RandomChains

Generated by Doxygen 1.8.13

Contents

Chapter 1

RandomChains

1.1 Program Description

The program RandomChains handles data and computes the number of expected random chains due to random fluctuations in the background for specific decay chains for experimental data. For the description of the method, see U. Forsberg et. al. Nuclear Physics A 953 (2016) 117–138. The main purposes of the program are:

- Provide free access to the code the Lund Nuclear Structure group has used
- Reproduce the numbers presented in the paper given above
- Try the method on user provided chains and experimental data

The complete program is located in the downloaded git repository. The program consists of the following three steps:

1. Read experimental data. Achieved with the constructor: RandomChains::RandomChains(int pixels, int bins, string folder)
2. Read decay chain/chains characteristics data. Achieved with the method: RandomChains::SetDecayChains(string input_chains)
3. Compute the number of expected random chains for the given decay chain/chains with the experimental data. Achieved with the method: RandomChains::Run()

1.1.1 How to run RandomChains?

The program is preferably controlled from the file `run_file.cc`.

When located in the directory of RandomChains type the following in the terminal to run the program:

1. `make`
2. `./run_file`

1.1.2 Computer requirements

The program has been successfully run on the following systems:

- Linux Ubuntu 14.04 with g++ version 4.8.4 and std=c++11

1.2 Experimental data

The Lund experimental data (see D. Rudolph et. al / Phys. Rev. Lett. 111, 112502 (2013)) is located in the folder `Lund_data` and **SHOULD NOT BE MODIFIED**. Within this folder the data is stored in '`.csv`' files. The data consists of spectra for the pixels in the implantation detector for different beam status and reconstructed or not. The structure of these files are:

The spectrum histogram bin values are given after each other separated with a ',' from the first to the last bin and then for the first to the last pixel (the pixel order does not matter). The fissions and in which pixel they have occurred are stored in another file. The data consists of four files:

- `beam_on.csv`: The spectra for all pixels for beam ON. **OPTIONAL!**
- `rec_beam_on.csv`: The reconstructed spectra (for escaped alpha particles) for all pixels for beam ON. **MANDATORY!**
- `rec_beam_off.csv`: The reconstructed spectra (for escaped alpha particles) for all pixels for beam OFF. **MANDATORY!**
- `pixels_with_fissions.csv`: The pixel number where a fission has occurred are given in this file. If two fissions have occurred within a pixel this will be presented in this file by two occurrences of this pixel number. **MANDATORY!**

To be able to read in the experimental data, the number of pixels in the implantation detector, the total number of bins in each of the spectra and the folder in which the data is stored, need to be given as input.

1.2.1 FOR USERS WHO WISH TO RUN THE PROGRAM ON OTHER EXPERIMENTAL DATA:

Create a new folder with files with the same names as in the `Lund_data` folder and insert the new data here. In the program, the user only needs to provide the name of this created folder, the number of pixels and the total number of bins for the data in the constructor: RandomChains::RandomChains(int pixels, int bins, string folder)

1.3 Decay chains

From a user perspective, what defines a decay chain is the following characteristics:

- `chain_length`: *x*, where *x* is the length of the chain
- `decay_type`: *a*=alpha, *e*=escape and *f*=fission
- `beam_status`: *1*=ON and *0*=OFF
- `time_span`: *t*, where *t* is the length of the time window during which the decay is accepted.

In the program the following limits, given in bins *E*, in the spectra determines the decay type:

a=alpha	lower_limit_alphas <= E < upper_limit_alphas
e=escape	lower_limit_escapes <= E < upper_limit_escapes
Implants (only for beam ON) are defined as:	
implants	lower_limit_implants <= E < upper_limit_implants

The decay chains are given to the program with an input file. In the downloaded repository examples of such input files are dump_article.txt and dump_test.txt. If no input of decay chains are provided the user can choose between two options:

- 0: '*Reproduce article numbers*', i.e. reproduce the numbers presented in U. Forsberg et. al. Nuclear Physics A 953 (2016) 117-138.
- 2: '*Test run*'. With this option the program is tested on trivial data. The obtained number from the program is compared to a value obtained through the calculation of a simple formula.

