.1 operator=() [1/2]

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

Copy assignment.

Parameters

in	*old*	pbject to be copied

5.6.3.2 operator=() [2/2]

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

Move assignment.

Parameters

in	*old*	pbject to be moved

5.6.3.3 ranInt()

```
uint64_t BayesicSpace::RanDraw::ranInt ( ) const  [inline]
```

Generate random integer.

Returns

An unsigned random 64-bit integer

5.6.3.4 rchisq()

```
double BayesicSpace::RanDraw::rchisq (
            const double & nu ) const  [inline]
```

A chi-square deviate.

Generates a χ^2 random variable with degrees of freedom $\nu > 0.0$.

Parameters

in	*nu*	degrees of freedom

Returns

a sample from the χ^2 distribution

5.6.3.5 rdirichlet()

```
void RanDraw::rdirichlet (
            const vector< double > & alpha,
            vector< double > & p ) const
```

A Dirichlet deviate.

Generates a vector of probabilities, given a vector of concetration parameters $\alpha_K > 0$.

Parameters

in	*alpha*	vector of concentration parameters
out	*p*	vector of probabilities, must be the same length as α.

5.6.3.6 rgamma() [1/2]

```
double RanDraw::rgamma (
            const double & alpha ) const
```

A standard Gamma deviate.

Generates a Gamma random variable with shape $\alpha > 0$ and standard scale $\beta = 1.0$. Implements the Marsaglia and Tsang (2000) method.

Parameters

in	*alpha*	shape parameter α

Returns

a sample from the standard Gamma distribution

5.6.3.7 rgamma() [2/2]

```
double BayesicSpace::RanDraw::rgamma (
            const double & alpha,
            const double & beta ) const  [inline]
```

A general Gamma deviate.

Generates a Gamma random variable with shape $\alpha > 0$ and scale $\beta > 0$.

Parameters

in	*alpha*	shape parameter α
in	*beta*	scale parameter β

Returns

a sample from the general Gamma distribution

5.6.3.8 rnorm() [1/3]

```
double RanDraw::rnorm ( ) const
```

A standard Gaussian deviate.

Generates a Gaussian random value with mean $\mu = 0.0$ and standard deviation $\sigma = 1.0$. Implemented using a version of the Marsaglia and Tsang (2000) ziggurat algorithm, modified according to suggestions in the GSL implementation of the function.

Returns

a sample from the standard Gaussian distribution

5.6.3.9 rnorm() [2/3]

```
double BayesicSpace::RanDraw::rnorm (
            const double & mu,
            const double & sigma ) const  [inline]
```

A Gaussian deviate.

Generates a Gaussian random value with mean μ and standard deviation σ. Implemented using a version of the Marsaglia and Tsang (2000) ziggurat algorithm, modified according to suggestions in the GSL implementation of the function.

Parameters

in	*mu*	standard deviation
in	*sigma*	standard deviation

Returns

a sample from the Gaussian distribution

5.6.3.10 rnorm() [3/3]

```
double BayesicSpace::RanDraw::rnorm (
            const double & sigma ) const  [inline]
```

A zero-mean Gaussian deviate.

Generates a Gaussian random value with mean $\mu = 0.0$ and standard deviation σ. Implemented using a version of the Marsaglia and Tsang (2000) ziggurat algorithm, modified according to suggestions in the GSL implementation of the function.

Parameters

in	*sigma*	standard deviation

Returns

a sample from the zero-mean Gaussian distribution

5.6.3.11 runif()

```
double BayesicSpace::RanDraw::runif ( ) const  [inline]
```

Generate a uniform deviate.

Returns

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

5.6.3.12 runifno()

```
double RanDraw::runifno ( ) const
```

Generate a non-one uniform deviate.

Returns

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

5.6.3.13 runifnz()

```
double RanDraw::runifnz ( ) const
```

Generate a non-zero uniform deviate.

Returns

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

5.6.3.14 runifop()

```
double RanDraw::runifop ( ) const
```

Generate an open-interval uniform deviate.

