Проверка гипотез

Создано системой Doxygen 1.14.0

Глава 1

Иерархический список классов

1.1 Иерархия классов

Иерархия классов.

Глава 2

Алфавитный указатель классов

2.1 Классы

Классы с их кратким описанием.

Глава 3

Список файлов

3.1 Файлы

Полный список документированных файлов.

Глава 4

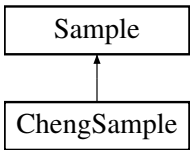
Классы

4.1 Класс ChengSample

Хранение и моделирование выборки методом Ченга.

#include <ChengSample.h>

Граф наследования:ChengSample:



Открытые члены

- ChengSample (int mm)
- void simulate (const Distribution &distr, int sz, std::mt19937 &gen) override

Открытые члены унаследованные от Sample

- int at (int i) const
- int size () const
- int getNstates () const

Дополнительные унаследованные члены

Защищенные данные унаследованные от Sample

- int * sample
- int nstates

4.1.1 Подробное описание

Хранение и моделирование выборки методом Ченга.

В конструкторе класса ChengSample передаётся значение параметра m, используемого для моделирования выборки методом Ченга.

Примеры

Examples.cpp.

4.1.2 Конструктор(ы)

4.1.2.1 ChengSample()

```
ChengSample::ChengSample (
            int mm)   [inline]
```

Конструктор ChengSample::ChengSample(mm) принимает на вход целочисленный положительный параметр mm, используемый при моделировании выборки в функции ChengSample::simulate.

4.1.3 Методы

4.1.3.1 simulate()

```
void ChengSample::simulate (
            const Distribution & distr,
            int sz,
            std::mt19937 & gen)   [override], [virtual]
```

См. Sample::simulate. Для моделирования распределения distr используется метод Ченга.

Замещает Sample.

Объявления и описания членов классов находятся в файлах:

- ChengSample.h
- ChengSample.cpp

4.2 Класс ChiSquare

Критерий хи-квадрат.

#include <ChiSquare.h>

Открытые члены

- ChiSquare (Sample *sample, Distribution &distr)
- void set (Sample *sample, Distribution &distr)
- double get_pvalue () const
- double get_statistic () const
- int get_df () const
- const int * get_emp_freq () const
- const double * get_theor_freq () const
- int getNstates () const

Защищенные члены

- void calculate ()

Защищенные данные

- int * emp_freq
- double * theor_freq
- double t
- double pvalue
- int df
- int nstates

4.2.1 Подробное описание

Критерий хи-квадрат.

На вход (в конструкторе или функции ChiSquare::set) подаётся выборка (указатель на Sample) и распределение (объект типа Distribution), на согласие с которым проверяется выборка.

После задания выборки и распределения вычисляется значение статистики критерия, число степеней свободы и значаение p-value. При вычислении происходит группировка состояний, для удовлетворения условиям применимости критерия хи-квадрат, а именно: все эмпирические частоты сгруппированных состояний должны быть не меньше 5.

Важно: изменение выборки не повлияет на содержимое объекта класса ChiSquare.

Примеры

Examples.cpp.

4.2.2 Конструктор(ы)

4.2.2.1 ChiSquare()

```
ChiSquare::ChiSquare (
            Sample * sample,
            Distribution & distr)
```

На вход конструктора ChiSquare::ChiSquare(sample, distr) подаётся указатель sample (типа Sample*) на выборку, а также распределение distr типа Distribution, на согласие с которым проверяется выборка. После, вычисляется статистика критерия, число степеней свободы, и значение p-value.

Объявления и описания членов классов находятся в файлах:

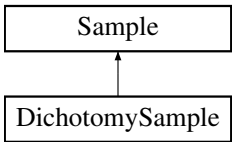
- ChiSquare.h
- ChiSquare.cpp

4.3 Класс DichotomySample

Хранение и моделирование выборки методом дихотомии.

#include <DichotomySample.h>

Граф наследования:DichotomySample:



Открытые члены

- void simulate (const Distribution &distr, int sz, std::mt19937 &gen) override

Открытые члены унаследованные от Sample

- int at (int i) const
- int size () const
- int getNstates () const

Дополнительные унаследованные члены

Защищенные данные унаследованные от Sample

- int * sample
- int nstates

4.3.1 Подробное описание

Хранение и моделирование выборки методом дихотомии.

Примеры

Examples.cpp.

