BayesLiDiCal

Generated by Doxygen 1.8.20

Chapter 1

Main Page

This repository contains software made available by JanBiotech to estimate cell counts from limited dilution assays.

1.0.1 Simulations

The *simulation* folder contains the R code for simulating such assays and plotting the results. Running these simulations trains intuition and provides the ground truth for testing of model implementations. The simulations currently implemented follow the protocol of the QVOA assay to estimate the latent reservoir of replication competent HIV. The scripts depend on the data.table R package. Plotting depends on ggplot2 and showtext packages. I use the Myriad Pro fonts, users can substitute their own in the font_add() function call in the simPlots.Rnw script.

1.0.2 Quantal dilution assay analyses

The R package for Bayesian analyses of quantal dilution assays (such as QVOA) is in the BayesQLD directory of this repository. It depends only on Rcpp. It can be installed directly from GitHub by running install_github("↩ JanBiotech/BayesLiDiCal/BayesQLD"), which requires the devtools package.

BayesQLD is still in development, check back for newer versions.

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::BayesQLD Class Reference

Dilution assay model class.

`#include <model.hpp>`

Public Member Functions

- BayesQLD ()
  Default constructor.
- BayesQLD (const vector< double > &pWellN, const vector< double > &totWellN, const vector< double > &dilutionFrac)
  Constructor.
- ~BayesQLD ()
  Destructor.
- BayesQLD (const BayesQLD &in)
  Copy constructor.
- BayesQLD & operator= (const BayesQLD &in)
  Copy assignment operator.
- BayesQLD (BayesQLD &&in)
  Move constructor.
- BayesQLD & operator= (BayesQLD &&in)
  Move assignment operator.
- void sampler (const uint32_t &Nburnin, const uint32_t &Nsamples, vector< double > &thetaSamp, vector< uint32_t > &accept)
  Sampler.

5.1.1 Detailed Description

Dilution assay model class.

Keeps the data, fits the model, and saves samples from parameter distributions.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 BayesQLD() [1/3]

```
BayesQLD::BayesQLD (
            const vector< double > & pWellN,
            const vector< double > & totWellN,
            const vector< double > & dilutionFrac )
```

Constructor.

The constructor intializes θ.

Parameters

in	*pWellN*	vector with positive well numbers
in	*totWellN*	vector with total well numbers
in	*dilutionFrac*	vector with dilutino fractions

5.1.2.2 BayesQLD() [2/3]

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

Copy constructor.

Parameters

in	*in*	the object to be copied

5.1.2.3 BayesQLD() [3/3]

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

Move constructor.

Parameters

in	*in*	the object to be moved

5.1.3 Member Function Documentation

5.1.3.1 operator=() [1/2]

```
BayesQLD& BayesicSpace::BayesQLD::operator= (
            BayesQLD && in )
```

Move assignment operator.

Parameters

in	*object*	to be assigned

Returns

BayesQLD object

5.1.3.2 operator=() [2/2]

```
BayesQLD& BayesicSpace::BayesQLD::operator= (
            const BayesQLD & in )
```

Copy assignment operator.

Parameters

in	*object*	to be assigned

Returns

BayesQLD object

5.1.3.3 sampler()

```
void BayesQLD::sampler (
            const uint32_t & Nburnin,
            const uint32_t & Nsamples,
            vector< double > & thetaSamp,
            vector< uint32_t > & accept )
```

Sampler.

Runs the sampler. The output vectors are appended.

Parameters

in	*Nburnin*	number of burn-in iterations
in	*Nsamples*	number of sampling iterations
out	*thetaSamp*	sample of θ values
out	*accept*	vector of accept/reject events

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

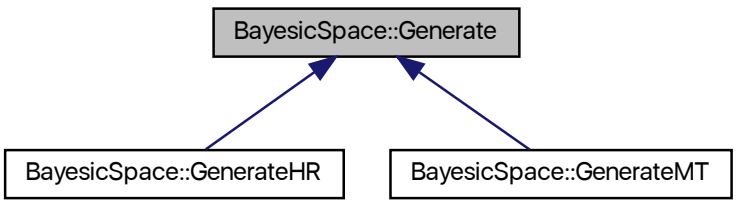
- BayesQLD/src/model.hpp
- BayesQLD/src/model.cpp

5.2 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.

