Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Новосибирский государственный технический университет»

Кафедра теоретической и прикладной информатики

КУРСОВОЙ ПРОЕКТ

по теме «Проверка гипотезы о виде распределения по критерию Хи-квадрат Пирсона»

курса «Компьютерные технологии моделирования и анализа данных»

Выполнил:

студент ФПМИ,

гр. ПММ-01

Кастин В. С.

Проверил:

Профессор каф. ТПИ

Постовалов С. Н.

Дата: 21.12.2020

Новосибирск 2020

**Содержание**

[Введение 3](#_Toc59547982)

[Постановка задачи 4](#_Toc59547983)

[Аналитический обзор 6](#_Toc59547984)

[Результаты исследований 7](#_Toc59547985)

[Вывод 9](#_Toc59547986)

[Список использованных источников 10](#_Toc59547987)

[Код программы 11](#_Toc59547988)

# Введение

В современной обществе статистическая проверка гипотез является одной из важнейших задач как в научной области, так и во многих других областях. В зависимости от гипотезы используются различные инструменты для её проверки, и одними из таких являются критерии согласия, где одним из самых популярных считается критерий χ2 Пирсона

Критерий χ2 Пирсона - это непараметрический критерий согласия, который позволяет оценить значимость различий между эмпирическими данными, попадающими в предварительно определённые интервалы, и теоритическими данными, имеющих ожидаемую частоту попадания в интервалы при справедливой нулевой гипотезе. Одним из наиболее частых способов использования данного критерия является проверка гипотезы о принадлежности выборки к определённому виду распределения.

Целью данной работы является разработка критерия χ2 Пирсона и проверка нулевой гипотезе о принадлежности выборок к определённому закону распределения с помощью разработанного критерия.

# Постановка задачи

Для проверки согласия различают простые и сложные гипотезы.

Простая гипотеза представляет из себя следующее:

,

где *f(x, θu)* – функция плотности, а *θu* – известный скалярный или векторный параметр теоритического распределения, с которым проверяют согласие.

Первым этапом при использовании критерия χ2 Пирсона для простой гипотезы является группирование эмпирической выборки. Вся выборка разбивается на *k* интервалов граничными точками.

Далее рассчитывается *ni* значений, попавших в *i*-й интервал, а также вероятность попадания в интервал, равная функции теоритического распределения.

Затем вычисляется статистика критерия χ2 Пирсона по следующей формуле:



При справедливости нулевой гипотезы предельным распределением статистики является -распределение с числом степеней свободы *r = k – 1*, и, если по выборке оценивалось p параметров, то *r = k – p – 1*.

При сложной гипотезе, при условии, что оценки параметров находятся в результате минимизации статистики по этой же самой выборке, статистика асимптотически распределена как с числом степеней свободы *r = k – p – 1.*

Статистика имеет это же распределение, если в качестве метода оценивания выбирают метод максимального правдоподобия и оценки вычисляют по сгруппированным данных в результате максимизации по *θ* функции правдоподобия [1].

После получения статистики находится достигнутый уровень значимости, он же *p-value*, который находится либо с помощью таблицы распределения χ2 Пирсона, либо с помощью метода Монте –Карло.

В свою очередь, достигнутый уровень значимости сравнивается с заданным уровнем значимости, он же *α-value*, который чаще всего задаётся в диапазоне от *0.005* до *0.1.* И, по итогу сравнения, если достигнутый уровень значимости больше заданного, то нулевая гипотеза принимается, в обратном же случае гипотеза отклоняется. Это говорит о том, входит ли полученное значение статистики в критическую область или же нет.

По рекомендациям ВНИИМетрологии [2] количество наблюдений, попадающих в интервал должно быть не менее 5-10.

Кроме того, в ходе выполнения лабораторных работ по курсу «Компьютерные технологии моделирования и анализа данных» было выяснено, что общее количество наблюдений должно быть не менее 50, а лучше не менее 100.

# Аналитический обзор

До разработки критерия χ2 полагалось, что экспериментальные данные подчиняются нормальному распределению, однако Карл Пирсон, английский математик, заметил, что некоторые результаты измерений не укладываются в нормальное распределение, а подчиняются каким-то другим распределениям. Примером может служить игральная кость, результаты которой будут стремиться к равномерному закону распределения.

В 1900 году Карл Пирсон разработал и предложил критерий χ2 для анализа таблиц сопряженности.