1.3.1 FOR USERS WHO WISH TO SPECIFY HER/HIS OWN DECAY CHAINS:

Have a look at the file dump_articles.txt. This file contains the input data if one wants to reproduce the article numbers by the Lund group. Edit this file after your own specifications and save it. In the method RandomChains::SetDecayChains(string input_chains) provide the name of your created file as the string input_chains.

OBS: The duration of the experiment is also given in the same file.

1.4 Calculate the expected number of random chains

The expected number of random chains is calculated with the method described in U. Forsberg et. al. Nuclear Physics A 953 (2016) 117-138.

1.5 Files and folders

A list of files and folders is provided below:

RandomChains.cc: The class RandomChains is implemented here. All methods and functions can be found here.

RandomChains.h: Header file for RandomChains.cc.

run_file.cc: From this file the user should control and execute the program. Examples of how this can be done already exists here.

Makefile: By typing make within the folder this file is run and the program is built and the executable run_file is created.

run_file: The program executable which can be run with ./run_file

dump_input.txt: User input chains are dumped to the following file. It has the same format as the input chains files should be given.

dump_article.txt: Input chains to reproduce the numbers in the Lund article are dumped to this file.

dump_test.txt: Input chains for the trivial test which can be run to verify the workings of the program are dumped to this file.

Lund_data: Folder which contains the Lund experimental data.

Lund_data/beam_on.csv: Comma separated file of the spectra for all pixels in the implantation detector for beam ON.

Lund_data/rec_beam_on.csv: Comma separated file of the reconstructed spectra for all pixels in the implantation detector for beam ON.

Lund_data/rec_beam_off.csv: Comma separated file of the reconstructed spectra for all pixels in the implantation detector for beam OFF.

Lund_data/pixels_with_fissions.csv: Comma separated file of the pixels in which a fission occurred.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

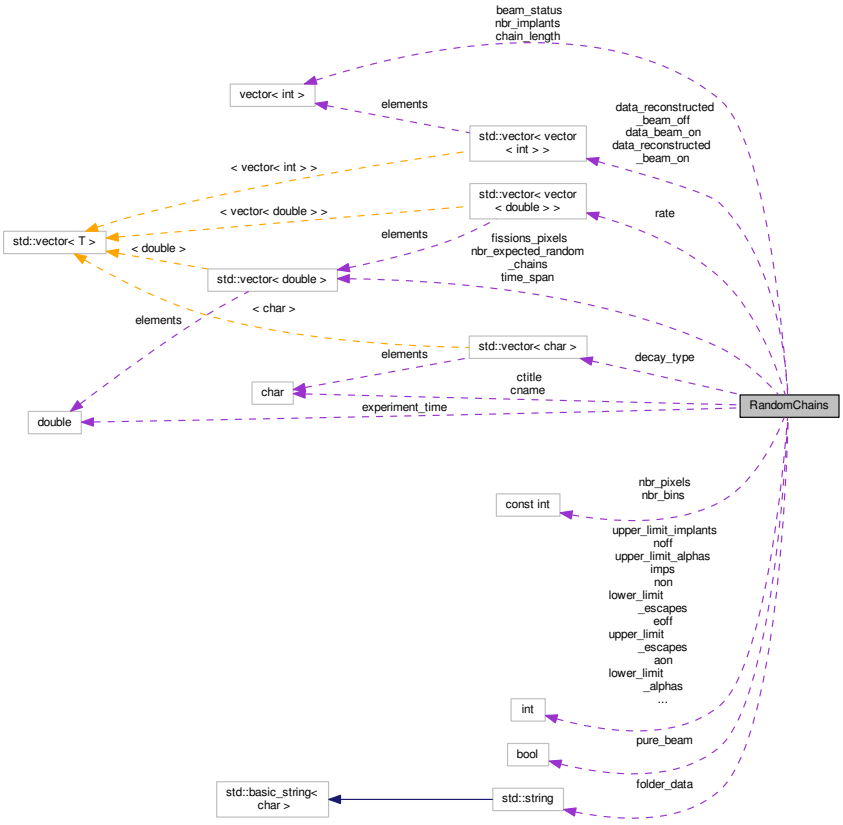
Chapter 4

Data Structure Documentation

4.1 RandomChains Class Reference

`#include <RandomChains.h>`

Collaboration diagram for RandomChains:



