My Project

Generated by Doxygen 1.9.3

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

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 ChiSquare Class Reference

provides methods for calculating P-value for a given distribution and sample.

`#include "src/chisquare.h"`

Public Member Functions

- ChiSquare (NB_Distribution &distribution, const NB_Generator &generator)
- void set_data (NB_Distribution &distribution, const NB_Generator &generator)
- std::vector< double > **get_theoretical_frequency** () const
- std::vector< int > **get_empirical_frequency** () const
- std::vector< double > **get_theoretical_frequency_grouped** ()
- std::vector< int > **get_empirical_frequency_grouped** ()
- double **get_test_stat** ()
- int **get_degree_of_freedom** ()
- double **get_p_value** ()

4.1.1 Detailed Description

provides methods for calculating P-value for a given distribution and sample.

Author

Egor Tkachenko `tkachenko.egor.a@gmail.com`

Version

1.0

Examples

main.cpp.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 ChiSquare()

```
ChiSquare::ChiSquare (
            NB_Distribution & distribution,
            const NB_Generator & generator )
```

Constructor

Parameters

in	*distribution*	The NB_Distribution object, stores distribution function
in	*generator*	The NB_Generator object, stores sample

4.1.3 Member Function Documentation

4.1.3.1 set_data()

```
void ChiSquare::set_data (
            NB_Distribution & distribution,
            const NB_Generator & generator )
```

set new distribution and sample

Parameters

in	*distribution*	The NB_Distribution object, stores distribution function
in	*generator*	The NB_Generator object, stores sample

Examples

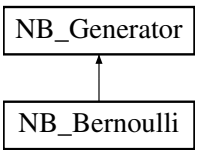
main.cpp.

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

- src/chisquare.h
- src/chisquare.cpp

4.2 NB_Bernoulli Class Reference

Inheritance diagram for NB_Bernoulli:



Public Member Functions

- **NB_Bernoulli** (NB_Distribution *distribution, int sample_size, std::mt19937 &rand_gen)
- int Generate () override

Additional Inherited Members

4.2.1 Detailed Description

Examples

main.cpp.

4.2.2 Member Function Documentation

4.2.2.1 Generate()

```
int NB_Bernoulli::Generate ( )  [override], [virtual]
```

Implements NB_Generator.

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

- src/nb_bernoulli.h
- src/nb_bernoulli.cpp

4.3 NB_Distribution Class Reference

Public Member Functions

- **NB_Distribution** (double p, int k)
- double **get_p** () const
- int **get_k** () const
- std::vector< double > **get_density** ()
- std::string **get_name** () const
- int **get_size** () const
- void **ExtendDensity** (int len=0)

4.3.1 Detailed Description

Examples

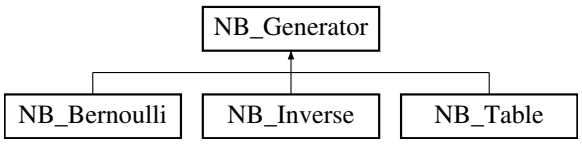
main.cpp.

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

- src/nb_distribution.h
- src/nb_distribution.cpp

4.4 NB_Generator Class Reference

Inheritance diagram for NB_Generator:



Public Member Functions

- **NB_Generator** (int sample_size, std::mt19937 &rand_gen)
- virtual int **Generate** ()=0
- int * **GenerateSample** ()
- int * **get_sample** () const
- int **get_sample_size** () const

Protected Attributes

- std::mt19937 & **rand_gen_**
- int * **sample_**
- int **sample_size_**

4.4.1 Detailed Description

Examples

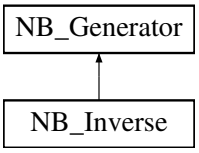
main.cpp.

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

- src/nb_generator.h
- src/nb_generator.cpp

4.5 NB_Inverse Class Reference

Inheritance diagram for NB_Inverse:



Public Member Functions

- **NB_Inverse** (NB_Distribution *distribution, int sample_size, std::mt19937 &rand_gen)
- int Generate () override

Additional Inherited Members

4.5.1 Detailed Description

Examples

main.cpp.

