Utilities

Generated by Doxygen 1.9.3

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, 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

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::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 std::vector< size_t > &vec)
  Vector constructor.
- Index (const std::string &inFileName)
  File read constructor.
- Index (const Index &in)
  Copy constructor.
- Index & operator= (const Index &in)
  Copy assignment operator.
- Index (Index &&in) noexcept
  Move constructor.
- Index & operator= (Index &&in) noexcept
  Move assignment operator.
- ~**Index** ()
  Destructor.
- const std::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 std::vector< size_t > &newVec)

  Update the index.

4.1.1 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.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Index() [1/6]

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

Group constructor.

Sets up empty groups.

Parameters

in	*Ngroups*	number of groups to set up

4.1.2.2 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

4.1.2.3 Index() [3/6]

```
Index::Index (
            const std::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

4.1.2.4 Index() [4/6]

```
Index::Index (
            const std::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

4.1.2.5 Index() [5/6]

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

Copy constructor.

Parameters

in	*in*	Index to be copied

4.1.2.6 Index() [6/6]

```
Index::Index (
            Index && in )  [noexcept]
```

Move constructor.

Parameters

in	*in*	Index object to be moved

4.1.3 Member Function Documentation

4.1.3.1 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

4.1.3.2 groupNumber()

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

Number of groups.

Returns

number of groups

4.1.3.3 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

4.1.3.4 neGroupNumber()

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

Number of non-empty groups.

Returns

number of non-empty groups

4.1.3.5 operator=() [1/2]

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

Copy assignment operator.

Parameters

in	*in*	object to be copied

Returns

an Index object

4.1.3.6 operator=() [2/2]

```
Index & Index::operator= (
            Index && in )  [noexcept]
```

Move assignment operator.

Parameters

in	*in*	object to be moved

Returns

an Index object

4.1.3.7 operator[]()

```
const std::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

4.1.3.8 size()

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

Total sample size.

Returns

total sample size

4.1.3.9 update()

```
void Index::update (
            const std::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:

- include/index.hpp
- src/index.cpp

4.2 BayesicSpace::NumerUtil Class Reference

Numerical utilities collection.

```
#include <utilities.hpp>
```

Public Member Functions

- void swapXOR (size_t &i, size_t &j) const noexcept

  Swap two `size_t` values.
- double logit (const double &p) const noexcept

  Logit function.
- double logistic (const double &x) const noexcept

  Logistic function.
- double lnGamma (const double &x) const noexcept

  Logarithm of the Gamma function.
- double digamma (const double &x) const noexcept

  Digamma function.
- double dotProd (const std::vector< double > &v) const noexcept

  Vector self-dot-product.
- double dotProd (const std::vector< double > &v1, const std::vector< double > &v2) const noexcept

  Dot-product of two vectors.
- void updateWeightedMean (const double &xn, const double &wn, double &mu, double &w) const noexcept

  Weighted mean update.
- double mean (const double arr[ ], const size_t &len) const noexcept

  Mean of an array.

4.2.1 Detailed Description

Numerical utilities collection.

Implements numerical functions for use throughout the project.

4.2.2 Member Function Documentation

4.2.2.1 digamma()

```
double NumerUtil::digamma (
            const double & x ) const  [noexcept]
```

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

4.2.2.2 dotProd() [1/2]

```
double NumerUtil::dotProd (
            const std::vector< double > & v ) const  [noexcept]
```

Vector self-dot-product.

Parameters

in	*v*	vector

Returns

dot-product value

4.2.2.3 dotProd() [2/2]

```
double NumerUtil::dotProd (
            const std::vector< double > & v1,
            const std::vector< double > & v2 ) const  [noexcept]
```

Dot-product of two vectors.

Parameters

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

Returns

dot-product value

4.2.2.4 lnGamma()

```
double NumerUtil::lnGamma (
            const double & x ) const  [noexcept]
```

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

$\log \Gamma(x)$

4.2.2.5 logistic()

```
double NumerUtil::logistic (
            const double & x ) const  [noexcept]
```

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

4.2.2.6 logit()

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

Logit function.