Returns

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

5.6.3.15 sampleInt() [1/2]

```
uint64_t BayesicSpace::RanDraw::sampleInt (
            const uint64_t & max ) const  [inline]
```

Sample and integer from the $[0, n)$ interval.

Parameters

in	*max*	the maximal value n (does not appear in the sample)

Returns

sampled value

5.6.3.16 sampleInt() [2/2]

```
uint64_t RanDraw::sampleInt (
            const uint64_t & min,
            const uint64_t & max ) const
```

Sample and integer from the $[m, n)$ interval.

Throws `string` "Lower bound not smaller than upper bound" if $m \geq n$.

Parameters

in	*min*	the minimal value m (can appear in the sample)
in	*max*	the maximal value n (does not appear in the sample)

Returns

sampled value

5.6.3.17 shuffleUint()

```
vector< uint64_t > RanDraw::shuffleUint (
            const uint64_t & N )
```

Draw non-negative intergers in random order.

Uses the Fisher-Yates-Durstenfeld algorithm to produce a random shuffle of integers in $[0, N)$.

Parameters

in	*Nmax*	the upper bound of the integer sequence

Returns

vector of N shuffled integers

5.6.3.18 type()

```
string BayesicSpace::RanDraw::type ( ) const  [inline]
```

Query RNG kind.

Find out the kind of (P)RNG in use.

Returns

String reflecting the RNG type

5.6.3.19 vitter()

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

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 **[vitter84a] [vitter87a]** 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

5.6.3.20 vitterA()

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

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 **[vitter84a] [vitter87a]** 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

Chapter 6

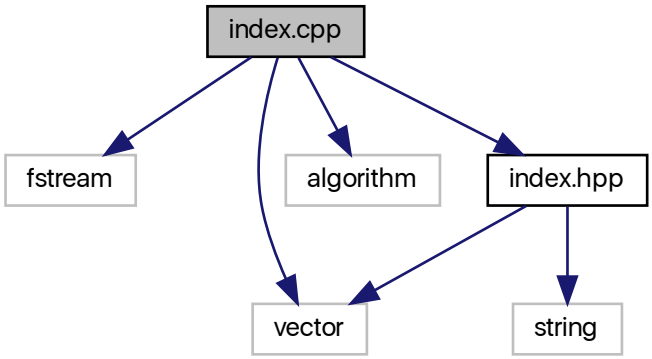
File Documentation

6.1 index.cpp File Reference

Connect lines with populations.

```
#include <fstream>
#include <vector>
#include <algorithm>
#include "index.hpp"
```

Include dependency graph for index.cpp:



6.1.1 Detailed Description

Connect lines with populations.

Author

Anthony J. Greenberg

Copyright

Version

1.0

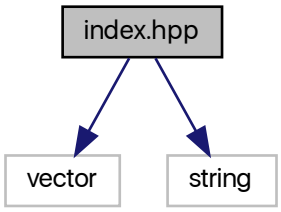
Implementation of a class that relates individuals to groups, similar to an factor in R.

6.2 index.hpp File Reference

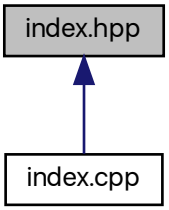
Connect lines with groups.

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

Include dependency graph for index.hpp:



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



Classes

- class BayesicSpace::Index

  Group index.

6.2.1 Detailed Description

Connect lines with groups.

Author

Anthony J. Greenberg

Copyright

Version

1.0

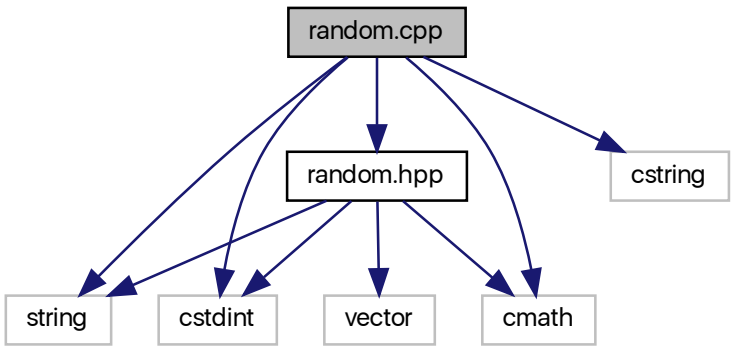
Definitions and interface documentation for a class that relates individuals to groups, similar to an factor in R.