Public Member Functions

- RandomChains (int pixels=1024, int bins=4096, string folder="Lund_data")
- void ReadExperimentalData ()
- void SetDecayChains (string input_chains="")
- void Run ()
- ~RandomChains ()
- void print_result ()
- void print_test_result ()
- void dump_input_to_file ()

Private Member Functions

- void read_exp_file (string file_name)
- void generate_test_data ()
- void calculate_implants ()
- void calculate_rates ()
- void calculate_expected_nbr_random_chains ()
- void set_test_chains ()
- void set_article_chains ()
- void set_chains_from_input_file (string input_file)
- void rate_calc (char type, int beam)

Private Attributes

- const int nbr_pixels
- const int nbr_bins
- string folder_data
- int run_type
- double experiment_time
- bool pure_beam = true
- int lower_limit_alphas
- int upper_limit_alphas
- int lower_limit_escapes
- int upper_limit_escapes
- int lower_limit_implants
- int upper_limit_implants
- vector< vector< int > > data_beam_on
- vector< vector< int > > data_reconstructed_beam_on
- vector< vector< int > > data_reconstructed_beam_off
- vector< double > fissions_pixels
- vector< int > nbr_implants
- vector< int > chain_length
- vector< int > beam_status
- vector< char > decay_type
- vector< double > time_span
- vector< vector< double > > rate
- vector< double > nbr_expected_random_chains
- int eon
- int eoff
- int non
- int noff
- int aon
- int aoff
- int imps
- int fissions
- char cname [64]
- char ctitle [64]

4.1.1 Detailed Description

Definition at line 13 of file RandomChains.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 RandomChains()

```
RandomChains::RandomChains (
            int pixels = 1024,
            int bins = 4096,
            string folder = "Lund_data" )
```

In the constructor the experimental data are read in from ".csv" files in the folder given as input argument. The number of pixels in the implantation detector and the total number of bins in each spectrum are also provided as input arguments.

Parameters

pixels	number of pixels in the spectrum data
bins	total number of bins in every spectrum
folder	name of the folder which contains the experimental data

Returns

returns object of the class RandomChains

See also

ReadExperimentalData()

The following is initialised:

- RandomChains::folder_data
- RandomChains::data_beam_on
- RandomChains::data_reconstructed_beam_on
- RandomChains::data_reconstructed_beam_off

Definition at line 240 of file RandomChains.cc.

4.1.2.2 ~RandomChains()

```
RandomChains::~RandomChains ( )
```

The destructor of RandomChains. Object is deleted.

Definition at line 469 of file RandomChains.cc.

4.1.3 Member Function Documentation

4.1.3.1 read_exp_file()

```
void RandomChains::read_exp_file (
            string read_file )  [private]
```

The experimental data files are read in. The experimental data in the comma separated files are read in from the folder provided in the constructor. The files read in are: "beam_on.csv", "recon_beam_on.csv", "recon_beam_off.csv" and "pixels_with_fissions.csv"

Parameters

read_file	the name of the file to be read in.

The following is initialised:

- RandomChains::data_beam_on
- RandomChains::data_reconstructed_beam_on
- RandomChains::data_reconstructed_beam_off
- RandomChains::fissions_pixels

Definition at line 393 of file RandomChains.cc.

Referenced by ReadExperimentalData().

4.1.3.2 generate_test_data()

```
void RandomChains::generate_test_data ( )  [private]
```

The test data is generated. The test data is generated in this method. All the read in data is substituted.

The following is initialised:

- RandomChains::eon

- RandomChains::eoff
- RandomChains::non
- RandomChains::noff
- RandomChains::aon
- RandomChains::aoff
- RandomChains::imps
- RandomChains::fissions
- RandomChains::data_beam_on
- RandomChains::data_reconstructed_beam_on
- RandomChains::data_reconstructed_beam_off
- RandomChains::fissions_pixels

Definition at line 491 of file RandomChains.cc.

Referenced by SetDecayChains().

4.1.3.3 calculate_implants()

```
void RandomChains::calculate_implants ( )  [private]
```

Calculates the number of implants. The number of implants in every pixel is calculated with the lower and upper limits set in *SetDecayChains*.