Parameters

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

Returns

logit transformation

4.2.2.7 mean()

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

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

4.2.2.8 swapXOR()

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

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

4.2.2.9 updateWeightedMean()

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

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:

- include/utilities.hpp
- src/utilities.cpp

4.3 BayesicSpace::RanDraw Class Reference

Random number generating class.

`#include <random.hpp>`

Public Member Functions

- RanDraw ()
  Default constructor.
- RanDraw (const uint64_t &seed)
  Constructor with seed.
- ~**RanDraw** ()
  Destructor.
- RanDraw (const RanDraw &old)=delete
  Copy constructor (deleted)
- RanDraw (RanDraw &&old)
  Move constructor.
- RanDraw & operator= (const RanDraw &old)=delete

Copy assignment (deleted)

- RanDraw & operator= (RanDraw &&old)

  Move assignment.
- uint64_t ranInt () noexcept

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

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

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

  Draw non-negative integers in random order.
- double runif () noexcept

  Generate a uniform deviate.
- double runifnz () noexcept

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

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

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

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

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

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

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

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

  A Dirichlet deviate.
- double rchisq (const double &nu) noexcept

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

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

  Sample from Vitter's distribution, method D.

4.3.1 Detailed Description

Random number generating class.

Generates pseudo-random deviates from a number of distributions. Uses an implementation of the 64-bit MT19937 ("Mersenne Twister") **[matsumoto98a]** pseudo-random number generator (PRNG) for random integers. This implementation of MT is ∼35% faster than in `std::random` and ∼250-fold faster than hardware RDRAND.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 RanDraw() [1/4]

```
BayesicSpace::RanDraw::RanDraw ( )  [inline]
```

Default constructor.

Seeded internally with a random number.

4.3.2.2 RanDraw() [2/4]

```
RanDraw::RanDraw (
            const uint64_t & seed )
```

Constructor with seed.

Sets the provided seed.

Parameters

in	*seed*	seed value

4.3.2.3 RanDraw() [3/4]

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

Copy constructor (deleted)

Parameters

in	*old*	object to be copied

4.3.2.4 RanDraw() [4/4]

```
RanDraw::RanDraw (
            RanDraw && old )
```

Move constructor.

Parameters

in	*old*	object to be moved

4.3.3 Member Function Documentation

4.3.3.1 operator=() [1/2]

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

Copy assignment (deleted)

Parameters

in	*old*	object to be copied

4.3.3.2 operator=() [2/2]

```
RanDraw & RanDraw::operator= (
            RanDraw && old )
```

Move assignment.

Parameters

in	*old*	object to be moved

4.3.3.3 ranInt()

```
uint64_t RanDraw::ranInt ( )  [noexcept]
```

Generate random integer.

Returns

An unsigned random 64-bit integer

4.3.3.4 rchisq()

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

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

4.3.3.5 rdirichlet()

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

A Dirichlet deviate.

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

Parameters

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

4.3.3.6 rgamma() [1/2]

```
double RanDraw::rgamma (
            const double & alpha )  [noexcept]
```

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

4.3.3.7 rgamma() [2/2]

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

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

4.3.3.8 rnorm() [1/3]

```
double RanDraw::rnorm ( )  [noexcept]
```

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

4.3.3.9 rnorm() [2/3]

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

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

4.3.3.10 rnorm() [3/3]

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

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

4.3.3.11 runif()

```
double BayesicSpace::RanDraw::runif ( )  [inline], [noexcept]
```

Generate a uniform deviate.

Returns

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

4.3.3.12 runifno()

```
double RanDraw::runifno ( )  [noexcept]
```

Generate a non-one uniform deviate.

Returns

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

4.3.3.13 runifnz()

```
double RanDraw::runifnz ( )  [noexcept]
```

Generate a non-zero uniform deviate.

Returns

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

4.3.3.14 runifop()

```
double RanDraw::runifop ( )  [noexcept]
```

Generate an open-interval uniform deviate.

Returns

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

4.3.3.15 sampleInt() [1/2]

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

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

Parameters

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

Returns

sampled value

4.3.3.16 sampleInt() [2/2]

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

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

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

4.3.3.17 shuffleUint()

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

Draw non-negative integers in random order.