4.3.2 Методы

4.3.2.1 simulate()

```
void DichotomySample::simulate (
            const Distribution & distr,
            int sz,
            std::mt19937 & gen)   [override], [virtual]
```

См. Sample::simulate. Для моделирования распределения distr используется метод дихотомии.

Замещает Sample.

Объявления и описания членов классов находятся в файлах:

- DichotomySample.h
- DichotomySample.cpp

4.4 Класс Distribution

Дискретное распределение.

#include <Distribution.h>

Открытые члены

- Distribution (const double *weights, int n)
- double get_p (int k) const
- void set (const double *weights, int n)
- int getNstates () const

4.4.1 Подробное описание

Дискретное распределение.

Класс Distribution хранит произвольное дискретное распределение с конечным числом состояний. На вход (в конструкторе или функции Distribution::set) подаётся массив с весами состояний и число состояний. При помощи функции Distribution::get_p(n) можно получить вероятности с котороми случайная величина принимает заданное значение.

Примеры

Examples.cpp.

4.4.2 Конструктор(ы)

4.4.2.1 Distribution()

```
Distribution::Distribution (
        const double * weights,
        int n)
```

На вход подаётся массив weights типа double и длины n, в котором находятся веса значений случайной величины.

4.4.3 Методы

4.4.3.1 get_p()

```
double Distribution::get_p (
        int k) const    [inline]
```

Функция Distribution::get_p(k) возвращает вероятность p_k того, что случайная величина с данным распределением примет значение k. Если $(w_0, \ldots, w_{n-1})$ — веса, то $p_k = w_k / \sum_{i=0}^{n-1} w_i$.

Примеры

Examples.cpp.

Объявления и описания членов классов находятся в файлах:

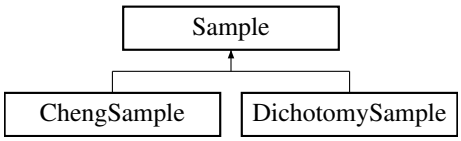
- Distribution.h
- Distribution.cpp

4.5 Класс Sample

Хранения и моделирование выборки.

#include <Sample.h>

Граф наследования:Sample:



Открытые члены

- virtual void simulate (const Distribution &distr, int sz, std::mt19937 &gen)=0
- int at (int i) const
- int size () const
- int getNstates () const

Защищенные данные

- int * sample
- int nstates

4.5.1 Подробное описание

Хранения и моделирование выборки.

Sample — виртуальный класс, в котором хранится и моделируется целочисленная выборка. Выборка хранится в сгруппированном виде. Метод моделированния определяется в наследнике.

Примеры

Examples.cpp.

4.5.2 Методы

4.5.2.1 at()

int Sample::at (
int i) const [inline]

Функция Sample::at даёт безопасный доступ к элементам сгруппированной выборки. Sample::at(i) возвращает количество значений в выборке равных i.

Примеры

Examples.cpp.

4.5.2.2 getNstates()

int Sample::getNstates () const [inline]

Функция Sample::getNstates() возвращает число возможных состояний: гарантируется, что все значения в выборке лежат в пределах от 0 до Sample::getNstates()-1.

4.5.2.3 simulate()

virtual void Sample::simulate (
const Distribution & distr,
int sz,
std::mt19937 & gen) [pure virtual]

Функция Sample::simulate моделирует выборку размера sz с распределением описанным в distr, используя генератор случайных чисел gen.

Замещается в ChengSample и DichotomySample.

Примеры

Examples.cpp.

4.5.2.4 size()

int Sample::size () const

Функция Sample::size возвращает размер выборки.

Объявления и описания членов классов находятся в файлах:

- Sample.h
- Sample.cpp

Глава 5

Файлы

5.1 ChengSample.h

```
00001 #pragma once
00002 #include "Sample.h"
00003
00005
00007 class ChengSample : public Sample {
00008 public:
00009     ChengSample(int mm) : Sample(), m(mm) {}
00011     void simulate(const Distribution& distr, int sz, std::mt19937& gen) override;
00013 private:
00014     int m;
00015 };
00016
```