5.2.1 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.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Generate() [1/2]

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

Protected copy constructor.

Parameters

in	*old*	object to copy

5.2.2.2 Generate() [2/2]

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

Protected move constructor.

Parameters

in	*old*	object to move

5.2.3 Member Function Documentation

5.2.3.1 operator=() [1/2]

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

Protected copy assignment operator.

Parameters

in	*old*	object to copy

5.2.3.2 operator=() [2/2]

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

Protected move assignment.

Parameters

in	*old*	object to move

5.2.3.3 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:

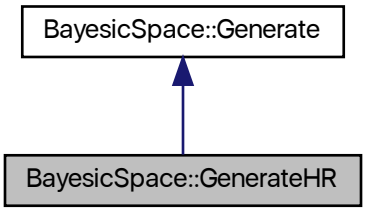
- BayesQLD/src/random.hpp

5.3 BayesicSpace::GenerateHR Class Reference

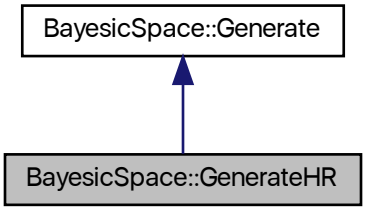
Hardware random number generating class.

```
#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

  Generate a random 64-bit unsigned integer.

Additional Inherited Members

5.3.1 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`.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 GenerateHR() [1/2]

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

Copy constructor.

Parameters

in	*old*	object to copy

5.3.2.2 GenerateHR() [2/2]

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

Move constructor.

Parameters

in	*old*	object to move

5.3.3 Member Function Documentation

5.3.3.1 operator=() [1/2]

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

Copy assignment operator.

Parameters

in	*old*	object to copy

5.3.3.2 operator=() [2/2]

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

Move assignment.

Parameters

in	*old*	object to move

5.3.3.3 ranInt()

```
uint64_t GenerateHR::ranInt ( ) const  [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:

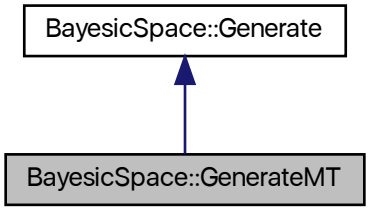
- BayesQLD/src/random.hpp
- BayesQLD/src/random.cpp

5.4 BayesicSpace::GenerateMT Class Reference

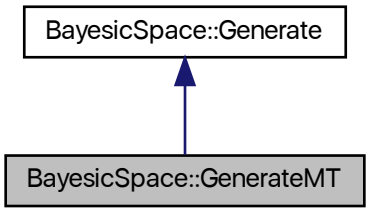
Pseudo-random number generator.

```
#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

  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

5.4.1 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.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 GenerateMT() [1/3]

`GenerateMT::GenerateMT ( )`

Default constructor.

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

5.4.2.2 GenerateMT() [2/3]

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

Copy constructor.

Parameters

in	*old*	object to copy

5.4.2.3 GenerateMT() [3/3]

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

Move constructor.

Parameters

in	*old*	object to move

5.4.3 Member Function Documentation

5.4.3.1 operator=() [1/2]

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

Copy assignment operator.

Parameters

in	*old*	object to copy

5.4.3.2 operator=() [2/2]

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

Move assignment.

Parameters

in	*old*	object to move

5.4.3.3 ranInt()

```
uint64_t GenerateMT::ranInt ( ) const  [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:

- BayesQLD/src/random.hpp
- BayesQLD/src/random.cpp

5.5 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.

5.5.1 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").

5.5.2 Constructor & Destructor Documentation

5.5.2.1 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.

5.5.2.2 RanDraw() [2/3]

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

Copy constructor.

Parameters

in	*old*	pbject to be copied

5.5.2.3 RanDraw() [3/3]

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

Move constructor.

Parameters

in	*old*	pbject to be moved

5.5.3 Member Function Documentation

5.5.3.1 operator=() [1/2]

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

Copy assignment.

Parameters

in	*old*	pbject to be copied

5.5.3.2 operator=() [2/2]

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

Move assignment.

Parameters

in	*old*	pbject to be moved

5.5.3.3 ranInt()

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

Generate random integer.