4.5.2 Member Function Documentation

4.5.2.1 Generate()

```
int NB_Inverse::Generate ( )  [override], [virtual]
```

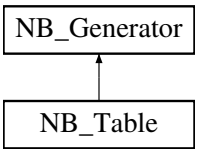
Implements NB_Generator.

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

- src/nb_inverse.h
- src/nb_inverse.cpp

4.6 NB_Table Class Reference

Inheritance diagram for NB_Table:



Public Member Functions

- **NB_Table** (NB_Distribution *distribution, int sample_size, std::mt19937 &rand_gen)
- int Generate () override

Additional Inherited Members

4.6.1 Detailed Description

Examples

main.cpp.

4.6.2 Member Function Documentation

4.6.2.1 Generate()

```
int NB_Table::Generate ( )  [override], [virtual]
```

Implements NB_Generator.

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

- src/nb_table.h
- src/nb_table.cpp

Chapter 5

File Documentation

5.1 chisquare.h

```
1 #ifndef CHISQUARE_H
2 #define CHISQUARE_H
3
4 #include <vector>
5 #include "nb_distribution.h"
6 #include "nb_generator.h"
7 #include "probdist.h"
8
16 class ChiSquare
17 {
18 public:
24     ChiSquare(NB_Distribution& distribution, const NB_Generator& generator);
30     void set_data(NB_Distribution& distribution, const NB_Generator& generator);
31
32     std::vector<double> get_theoretical_frequency() const;
33     std::vector<int> get_empirical_frequency() const;
34     std::vector<double> get_theoretical_frequency_grouped();
35     std::vector<int> get_empirical_frequency_grouped();
36     double get_test_stat();
37     int get_degree_of_freedom();
38     double get_p_value();
39
40 private:
41     std::vector<double> theoretical_frequency_;
42     std::vector<int> empirical_frequency_;
43     std::vector<double> theoretical_frequency_grouped_;
44     std::vector<int> empirical_frequency_grouped_;
45     double test_stat_;
46     int degree_of_freedom_;
47     double p_value_;
48     bool grouped_is_relevant_;
49     bool stats_are_relevant_;
50
51     void group();
52     void calc_p_value();
53 };
54
55 #endif // CHISQUARE_H
```

5.2 nb_bernoulli.h

```
1 #ifndef NB_BERNOULLI_H
2 #define NB_BERNOULLI_H
3
4 #include "nb_generator.h"
5 #include "nb_distribution.h"
6
7 class NB_Bernoulli : public NB_Generator
8 {
9 public:
10     NB_Bernoulli(NB_Distribution* distribution, int sample_size, std::mt19937& rand_gen);
11     ~NB_Bernoulli();
12     int Generate() override;
13
```

```
14 private:
15     int k_;
16     double p_;
17 };
18
19 #endif // NB_BERNOULLI_H
```

5.3 nb_distribution.h

```
1 #ifndef NB_DISTRIBUTION_H
2 #define NB_DISTRIBUTION_H
3
4 #include <vector>
5 #include <string>
6 #include "math.h"
7
8
9 class NB_Distribution
10 {
11 public:
12     NB_Distribution(double p, int k);
13     ~NB_Distribution();
14     double get_p() const;
15     int get_k() const;
16     std::vector<double> get_density();
17     std::string get_name() const;
18     int get_size() const;
19     void ExtendDensity(int len = 0);
20
21 private:
22     double p_;
23     int k_;
24     std::vector<double> density_;
25     double residual_;
26     // 10^6 = max sample_size, EPS * sample_size < 5
27     const double EPS = 1e-6;
28 };
29
30 #endif // NB_DISTRIBUTION_H
```