The following is initialised:

- RandomChains::nbr_implants

Definition at line 726 of file RandomChains.cc.

Referenced by Run().

4.1.3.4 calculate_rates()

```
void RandomChains::calculate_rates ( )  [private]
```

This method calculates the rates in every pixel for the specific decay types, one decay at a time. The rate in every pixel is calculated with the lower and upper limits set in *SetDecayChains*.

The following is initialised:

- RandomChains::nbr_implants

Definition at line 750 of file RandomChains.cc.

4.1.3.5 calculate_expected_nbr_random_chains()

```
void RandomChains::calculate_expected_nbr_random_chains ( )  [private]
```

This method calculates the TOTAL number of expected random chains for the input decay chain/chains. On the basis of the rates calculated for every decay the expected number of random chains due to random fluctuations in the background are determined per pixel and decay chain. The values of every pixel are then summed for every decay chain to a final value.

The following is initialised:

- RandomChains::nbr_expected_random_chains

Definition at line 825 of file RandomChains.cc.

Referenced by Run().

4.1.3.6 set_test_chains()

```
void RandomChains::set_test_chains ( )  [private]
```

The test chains are set. This method defines the test chains characteristics and limits for the different signal types. The experiment duration is also set. The following is initialised:

- RandomChains::chain_length
- RandomChains::beam_status
- RandomChains::decay_type
- RandomChains::time_span
- RandomChains::lower_limit_alphas, upper_limit_alphas
- RandomChains::lower_limit_escapes, upper_limit_escapes
- RandomChains::lower_limit_implants, upper_limit_implants
- RandomChains::experiment_time

Definition at line 549 of file RandomChains.cc.

Referenced by SetDecayChains().

4.1.3.7 set_article_chains()

```
void RandomChains::set_article_chains ( )  [private]
```

The Lund article chains are set. This method defines the Lund article chains characteristics and limits for the different signal types. The experiment duration is also set.

The following is initialised:

- RandomChains::chain_length
- RandomChains::beam_status
- RandomChains::decay_type
- RandomChains::time_span
- RandomChains::lower_limit_alphas, upper_limit_alphas
- RandomChains::lower_limit_escapes, upper_limit_escapes
- RandomChains::lower_limit_implants, upper_limit_implants
- RandomChains::experiment_time

Definition at line 578 of file RandomChains.cc.

Referenced by SetDecayChains().

4.1.3.8 set_chains_from_input_file()

```
void RandomChains::set_chains_from_input_file (
            string input_chains )  [private]
```

This method sets the chain/chains from a given input file. This method sets the chains from a file. It is necessary that the format of the file which is to be read in follows the correct format. The spaces are important! What is read in is printed in the terminal window and can also be found in the file dumped after a run.

Parameters

input_chains	file name of the input chains which are provided by the user in the method *SetDecayChains*

The following is initialised:

- RandomChains::chain_length
- RandomChains::beam_status
- RandomChains::decay_type
- RandomChains::time_span
- RandomChains::lower_limit_alphas, upper_limit_alphas

- RandomChains::lower_limit_escapes, upper_limit_escapes
- RandomChains::lower_limit_implants, upper_limit_implants
- RandomChains::experiment_time

Definition at line 654 of file RandomChains.cc.

Referenced by SetDecayChains().

4.1.3.9 rate_calc()

```
void RandomChains::rate_calc (
            char type,
            int beam )  [private]
```

This method calculates the rates in every pixel for the specific decay types and beam status for a decay. Given the decay type and beam status the rate for every pixel is calculated and stored in the 2D vector *rate*.

Parameters

type	decay type, i.e. 'a', 'e' or 'f'.
beam	beam status, i.e. 1 or 0.

The following is intialised:

- RandomChains::rate

Definition at line 768 of file RandomChains.cc.

Referenced by calculate_rates(), and Run().

4.1.3.10 ReadExperimentalData()

```
void RandomChains::ReadExperimentalData ( )
```

The experimental data is read in. The experimental data is read in from the folder provided in the constructor. All vectors are initialised. The spectrum data and fission data are read in with the method *read_exp_file(string file_name)* .