Returns

An unsigned random 64-bit integer

5.5.3.4 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

5.5.3.5 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

5.5.3.6 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

5.5.3.7 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

5.5.3.8 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

5.5.3.9 runif()

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

Generate a uniform deviate.

Returns

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

5.5.3.10 runifno()

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

Generate a non-one uniform deviate.

Returns

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

5.5.3.11 runifnz()

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

Generate a non-zero uniform deviate.

Returns

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

5.5.3.12 runifop()

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

Generate an open-interval uniform deviate.

Returns

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

5.5.3.13 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

5.5.3.14 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

5.5.3.15 shuffleUint()

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

Draw non-negative intergers in random order.

Uses 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

5.5.3.16 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

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

- BayesQLD/src/random.hpp
- BayesQLD/src/random.cpp

Chapter 6

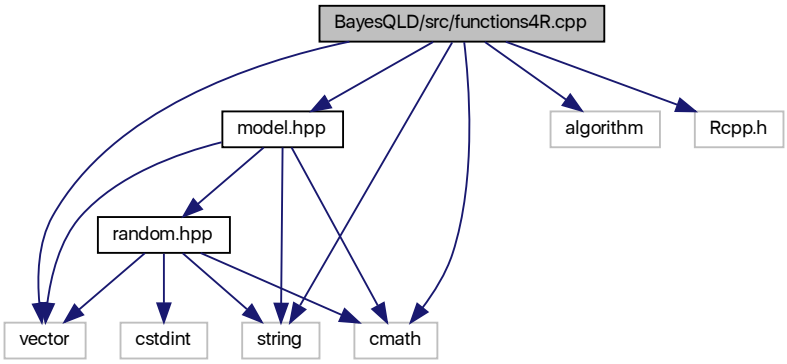
File Documentation

6.1 BayesQLD/src/functions4R.cpp File Reference

R interface to the MCMC sampler.

```
#include <vector>
#include <string>
#include <cmath>
#include <algorithm>
#include <Rcpp.h>
#include "model.hpp"
```

Include dependency graph for functions4R.cpp:



Functions

- Rcpp::List **runSampler** (const std::vector< double > &nPos, const std::vector< double > &nWells, const std↩ ::vector< double > &dilFrac, const int32_t &nChains, const int32_t &nBurnin, const int32_t &nSample)

6.1.1 Detailed Description

R interface to the MCMC sampler.

Author

Anthony J. Greenberg

Copyright

Version

1.0

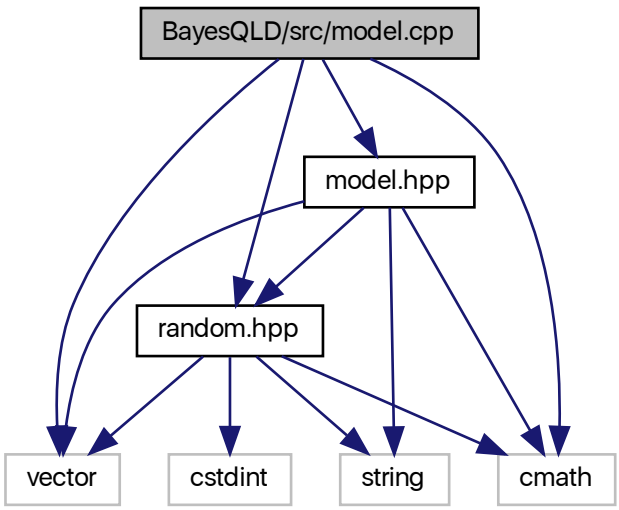
Contains the R interface to quantile limited dilution essay model fitting.

6.2 BayesQLD/src/model.cpp File Reference

Model for dilution series.

```
#include <vector>
#include <cmath>
#include "model.hpp"
#include "random.hpp"
```

Include dependency graph for model.cpp:



6.2.1 Detailed Description

Model for dilution series.