Спустя 20 лет английский статистик Роналд Фишер заметил, что в формуле критерия χ2 Пирсонаесть зависимое слагаемое и для корректировки стало учитываться такое понятие как «степень свободы» - оно же число независимых слагаемых [3].

Первая научная публикация по критерию χ2 выпущена самим Карлом Пирсоном в журнале «Philosophical Magazine» под названием «On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be seasonably supposed to have arisen from random sampling» [4], где критерий χ2 был впервые представлен.

Также по данному критерию за время его существования было выпущено множество публикаций, основные из которых, по крайней мере, в русской литературе это публикации Никулина М. С. [6, 7], а также публикации Лемешко Б. Ю., Постовалова С. Н. и Чимитовой Е. В. [8, 9, 10], в которых описывается как сам критерий, так и способы его применения.

# Результаты исследований

Как уже было сказано, целью исследования является проверка нулевой гипотезы о принадлежности эмпирического распределения теоритическому с помощью критерия χ2 Пирсона. В качестве законов распределения для проведения исследования будут использованы экспоненциальное распределения и распределение Рэлея. Размерность эмпирических выборок будет составлять *n = 50, 100 и 1000,* количество интервалов для них будет составлять 8, 10 и 14 соответственно*.* Таким образом, исследование будет проводиться на шести выборках.Значение *N* для метода Монте-Карло (ММК) будет равно 16600. Заданным уровнем значимости для принятия нулевой гипотезы будет *α-value = 0.05.* Проверяться будут простые гипотезы.

Ниже приведена таблица результатов исследования:

|  |  |  |  |
| --- | --- | --- | --- |
| **Экспоненциальное распределение** | **n = 50** | **n = 100** | **n = 1000** |
| Статистика | 2.1931 | 6.29451 | 14.7513 |
| Достигаемый *p-value* (предельное распр. статистики) | 0.05 < p-value < 0.95 | 0.05 < p-value < 0.95 | 0.05 < p-value < 0.95 |
| Достигаемый *p-value* (ММК) | 0.959036 | 0.718072 | 0.36006 |
| Результат проверки | Принимается | Принимается | Принимается |
| **Распределение Рэлея** | **n = 50** | **n = 100** | **n = 1000** |
| Статистика | 5.06425 | 4.47639 | 16.2191 |
| Достигаемый *p-value* (предельное распр. статистики) | 0.05 < p-value < 0.95 | 0.05 < p-value < 0.95 | 0.05 < p-value < 0.95 |
| Достигаемый *p-value* (ММК) | 0.672048 | 0.883916 | 0.270843 |
| Результат проверки | Принимается | Принимается | Принимается |

Далее приведена таблица результатов через проверку в ISW:

|  |  |  |  |
| --- | --- | --- | --- |
| **Экспоненциальное распределение** | **n = 50** | **n = 100** | **n = 1000** |
| Статистика | 2.19313 | 6.29453 | 14.75124 |
| Достигаемый *p-value* | 0.948382 | 0.710115 | 0.32313 |
| Результат проверки | Принимается | Принимается | Принимается |
| **Распределение Рэлея** | **n = 50** | **n = 100** | **n = 1000** |
| Статистика | 5.06453 | 4.47607 | 16.218907 |
| Достигаемый *p-value* | 0.652088 | 0.877385 | 0.237510 |
| Результат проверки | Принимается | Принимается | Принимается |

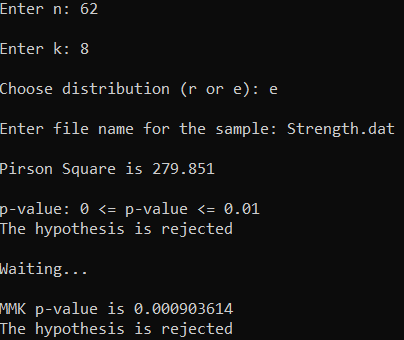
Как видно по результатам таблиц, каждая из нулевых гипотез о том, что эмпирическое распределение принадлежит теоритическому, принимается. Кроме того, по данным из обоих таблиц видно, что значения p-value полученные с помощью ISW, довольно близки к значениям, полученным с помощью ММК, что говорит о правильности разработанной программы.

Далее будет проведена проверка гипотезы на реальных данных, которые представляют собой прочность стекловолокна толщиной 1.5 см., измеренную в национальной физической лаборатории Англии [10]. Будет проверяться гипотеза о принадлежности к экспоненциальному распределению.