See also

RandomChains::read_exp_file(string file_name)

The following data is initialised:

- RandomChains::data_beam_on
- RandomChains::data_reconstructed_beam_on
- RandomChains::data_reconstructed_beam_off
- RandomChains::fissions_pixels
- RandomChains::nbr_implants

Definition at line 353 of file RandomChains.cc.

Referenced by RandomChains().

4.1.3.11 SetDecayChains()

```
void RandomChains::SetDecayChains (
            string input_chains = "" )
```

This method sets the decay chain/chains characteristics. This is the main method for setting the decay chains. In this method the decay chain/chains characteristics are set with an input file provided as input argument. If no input file is given, the user gets to choose the type of run; reproduce article numbers or test the program on trivial data. Depending on the type of run different methods are invoked. To verify that this step was made successfully, the input is dumped to a file named "dump_input.txt" for user provided input and "dump_article.txt" for reproduce article numbers and "dump_test.txt" for the test.

Parameters

input_chains	the name of the file from which the input should be read

See also

set_test_chains()
generate_test_data()
set_article_chains()
set_chains_from_input_file(string input_file)
dump_input_to_file()

The following is initialised:

- RandomChains::run_type

Definition at line 263 of file RandomChains.cc.

Referenced by main().

4.1.3.12 Run()

```
void RandomChains::Run ( )
```

This method computes the expected number of random chains. With this method the expected number of random chains due to random fluctuations in the background for the decay chain/chains and experimental data provided. First the number of implants per pixel is calculated. Then the background rates in every pixel for the different decay types are calculated. This is followed by the calculation of the expected number of random chains. Finally, the results are printed in the terminal window.

See also

RandomChains::calculate_implants()
RandomChains::rate_calc()
RandomChains::calculate_expected_nbr_random_chains()
RandomChains::print_result()

Definition at line 322 of file RandomChains.cc.

Referenced by main().

4.1.3.13 print_result()

```
void RandomChains::print_result ( )
```

The results of the run are printed. This is the method that is invoked at the end of the constructor and it presents the result of the run in the terminal window. If the test was run another member function is called for further output.

The following is initialised:

- RandomChains::nbr_expected_random_chains

Definition at line 876 of file RandomChains.cc.

Referenced by Run().

4.1.3.14 print_test_result()

```
void RandomChains::print_test_result ( )
```

The results of the test run are printed. This method calculates the random chains in the test run with a simple formula and prints this result in the terminal window.

Definition at line 894 of file RandomChains.cc.

Referenced by print_result().

4.1.3.15 dump_input_to_file()

```
void RandomChains::dump_input_to_file ( )
```

This method dumps the input chains to a file. This method dumps the set chains to a file. If the test is run the file name is "dump_test.txt" and if the reproduce article numbers is run the file name is "dump_article.txt". Otherwise the file name is "dump_input.txt".

Definition at line 596 of file RandomChains.cc.

Referenced by SetDecayChains().

4.1.4 Field Documentation

4.1.4.1 nbr_pixels

```
const int RandomChains::nbr_pixels  [private]
```

Definition at line 15 of file RandomChains.h.

Referenced by calculate_expected_nbr_random_chains(), calculate_implants(), generate_test_data(), print_test↩_result(), rate_calc(), read_exp_file(), and ReadExperimentalData().

4.1.4.2 nbr_bins

```
const int RandomChains::nbr_bins  [private]
```

Definition at line 16 of file RandomChains.h.

Referenced by generate_test_data(), read_exp_file(), and ReadExperimentalData().

4.1.4.3 folder_data

```
string RandomChains::folder_data  [private]
```

Definition at line 18 of file RandomChains.h.

Referenced by RandomChains(), read_exp_file(), and ReadExperimentalData().

4.1.4.4 run_type

```
int RandomChains::run_type  [private]
```

Definition at line 21 of file RandomChains.h.

Referenced by dump_input_to_file(), print_result(), and SetDecayChains().

4.1.4.5 experiment_time

```
double RandomChains::experiment_time  [private]
```

Definition at line 24 of file RandomChains.h.

Referenced by dump_input_to_file(), print_test_result(), rate_calc(), set_article_chains(), set_chains_from_input↩_file(), and set_test_chains().