Author

Anthony J. Greenberg

Copyright

Version

1.0

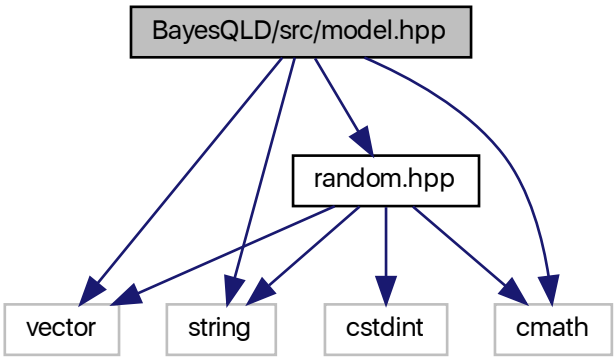
Function implementation for estimating the number of positives from a quantal limited dilution assay.

6.3 BayesQLD/src/model.hpp File Reference

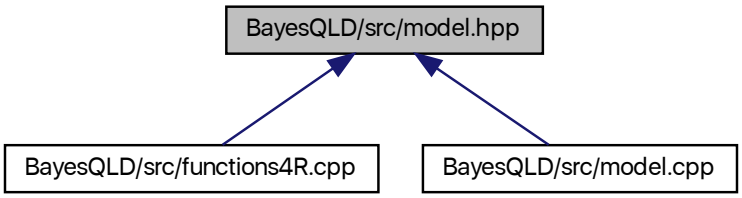
Model for dilution series.

```
#include <vector>
#include <string>
#include <cmath>
#include "random.hpp"
```

Include dependency graph for model.hpp:



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



Classes

- class BayesicSpace::BayesQLD

  Dilution assay model class.

6.3.1 Detailed Description

Model for dilution series.

Author

Anthony J. Greenberg

Copyright

Copyright (c) 2019 JanBiotech, Inc.

Version

1.0

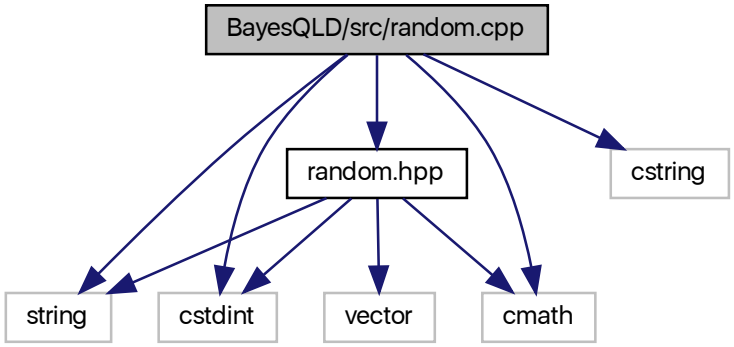
Class definition and interface documentation for estimating the number of positives from a quantal limited dilution assay.

6.4 BayesQLD/src/random.cpp File Reference

Random number generation.

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

Include dependency graph for random.cpp:



6.4.1 Detailed Description

Random number generation.

Author

Anthony J. Greenberg

Copyright

Copyright (c) 2017 Anthony J. Greenberg

Version

1.0

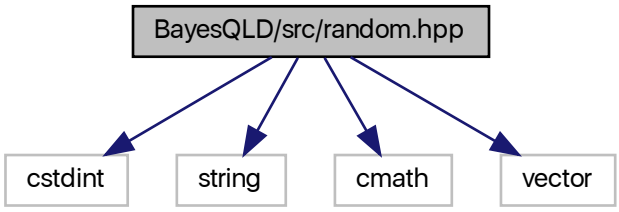
Class implementation for facilities that generate random draws from various distributions.

6.5 BayesQLD/src/random.hpp File Reference

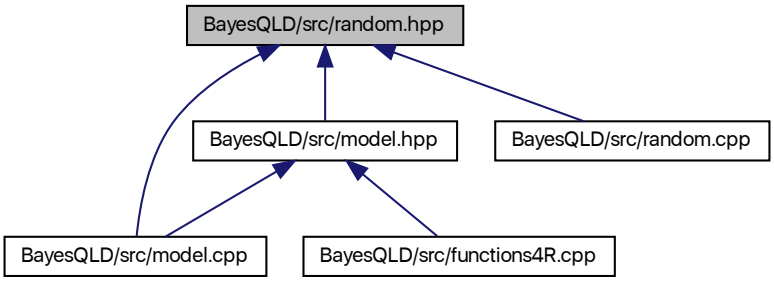
Random number generation.

```
#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.

6.5.1 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.

Index