Chi square test

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
- 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 )
```

Parameters

distribution	NB_Distribution object stores probability function.
generator	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

distribution	NB_Distribution object stores probability function.
generator	NB_Generator object stores sample.

Examples

main.cpp.

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

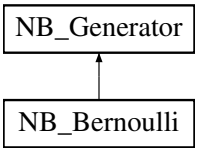
- src/chisquare.h
- src/chisquare.cpp

4.2 NB_Bernoulli Class Reference

Derived class for generating a negative binomial random value using the Bernoulli method.

```
#include "src/nb_bernoulli.h"
```

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

Derived class for generating a negative binomial random value using the Bernoulli method.

Author

Egor Tkachenko tkachenko.egor.a@gmail.com

Version

1.0

Examples

main.cpp.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 NB_Bernoulli()

```
NB_Bernoulli::NB_Bernoulli (
            NB_Distribution * distribution,
            int sample_size,
            std::mt19937 & rand_gen )
```

Parameters

distribution	pointer to the NB_Distribution object to get from it parameters for generation.
sample_size	size of the sample to be generated.
rand_gen	mt19937 random number generator.

4.2.3 Member Function Documentation

4.2.3.1 Generate()

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

Overridden method that implements generation using Bernoulli method.

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

Calculates the probability function of negative binomial distribution for given parameters.

```
#include "src/nb_distribution.h"
```

Public Member Functions

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

4.3.1 Detailed Description

Calculates the probability function of negative binomial distribution for given parameters.

Author

Egor Tkachenko tkachenko.egor.a@gmail.com

Version

1.0

Examples

main.cpp.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 NB_Distribution()

```
NB_Distribution::NB_Distribution (
            double p,
            int k )
```

Parameters

p	probability of success in Bernoulli trails.
k	predefined number of successes.

4.3.3 Member Function Documentation

4.3.3.1 ExtendProbabilities()

```
void NB_Distribution::ExtendProbabilities (
            int len = 0 )
```

Calculates the probabilities for the following len values or until the probability of the remaining ones is less than 10^-6.

Parameters

len	indicates how many next probabilities to calculate. If 0, calculate until the probability of the remaining ones is less than 10^-6.

Examples

main.cpp.

4.3.3.2 get_probabilities()

```
std::vector< double > NB_Distribution::get_probabilities ( )
```

Return calculated probabilities. If no probabilities have been calculated, calls ExtendProbabilities().

Examples

main.cpp.

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

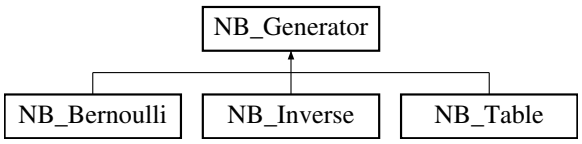
- src/nb_distribution.h
- src/nb_distribution.cpp

4.4 NB_Generator Class Reference

Base class for generating a sample of negative binomial distribution.

`#include "src/nb_generator.h"`

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

Base class for generating a sample of negative binomial distribution.

Author

Egor Tkachenko tkachenko.egor.a@gmail.com

Version

1.0

Examples

main.cpp.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 NB_Generator()

```
NB_Generator::NB_Generator (
            int sample_size,
            std::mt19937 & rand_gen )
```

Parameters

sample_size	size of the sample to be generated.
rand_gen	mt19937 random number generator.

4.4.3 Member Function Documentation

4.4.3.1 Generate()

```
virtual int NB_Generator::Generate ( )  [pure virtual]
```

Virtual method for different implementations of random value generation.

Implemented in NB_Bernoulli, NB_Inverse, and NB_Table.

4.4.3.2 GenerateSample()

```
int * NB_Generator::GenerateSample ( )
```

Method generating sample with Generate().

Examples

main.cpp.

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

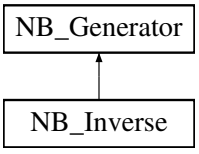
- src/nb_generator.h
- src/nb_generator.cpp

4.5 NB_Inverse Class Reference

Derived class for generating a negative binomial random value using the inverse CDF method.