4.1.4.6 pure_beam

```
bool RandomChains::pure_beam = true  [private]
```

Definition at line 27 of file RandomChains.h.

Referenced by calculate_implants(), read_exp_file(), and SetDecayChains().

4.1.4.7 lower_limit_alphas

```
int RandomChains::lower_limit_alphas  [private]
```

Definition at line 30 of file RandomChains.h.

Referenced by dump_input_to_file(), generate_test_data(), print_test_result(), rate_calc(), set_article_chains(), set_chains_from_input_file(), and set_test_chains().

4.1.4.8 upper_limit_alphas

```
int RandomChains::upper_limit_alphas  [private]
```

Definition at line 30 of file RandomChains.h.

Referenced by dump_input_to_file(), generate_test_data(), print_test_result(), rate_calc(), set_article_chains(), set_chains_from_input_file(), and set_test_chains().

4.1.4.9 lower_limit_escapes

```
int RandomChains::lower_limit_escapes  [private]
```

Definition at line 31 of file RandomChains.h.

Referenced by dump_input_to_file(), generate_test_data(), print_test_result(), rate_calc(), set_article_chains(), set_chains_from_input_file(), and set_test_chains().

4.1.4.10 upper_limit_escapes

```
int RandomChains::upper_limit_escapes  [private]
```

Definition at line 31 of file RandomChains.h.

Referenced by dump_input_to_file(), generate_test_data(), print_test_result(), rate_calc(), set_article_chains(), set_chains_from_input_file(), and set_test_chains().

4.1.4.11 lower_limit_implants

```
int RandomChains::lower_limit_implants  [private]
```

Definition at line 32 of file RandomChains.h.

Referenced by calculate_implants(), dump_input_to_file(), generate_test_data(), set_article_chains(), set_chains↩_from_input_file(), and set_test_chains().

4.1.4.12 upper_limit_implants

```
int RandomChains::upper_limit_implants  [private]
```

Definition at line 32 of file RandomChains.h.

Referenced by calculate_implants(), dump_input_to_file(), generate_test_data(), set_article_chains(), set_chains↩_from_input_file(), and set_test_chains().

4.1.4.13 data_beam_on

```
vector< vector<int> > RandomChains::data_beam_on  [private]
```

Definition at line 35 of file RandomChains.h.

Referenced by calculate_implants(), read_exp_file(), and ReadExperimentalData().

4.1.4.14 data_reconstructed_beam_on

```
vector< vector<int> > RandomChains::data_reconstructed_beam_on  [private]
```

Definition at line 36 of file RandomChains.h.

Referenced by calculate_implants(), generate_test_data(), rate_calc(), read_exp_file(), and ReadExperimental↩ Data().

4.1.4.15 data_reconstructed_beam_off

```
vector< vector<int> > RandomChains::data_reconstructed_beam_off  [private]
```

Definition at line 37 of file RandomChains.h.

Referenced by generate_test_data(), rate_calc(), read_exp_file(), and ReadExperimentalData().

4.1.4.16 fissions_pixels

```
vector<double> RandomChains::fissions_pixels  [private]
```

Definition at line 40 of file RandomChains.h.

Referenced by generate_test_data(), rate_calc(), read_exp_file(), and ReadExperimentalData().

4.1.4.17 nbr_implants

```
vector<int> RandomChains::nbr_implants  [private]
```

Definition at line 43 of file RandomChains.h.

Referenced by calculate_expected_nbr_random_chains(), calculate_implants(), and ReadExperimentalData().

4.1.4.18 chain_length

```
vector<int> RandomChains::chain_length  [private]
```

Definition at line 46 of file RandomChains.h.

Referenced by calculate_expected_nbr_random_chains(), dump_input_to_file(), set_article_chains(), set_chains↩ _from_input_file(), and set_test_chains().

4.1.4.19 beam_status

```
vector<int> RandomChains::beam_status  [private]
```

Definition at line 47 of file RandomChains.h.

Referenced by calculate_rates(), dump_input_to_file(), Run(), set_article_chains(), set_chains_from_input_file(), and set_test_chains().