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

Parameters

in	*N*	the upper bound of the integer sequence

Returns

vector of N shuffled integers

4.3.3.18 vitter()

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

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

4.3.3.19 vitterA()

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

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:

- include/random.hpp
- src/random.cpp

Chapter 5

File Documentation

5.1 include/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.

5.1.1 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.

5.2 index.hpp

Go to the documentation of this file.

```
1 /*
2  * Copyright (c) 2022 Anthony J. Greenberg
3  *
4  * Redistribution and use in source and binary forms, with or without modification, are permitted provided
       that the following conditions are met:
5  *
6  * 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the
       following disclaimer.
7  *
8  * 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.
9  *
10  * 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.
11  *
12  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED
       WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
13  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
       SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
14  * BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
       BUT NOT LIMITED TO, PROCUREMENT OF
15  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
       ON ANY THEORY OF LIABILITY, WHETHER
16  * 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
17  * THE POSSIBILITY OF SUCH DAMAGE.
18  */
19
21
30 #pragma once
31
32 #include <vector>
33 #include <string>
34
35 namespace BayesicSpace {
40     class Index {
41
42     public:
44         Index() {};
51         Index(const size_t &Ngroups);
59         Index(const size_t *arr, const size_t &N);
66         Index(const std::vector<size_t> &vec);
74         Index(const std::string &inFileName);
79         Index(const Index &in);
85         Index &operator=(const Index &in);
90         Index(Index &&in) noexcept;
96         Index &operator=(Index &&in) noexcept;
98         ~Index(){};
99
107         const std::vector<size_t> & operator[] (const size_t &i) const { return index_[i]; };
108
114         size_t groupSize(const size_t &i) const {return index_[i].size(); };
115
120         size_t size() const {return groupVal_.size(); };
121
126         size_t groupNumber() const {return index_.size(); };
127
132         size_t neGroupNumber() const;
133
142         size_t groupID(const size_t &ind) const {return groupVal_[ind]; };
143
151         void update(const std::vector<size_t> &newVec);
152
153     private:
158         std::vector< std::vector<size_t> > index_;
163         std::vector<size_t> groupVal_;
164
165     };
166
167 }
168
```

5.3 include/random.hpp File Reference

Random number generation.

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

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

Classes

- class BayesicSpace::RanDraw

  Random number generating class.

5.3.1 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.