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

Version

1.0

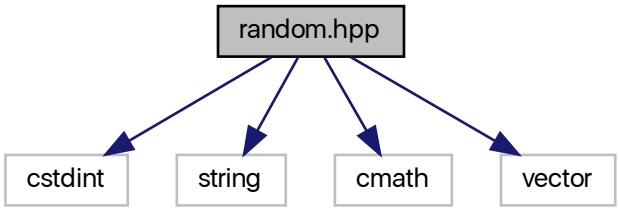
Class implementation for facilities that generate random draws from various distributions.

6.4 random.hpp File Reference

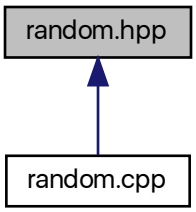
Random number generation.

```
#include <cstdint>
#include <string>
#include <cmath>
#include <vector>
```

Include dependency graph for random.hpp:



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



Classes

- class BayesicSpace::Generate

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

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

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

  Random number generating class.

6.4.1 Detailed Description

Random number generation.

Author

Anthony J. Greenberg

Copyright

Version

1.0

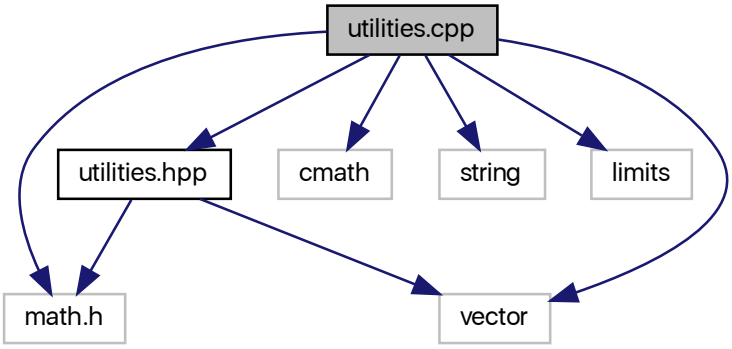
Class definition and interface documentation for facilities that generate random draws from various distributions.

6.5 utilities.cpp File Reference

Numerical utilities implementation.

```
#include <math.h>
#include <vector>
#include <cmath>
#include <string>
#include <limits>
#include "utilities.hpp"
```

Include dependency graph for utilities.cpp:



6.5.1 Detailed Description

Numerical utilities implementation.

Author

Anthony J. Greenberg

Copyright

Copyright (c) 2020 Anthony J. Greenberg

Version

1.0

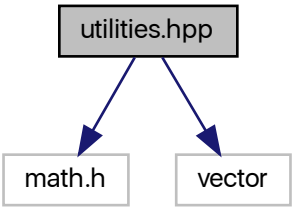
Class implementation for a set of numerical utilities. Implemented as a class because this seems to be the only way for these methods to be included using Rcpp with no compilation errors.

6.6 utilities.hpp File Reference

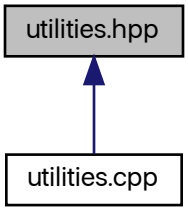
Numerical utilities.

```
#include <math.h>
#include <vector>
```

Include dependency graph for utilities.hpp:



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



Classes

- class BayesicSpace::NumerUtil

  Numerical utilities collection.

6.6.1 Detailed Description

Numerical utilities.

Author

Anthony J. Greenberg

Copyright

Version

1.0

Class definition for a set of numerical utilities. Implemented as a class because this seems to be the only way for these methods to be included using Rcpp with no compilation errors.

Index