4.1.4.20 decay_type

```
vector<char> RandomChains::decay_type  [private]
```

Definition at line 48 of file RandomChains.h.

Referenced by calculate_rates(), dump_input_to_file(), Run(), set_article_chains(), set_chains_from_input_file(), and set_test_chains().

4.1.4.21 time_span

```
vector<double> RandomChains::time_span  [private]
```

Definition at line 49 of file RandomChains.h.

Referenced by calculate_expected_nbr_random_chains(), dump_input_to_file(), print_test_result(), set_article_↩chains(), set_chains_from_input_file(), and set_test_chains().

4.1.4.22 rate

```
vector< vector<double> > RandomChains::rate  [private]
```

Definition at line 52 of file RandomChains.h.

Referenced by calculate_expected_nbr_random_chains(), and rate_calc().

4.1.4.23 nbr_expected_random_chains

```
vector<double> RandomChains::nbr_expected_random_chains  [private]
```

Definition at line 53 of file RandomChains.h.

Referenced by calculate_expected_nbr_random_chains(), and print_result().

4.1.4.24 eon

```
int RandomChains::eon  [private]
```

Definition at line 56 of file RandomChains.h.

Referenced by generate_test_data(), and print_test_result().

4.1.4.25 eoff

```
int RandomChains::eoff  [private]
```

Definition at line 57 of file RandomChains.h.

Referenced by generate_test_data(), and print_test_result().

4.1.4.26 non

```
int RandomChains::non  [private]
```

Definition at line 58 of file RandomChains.h.

Referenced by generate_test_data().

4.1.4.27 noff

```
int RandomChains::noff  [private]
```

Definition at line 59 of file RandomChains.h.

Referenced by generate_test_data().

4.1.4.28 aon

```
int RandomChains::aon  [private]
```

Definition at line 60 of file RandomChains.h.

Referenced by generate_test_data(), and print_test_result().

4.1.4.29 aoff

```
int RandomChains::aoff  [private]
```

Definition at line 61 of file RandomChains.h.

Referenced by generate_test_data(), and print_test_result().

4.1.4.30 imps

```
int RandomChains::imps  [private]
```

Definition at line 62 of file RandomChains.h.

Referenced by generate_test_data(), and print_test_result().

4.1.4.31 fissions

```
int RandomChains::fissions  [private]
```

Definition at line 63 of file RandomChains.h.

Referenced by generate_test_data(), and print_test_result().

4.1.4.32 cname

```
char RandomChains::cname[64]  [private]
```

Definition at line 65 of file RandomChains.h.

4.1.4.33 ctitle

```
char RandomChains::ctitle[64]  [private]
```

Definition at line 65 of file RandomChains.h.

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

- RandomChains.h
- RandomChains.cc

Chapter 5

File Documentation

5.1 dump_article.txt File Reference

5.2 dump_input.txt File Reference

5.3 dump_test.txt File Reference

5.4 Lund_data/beam_on.csv File Reference

5.5 Lund_data/pixels_with_fissions.csv File Reference

5.6 Lund_data/rec_beam_off.csv File Reference

5.7 Lund_data/rec_beam_on.csv File Reference

5.8 Makefile File Reference

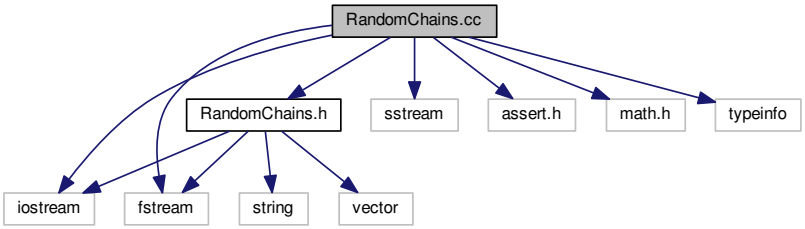
5.9 RandomChains.cc File Reference

Main program file.

```
#include <iostream>
#include <fstream>
#include <sstream>
#include "RandomChains.h"
#include <assert.h>
#include "math.h"
```

```
#include <typeinfo>
```

Include dependency graph for RandomChains.cc:



Functions

- double Poisson_pmf (int nbr_to_observe, double expected_value)
- int factorial (int k)

5.9.1 Detailed Description

Main program file.