5.4 nb_generator.h

```
1 #ifndef NB_GENERATOR_H
2 #define NB_GENERATOR_H
3
4 #include <vector>
5 #include <cstdlib>
6 #include <random>
7
8 class NB_Generator
9 {
10 public:
11     NB_Generator(int sample_size, std::mt19937& rand_gen);
12     virtual ~NB_Generator();
13
14     virtual int Generate()=0;
15     int* GenerateSample();
16     int* get_sample() const;
17     int get_sample_size() const;
18
19 protected:
20     std::mt19937& rand_gen_;
21     int* sample_;
22     int sample_size_;
23 };
24
25 #endif // NB_GENERATOR_H
```

5.5 nb_inverse.h

```
1 #ifndef NB_INVERSE_H
2 #define NB_INVERSE_H
3
4 #include "nb_generator.h"
5 #include "nb_distribution.h"
```

```
6
7 class NB_Inverse : public NB_Generator
8 {
9 public:
10     NB_Inverse(NB_Distribution *distribution, int sample_size, std::mt19937& rand_gen);
11     ~NB_Inverse();
12     int Generate() override;
13
14 private:
15     NB_Distribution *distribution_;
16 };
17
18 #endif // NB_INVERSE_H
```

5.6 nb_table.h

```
1 #ifndef NB_TABLE_H
2 #define NB_TABLE_H
3
4 #include "nb_generator.h"
5 #include "nb_distribution.h"
6
7 class NB_Table : public NB_Generator
8 {
9 public:
10     NB_Table(NB_Distribution *distribution, int sample_size, std::mt19937& rand_gen);
11     ~NB_Table();
12     int Generate() override;
13
14 private:
15     std::vector<double> table_;
16 };
17
18 #endif // NB_TABLE_H
```

5.7 probdist.h

```
1
2 void NORMAL( int type, double &x, double &p);
3 double pNormal(double x);
4 double xNormal(double prob);
5 void  CHI( int type, double n, double &x, double &p);
6 double pChi(double x, int n);
7 double xChi(double prob, int n);
```

Chapter 6

Example Documentation

6.1 main.cpp

```
#include "../src/nb_bernoulli.h"
#include "../src/nb_table.h"
#include "../src/nb_inverse.h"
#include "../src/chisquare.h"
// #include "libLibProject"
#include <ctime>
#include <algorithm>
#include <iostream>
using namespace std;
int main() {
    std::mt19937 rand_gen(time(nullptr));

    vector<double> density;
    // How to get negative binomial distribution
    NB_Distribution distribution_0(0.3, 10);
    density = distribution_0.get_density();

    // How to get first k probabilities of negative binomial distribution
    int k = 10;
    NB_Distribution distribution_1(0.2, 10);
    distribution_1.ExtendDensity(k);
    density = distribution_1.get_density();
    cout « density.size() « endl; // 10
    // To get full density call ExtendDensity() without args
    distribution_1.ExtendDensity();
    density = distribution_1.get_density();
    cout « density.size() « endl; // 144
    // How to generate sample
    NB_Generator *generator;
    enum class Method{
        Bernulli,
        Table,
        Inverse
    } method;
    // choose method
    method = Method::Bernulli;
    switch (method)
    {
    case Method::Bernulli:
        generator = new NB_Bernoulli(&distribution_1, 100, rand_gen);
        break;
    case Method::Table:
        generator = new NB_Table(&distribution_1, 100, rand_gen);
        break;
    case Method::Inverse:
        generator = new NB_Inverse(&distribution_1, 100, rand_gen);
        break;
    }
    generator->GenerateSample();
    // get sample
    // shallow copy
    int *sample = generator->get_sample();
    // use sample
    for (int i = 0; i < generator->get_sample_size(); ++i) {
        cout « sample[i] « " ";
    }
    cout « endl;
    // How to use chi square test
```

```
    // create object
    ChiSquare chi_square(distribution_0, *generator);
    // call needed getters
    cout << "degree of freedom = " << chi_square.get_degree_of_freedom() << endl;
    cout << "test statistic = " << chi_square.get_test_stat() << endl;
    cout << "p-value = " << chi_square.get_p_value() << endl;
    // reuse object
    chi_square.set_data(distribution_1, *generator);
    generator->GenerateSample();
    cout << "p-value = " << chi_square.get_p_value() << endl;
    delete generator;
    return 0;
}
```

Index