5.4 random.hpp

Go to the documentation of this file.

```
1 /*
2  * Copyright (c) 2022 Anthony J. Greenberg
3  *
4  * Redistribution and use in source and binary forms, with or without modification, are permitted provided
       that the following conditions are met:
5  *
6  * 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the
       following disclaimer.
7  *
8  * 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.
9  *
10  * 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.
11  *
12  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED
       WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
13  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
       SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
14  * BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
       BUT NOT LIMITED TO, PROCUREMENT OF
15  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
       ON ANY THEORY OF LIABILITY, WHETHER
16  * 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
17  * THE POSSIBILITY OF SUCH DAMAGE.
18  */
19
21
30 #pragma once
31
32 #include <vector>
33 #include <array>
34 #include <cstdint>
35 #include <cmath>
36
37 namespace BayesicSpace {
38
39     class RanDraw;
40
41
49     class RanDraw {
50     public:
55         RanDraw() : RanDraw( randomSeed_() ) {};
62         RanDraw(const uint64_t &seed);
64         ~RanDraw(){ };
69         RanDraw(const RanDraw &old) = delete;
74         RanDraw(RanDraw &&old);
75
80         RanDraw & operator= (const RanDraw &old) = delete;
85         RanDraw & operator= (RanDraw &&old);
90         uint64_t ranInt() noexcept;
96         uint64_t sampleInt(const uint64_t &max) noexcept { return this->ranInt() % max; };
104         uint64_t sampleInt(const uint64_t &min, const uint64_t &max) noexcept;
113         std::vector<uint64_t> shuffleUint(const uint64_t &N);
114
119         double runif() noexcept {return 5.42101086242752217E-20 * static_cast<double>( this->ranInt() ); };
124         double runifnz() noexcept;
129         double runifno() noexcept;
134         double runifop() noexcept;
141         double rnorm() noexcept;
149         double rnorm(const double &sigma) noexcept { return this->rnorm() * sigma; };
158         double rnorm(const double &mu, const double &sigma) noexcept { return mu + this->rnorm() * sigma; };
166         double rgamma(const double &alpha) noexcept;
175         double rgamma(const double &alpha, const double &beta) noexcept { return beta > 0.0 ? (
       this->rgamma(alpha) ) / beta : nan(""); };
183         void rdirichlet(const std::vector<double> &alpha, std::vector<double> &p) noexcept;
191         double rchisq(const double &nu) noexcept { return 2.0 * this->rgamma(nu / 2.0); };
202         uint64_t vitterA(const double &n, const double &N) noexcept;
213         uint64_t vitter(const double &n, const double &N) noexcept;
214     private:
215         // Mersenne twister constants
217         static const uint16_t n_;
219         static const uint16_t m_;
221         static const uint64_t um_;
223         static const uint64_t lm_;
```

```
225         static const uint64_t b_;
227         static const uint64_t c_;
229         static const uint64_t d_;
231         static const uint32_t l_;
233         static const uint32_t s_;
235         static const uint32_t t_;
237         static const uint32_t u_;
239         static const std::array<uint64_t, 2> alt_;
241         std::array<uint64_t, 312> mt_;
243         size_t mti_;
245         uint64_t x_;
250         static const double paramR_;
251
253         static const std::array<double, 128> ytab_;
258         static const std::array<uint64_t, 128> ktab_;
259
264         static const std::array<double, 128> wtab_;
266         uint64_t randomSeed_() const ;
267     };
268
269 }
270
271
```

5.5 include/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.

5.5.1 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.

5.6 utilities.hpp

Go to the documentation of this file.

```
1 /*
2  * Copyright (c) 2022 Anthony J. Greenberg
3  *
4  * Redistribution and use in source and binary forms, with or without modification, are permitted provided
       that the following conditions are met:
5  *
6  * 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the
       following disclaimer.
7  *
8  * 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.
9  *
10  * 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.
11  *
12  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED
       WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
13  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
       SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
14  * BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
       BUT NOT LIMITED TO, PROCUREMENT OF
15  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
       ON ANY THEORY OF LIABILITY, WHETHER
16  * 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
17  * THE POSSIBILITY OF SUCH DAMAGE.
18  */
19
21
31 #pragma once
32
33 #include <math.h>
34 #include <vector>
35
36 namespace BayesicSpace {
42     class NumerUtil {
43     public:
51     void swapXOR(size_t &i, size_t &j) const noexcept;
57     double logit(const double &p) const noexcept { return log(p) - log(1.0 - p); }
65     double logistic(const double &x) const noexcept;
74     double lnGamma(const double &x) const noexcept;
82     double digamma(const double &x) const noexcept;
88     double dotProd(const std::vector<double> &v) const noexcept;
95     double dotProd(const std::vector<double> &v1, const std::vector<double> &v2) const noexcept;
110      void updateWeightedMean(const double &xn, const double &wn, double &mu, double &w) const noexcept;
119     double mean(const double arr[], const size_t &len) const noexcept;
120
121     private:
123         static const double gCoeff_[14];
125         static const double bvalues_[22];
126     };
127 }
128
```

5.7 src/index.cpp File Reference

Connect lines with populations.

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

Include dependency graph for index.cpp:

5.8 src/random.cpp File Reference

Random number generation.

```
#include <array>
#include <vector>
#include <cmath>
#include <numeric>
#include <cassert>
#include <random>
#include "random.hpp"
```

Include dependency graph for random.cpp:

5.8.1 Detailed Description

Random number generation.

Author

Anthony J. Greenberg

Copyright

Version

1.0

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

5.9 src/utilities.cpp File Reference

Numerical utilities implementation.

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

Include dependency graph for utilities.cpp:

5.9.1 Detailed Description

Numerical utilities implementation.

Author

Anthony J. Greenberg

Copyright

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.