Ниже представлена проверяемая выборка объёмом 62:

0.55, 0.93, 1.25, 1.36, 1.49, 1.52, 1.61, 1.64, 1.68, 1.73, 1.81, 2.00, 0.74, 1.04, 1.27, 1.39, 1.49, 1.53, 1.59, 1.61, 1.66, 1.68, 1.76, 1.82, 2.01, 0.77, 1.11, 1.28, 1.42, 1.59, 1.54, 1.60, 1.62, 1.66, 1.69, 1.76, 1.84, 2.24, 0.81, 1.13, 1.29, 1.48, 1.50, 1.55, 1.61, 1.62, 1.66, 1.70, 1.77, 1.84, 0.84, 1.24, 1.30, 1.48, 1.51, 1.55, 1.61, 1.63, 1.67, 1.70, 1.78, 1.89.

Далее приведен скриншот работы программы:



В данном случае нулевая гипотеза была отклонена.

# Вывод

В ходе работы было установлено, что критерий χ2 Пирсона - это непараметрический критерий согласия, который позволяет оценить значимость различий между эмпирическими данными, попадающими в предварительно определённые интервалы, и теоритическими данными, имеющих ожидаемую частоту попадания в интервалы при справедливой нулевой гипотезе.

Также было установлено, что рекомендуемые условия его использования – это выборки не менее 50 наблюдений, и количество наблюдений в каждом интервале не менее 5-10.

Кроме того, была разработана программа реализующая следующий функционал:

* генерация выборок с экспоненциальным законом распределения и законом распределения Рэлея;
* чтение файлов формата dat;
* группирование данных;
* расчёт статистики критерия χ2 Пирсона;
* нахождение p-value с помощью предельного распределения статистики;
* нахождение p-value с помощью метода Монте-Карло;
* проверка нулевой гипотезы о принадлежности эмпирического распределения теоритическому.

В ходе исследования при проверке нулевой гипотезе о при принадлежности эмпирического распределения теоритическому на смоделированных данных гипотеза подтвердилась, а на реальных данных гипотеза была отклонена.

# Список использованных источников

1. Лемешко Б. Ю., Лемешко С. Б., Постовалов С. Н., Чимитова Е. В. Статистический анализ данных, моделирование и исследование вероятностных закономерностей. Компьютерный подход : монография / Б. Ю. Лемешко, С. Б. Лемешко, С. Н. Постовалов, Е. В. Чимитова. – Новосибирск : Изд-но НГТУ, 2011. – 888 с. (серия «Монография НГТУ»).
2. Бурдун Г.Д., Марков Б.Н. Основы метрологии. - М.: Изд-во стандартов, 1985. - 120 с.
3. selfedu (2020) Теория вероятностей #17: критерий хи квадрат (Пирсона) [видео-урок]// YouTube. (https://youtu.be/hvBHyDm4biM) Просмотрено: 21.12.2020.
4. Pearson, Karl (1900). "On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling". Philosophical Magazine. Series 5. 50 (302): 157 – 175.
5. Никулин М. С. Критерий хи-квадрат для непрерывных распределений с параметрами сдвига и масштаба // Теория вероятностей и её применение. — 1973. — Т. XVIII, № 3. — С. 583—591.
6. Никулин М. С. О критерии хи-квадрат для непрерывных распределе¬ний // Теория вероятностей и её применение. — 1973. — Т. XVIII. — № 3. — С. 675—676.
7. Лемешко Б. Ю., Постовалов С. Н. О зависимости предельных распределений статистик χ2 Пирсона и отношения правдоподобия от способа группирования данных // Заводская лаборатория. 1998. Т. 64. — № 5. — С. 56-63.
8. Лемешко Б. Ю., Чимитова Е. В. О выборе числа интервалов в критериях согласия типа χ2 // Заводская лаборатория. Диагностика материалов. 2003. Т. 69. — № 1. — С. 61-67.
9. Лемешко Б. Ю., Лемешко С. Б., Постовалов С. Н. Сравнительный анализ мощности критериев согласия при близких конкурирующих гипотезах. I. Проверка простых гипотез // Сибирский журнал индустриальной математики. 2008. — Т.11. — № 2(34). — С.96-111.
10. Shanker R, Fesshaye H, Selvaraj S. On modeling of lifetimes data using exponential and lindley distributions. Biom Biostat Int J. 2015;2(5):140‒147.