5.2 ChiSquare.h

```
00001 #pragma once
00002 #include "Sample.h"
00003 #include "Distribution.h"
00004
00006
00014 class ChiSquare{
00015 public:
00016     ChiSquare() : emp_freq(nullptr), theor_freq(nullptr), t(0), pvalue(0), df(0), nstates(0) {}
00017     ChiSquare(Sample* sample, Distribution& distr);
00021     ~ChiSquare();
00022     void set(Sample* sample, Distribution& distr);
00023     double get_pvalue() const { return pvalue; }
00024     double get_statistic() const { return t; }
00025     int get_df() const { return df; }
00026     const int* get_emp_freq() const { return emp_freq; }
00027     const double* get_theor_freq() const { return theor_freq; }
00028     int getNstates() const { return nstates; }
00029
00030
00031 protected:
00032     void calculate();
00033     int* emp_freq;
00034     double* theor_freq;
00035     double t;
00036     double pvalue;
00037     int df;
00038     int nstates;
00039 };
00040
```

5.3 DichotomySample.h

```
00001 #pragma once
00002 #include "Sample.h"
```

```
00003
00005 class DichotomySample : public Sample {
00006 public:
00007     void simulate(const Distribution& distr, int sz, std::mt19937& gen) override;
00009 };
00010
```

5.4 Distribution.h

```
00001 #pragma once
00002
00004
00009 class Distribution
00010 {
00011 public:
00012     Distribution() : probs(nullptr), nstates(0) {}
00013     Distribution(const double* weights, int n);
00015     ~Distribution() { delete[] probs; }
00016     double get_p(int k) const
00020     {
00021         if (k >= 0 && k < nstates)
00022             return probs[k];
00023         return 0;
00024     }
00025     void set(const double* weights, int n);
00026     int getNstates() const { return nstates; }
00027 private:
00028     int nstates;
00029     double* probs;
00030 };
00031
```

5.5 Sample.h

```
00001 #pragma once
00002 #include <random>
00003 #include "Distribution.h"
00004
00006
00009 class Sample {
00010 public:
00011     Sample() : sample(nullptr), nstates(0) {}
00012     virtual ~Sample() { delete[] sample; }
00013     virtual void simulate(const Distribution& distr, int sz, std::mt19937& gen) = 0;
00015     int at(int i) const { if (i >= 0 && i < nstates) return sample[i]; return 0; }
00017     int size() const;
00019     int getNstates() const { return nstates; }
00021
00022 protected:
00023     int* sample;
00024     int nstates;
00025 };
```

Глава 6

Примеры

6.1 Examples.cpp

```
// Examples.cpp : This file contains the 'main' function. Program execution begins and ends there.
//

#include <iostream>
#include <random>
#include "../BigProgram/DichotomySample.h"
#include "../BigProgram/ChengSample.h"
#include "../BigProgram/ChiSquare.h"

int main() {
    double weights1[3] = { 1.0, 2.0, 3.0 };
    Distribution distr1(weights1, 3);
    std::cout << "Create distribution P_1 with weights:\n";
    for (int i = 0; i < 3; ++i)
        std::cout << i << ": " << weights1[i] << ";\t";
    std::cout << "\nProbabilities:\n";
    for (int i = 0; i < 3; ++i)
        std::cout << i << ": " << distr1.get_p(i) << ";\t";

    std::mt19937 gen(42);

    Sample* sample;
    sample = new DichotomySample;
    sample->simulate(distr1, 100, gen);
    std::cout << "\n\nSample of distribution P_1 (dichotomy simulation method):\n";
    for (int i = 0; i < 3; ++i)
        std::cout << i << ": " << sample->at(i) << ";\t";
    delete sample;

    sample = new ChengSample(4);
    sample->simulate(distr1, 100, gen);
    std::cout << "\n\nSample (1) of distribution P_1 (Cheng simulation method with m=4):\n";
    for (int i = 0; i < 3; ++i)
        std::cout << i << ": " << sample->at(i) << ";\t";

    double weights0[] = { 1,3,3 };
    Distribution distr0(weights0, 3);
    std::cout << "\n\nCreate distribution P_0 with weights:\n";
    for (int i = 0; i < 3; ++i)
        std::cout << i << ": " << weights0[i] << ";\t";
    std::cout << "\nProbabilities:\n";
    for (int i = 0; i < 3; ++i)
        std::cout << i << ": " << distr0.get_p(i) << ";\t";

    std::cout << "\n\nApply chi-square criteria to Sample (1) and distribution P_0 (H_0 hypothesis):\n";
    ChiSquare chisq(sample, distr0);
    std::cout << "statistic: " << chisq.get_statistic() << "\t df: " << chisq.get_df() << "\t p-value: " << chisq.get_pvalue();

    delete sample;
}
```

Предметный указатель