From this file the program should be controlled.

Author

Anton Roth (anton.roth@nuclear.lu.se)

5.9.2 Function Documentation

5.9.2.1 Poisson_pmf()

```
double Poisson_pmf (
            int nbr_to_observe,
            double expected_value )
```

Poisson probability mass function. *p(k) = lambda^k exp(-lambda)/k!*

Parameters

expected_value	*lambda*
nbr_to_observe	*k*

Returns

p(k), i.e. the probability to observe k observations

Definition at line 926 of file RandomChains.cc.

Referenced by RandomChains::calculate_expected_nbr_random_chains(), and RandomChains::print_test_result().

5.9.2.2 factorial()

```
int factorial (
            int k )
```

Factorial

Parameters

k	

Returns

factorial of *k*

Definition at line 936 of file RandomChains.cc.

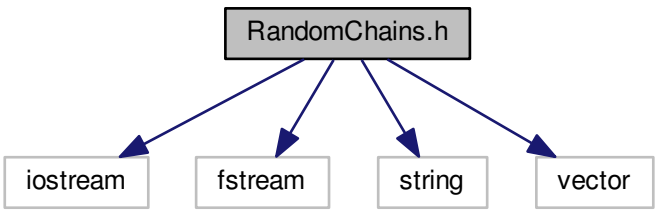
Referenced by Poisson_pmf().

5.10 RandomChains.h File Reference

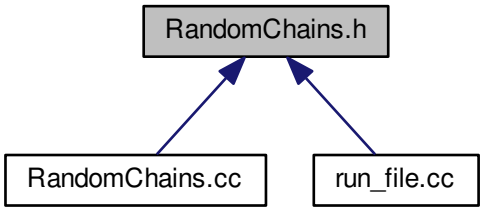
Header file for RandomChains.cc.

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
```

Include dependency graph for RandomChains.h:



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



Data Structures

- class RandomChains

Functions

- double Poisson_pmf (int nbr_to_observe, double expected_value)
- int factorial (int k)

5.10.1 Detailed Description

Header file for RandomChains.cc.

Author

Anton Roth (anton.roth@nuclear.lu.se)

5.10.2 Function Documentation

5.10.2.1 Poisson_pmf()

```
double Poisson_pmf (
            int nbr_to_observe,
            double expected_value )
```

Poisson probability mass function. *p(k) = lambda^ k exp(-lambda)/k!*

Parameters

expected_value	*lambda*
nbr_to_observe	*k*

Returns

p(k), i.e. the probability to observe k observations

Definition at line 926 of file RandomChains.cc.

Referenced by RandomChains::calculate_expected_nbr_random_chains(), and RandomChains::print_test_result().

5.10.2.2 factorial()

```
int factorial (
            int k )
```

Factorial

Parameters

k	

Returns

factorial of *k*

Definition at line 936 of file RandomChains.cc.

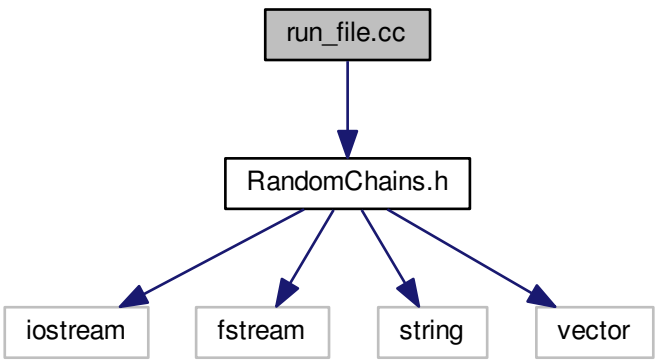
Referenced by Poisson_pmf().

5.11 run_file File Reference

5.12 run_file.cc File Reference

```
#include "RandomChains.h"
```

Include dependency graph for run_file.cc:



Functions

- int main ()

5.12.1 Function Documentation

5.12.1.1 main()

```
int main ( )
```

This function should be invoked to run the program RandomChains.

See also

RandomChains::RandomChains()
RandomChains::SetDecayChains()
RandomChains::Run()

Definition at line 17 of file run_file.cc.

Index