```
#include "src/nb_inverse.h"
```

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

Derived class for generating a negative binomial random value using the inverse CDF method.

Author

Egor Tkachenko tkachenko.egor.a@gmail.com

Version

1.0

Examples

main.cpp.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 NB_Inverse()

```
NB_Inverse::NB_Inverse (
            NB_Distribution * distribution,
            int sample_size,
            std::mt19937 & rand_gen )
```

Parameters

distribution	pointer to the NB_Distribution object to get from it parameters for generation.
sample_size	size of the sample to be generated.
rand_gen	mt19937 random number generator.

4.5.3 Member Function Documentation

4.5.3.1 Generate()

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

Overridden method that implements generation using inverse CDF method.

Implements NB_Generator.

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

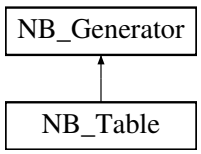
- src/nb_inverse.h
- src/nb_inverse.cpp

4.6 NB_Table Class Reference

Derived class for generating a negative binomial random value using the table method.

```
#include "src/nb_table.h"
```

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

Derived class for generating a negative binomial random value using the table method.

Author

Egor Tkachenko tkachenko.egor.a@gmail.com

Version

1.0

Examples

main.cpp.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 NB_Table()

```
NB_Table::NB_Table (
            NB_Distribution * distribution,
            int sample_size,
            std::mt19937 & rand_gen )
```

Parameters

distribution	pointer to the NB_Distribution object to get from it parameters for generation.
sample_size	size of the sample to be generated.
rand_gen	mt19937 random number generator.

4.6.3 Member Function Documentation

4.6.3.1 Generate()

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

Overridden method that implements generation using table method.

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:
23     ChiSquare(NB_Distribution& distribution, const NB_Generator& generator);
29     void set_data(NB_Distribution& distribution, const NB_Generator& generator);
30
31     std::vector<double> get_theoretical_frequency() const;
32     std::vector<int> get_empirical_frequency() const;
33     double get_test_stat();
34     int get_degree_of_freedom();
35     double get_p_value();
36
37 private:
38     std::vector<double> theoretical_frequency_;
39     std::vector<int> empirical_frequency_;
40     std::vector<double> theoretical_frequency_grouped_;
41     std::vector<int> empirical_frequency_grouped_;
42     double test_stat_;
43     int degree_of_freedom_;
44     double p_value_;
45     bool stats_are_relevant_;
46
47     void group();
48     void calc_p_value();
49 };
50
51 #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
14 class NB_Bernoulli : public NB_Generator
15 {
16 public:
22     NB_Bernoulli(NB_Distribution* distribution, int sample_size, std::mt19937& rand_gen);
23     ~NB_Bernoulli();
27     int Generate() override;
28
29 private:
30     int k_;
31     double p_;
32 };
33
34 #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
15 class NB_Distribution
16 {
17 public:
22     NB_Distribution(double p, int k);
23     ~NB_Distribution();
24     double get_p() const;
25     int get_k() const;
30     std::vector<double> get_probabilities();
31     std::string get_name() const;
32     int get_probabilities_size() const;
39     void ExtendProbabilities(int len = 0);
40
41 private:
42     double p_;
43     int k_;
44     std::vector<double> probabilities_;
45     double residual_;
46     // 10^6 = max sample_size, EPS * sample_size < 5
47     const double EPS = 1e-6;
48 };
49
50 #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
15 class NB_Generator
16 {
17 public:
22     NB_Generator(int sample_size, std::mt19937& rand_gen);
23     virtual ~NB_Generator();
27     virtual int Generate()=0;
31     int* GenerateSample();
32     int* get_sample() const;
33     int get_sample_size() const;
34
35 protected:
36     std::mt19937& rand_gen_;
37     int* sample_;
38     int sample_size_;
39 };
40
41 #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
14 class NB_Inverse : public NB_Generator
15 {
16 public:
22     NB_Inverse(NB_Distribution *distribution, int sample_size, std::mt19937& rand_gen);
23     ~NB_Inverse();
27     int Generate() override;
28
29 private:
30     NB_Distribution *distribution_;
31 };
32
33 #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
14 class NB_Table : public NB_Generator
15 {
16 public:
22     NB_Table(NB_Distribution *distribution, int sample_size, std::mt19937& rand_gen);
23     ~NB_Table();
27     int Generate() override;
28
29 private:
30     std::vector<double> table_;
31 };
32
33 #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 <ctime>
#include <algorithm>
#include <iostream>
using namespace std;
int main() {
    std::mt19937 rand_gen(time(nullptr));

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

    // How to get first k probabilities of negative binomial distribution
    int k = 10;
    NB_Distribution distribution_1(0.2, 10);
    distribution_1.ExtendProbabilities(k);
    probabilities = distribution_1.get_probabilities();
    cout « probabilities.size() « endl; // 10
    // To get full probabilities call ExtendDensity() without args
    distribution_1.ExtendProbabilities();
    probabilities = distribution_1.get_probabilities();
    cout « probabilities.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