# Код программы

#include <iostream>

#include <random>

#include <chrono>

#include <vector>

#include <algorithm>

#include <cmath>

#include <fstream>

#include <string>

**using** **namespace** std;

//Класс для генерации случайных чисел

**class** **RandomGenerator**

{

**public:**

**static** std::mt19937 & getMt19937();

**private:**

RandomGenerator();

~RandomGenerator() {}

**static** RandomGenerator& instance();

RandomGenerator(RandomGenerator **const**&) = **delete**;

RandomGenerator& **operator**= (RandomGenerator **const**&) = **delete**;

std::mt19937 mMt;

};

RandomGenerator::RandomGenerator() {

std::random\_device rd;

**if** (rd.entropy() != **0**) {

std::seed\_seq seed{ rd(), rd(), rd(), rd(), rd(), rd(), rd(), rd() };

mMt.seed(seed);

}

**else** {

**auto** seed = std::chrono::high\_resolution\_clock::now().time\_since\_epoch().count();

mMt.seed(seed);

}

}

RandomGenerator& RandomGenerator::instance() {

**static** RandomGenerator s;

**return** s;

}

std::mt19937 & RandomGenerator::getMt19937() {

**return** RandomGenerator::instance().mMt;

}

//Мои функции

//Генерация выборки

vector <**double**> sampleGeneration(**int** n)

{

std::mt19937 &mt = RandomGenerator::getMt19937();

std::uniform\_real\_distribution<**double**> dist(**0.0**, **1.0**);

vector <**double**> sample;

sample.resize(n);

**int** i = **0**;

**while** (i < n)

{

sample[i] = dist(mt);

i++;

}

**return** sample;

}

//Моделирование выборки с законом распределения Релея

vector <**double**> modelRayleighRaspr(**int** n)

{

vector <**double**> u1 = sampleGeneration(n);

vector <**double**> u2 = sampleGeneration(n);

vector <**double**> u3 = sampleGeneration(n);

vector <**double**> u4 = sampleGeneration(n);

vector <**double**> x;

x.resize(u1.size());

vector <**double**> y;

y.resize(u3.size());

**for** (**int** i = **0**; i < x.size(); i++)

{

x[i] = pow(-**2.0**\*log(u1[i]), **0.5**) \* cos(**2**\***3.142**\*u2[i]);

y[i] = pow(-**2.0**\*log(u3[i]), **0.5**) \* cos(**2**\***3.142**\*u4[i]);

}

vector <**double**> z;

z.resize(x.size());

**for** (**int** i = **0**; i < z.size(); i++)

{

z[i] = pow(pow(x[i], **2.0**) + pow(y[i], **2.0**), **0.5**);

}

**return** z;

}

//Моделирование экспоненциального распределения

vector <**double**> modelExpRaspr(**int** n, **double** a = **1.0**)

{

vector <**double**> u = sampleGeneration(n);

vector <**double**> x;

x.resize(u.size());

**for** (**int** i = **0**; i < x.size(); i++)

{

x[i] = -(**1**/a) \* log(**1** - u[i]);

}

**return** x;

}

//Функция плотности экспоненциального распределения

**double** expRaspr(**double** x, **double** m = **1.0**, **double** s = **0**)

{

**if**(x >= **0**)

**return** (**1** - exp(-m\*x));

**else** **return** **0**;

}

//Функция плотности распределения Рэлея

**double** relRaspr(**double** x, **double** m = **1.0**, **double** s = **0**)

{

**if**(x >= **0**)

**return** (**1** - exp(-(pow(x, **2**) / (**2** \* pow(m, **2**)))));

**else** **return** **0**;

}

//Критерий ХИ-квадрат Пирсона

**double** pirsonSquare(vector <**double**> &sample, **int** k, **char** r)

{

//Сортировка выборки для группирования

sort(sample.begin(), sample.begin()+sample.size());

//Шаг 1: группирование данных

//Шаг 1.1: размах выборки

**double** max = \*max\_element(sample.begin(), sample.end());

**double** min = \*min\_element(sample.begin(), sample.end());

**double** d = max - min;

**double** h = d / k;

//Шаг 1.2: Разбиение на интервалы

//Вектор для хранения попаданий в каждый интервал

vector <**double**> kLength;

kLength.resize(k);

**for**(**int** i = **0**, j = **0**; i < sample.size()-**1**; i++)

{

**if**(sample[i] <= (sample[**0**] + h))

{

//Если случайная величина попадает в интервал, то частота попадания в данный интервал увеличивается на 1

kLength[j]++;

//Обновление шага до начального значения

h = d / k;

//Сброс j до первого интервала

j = **0**;

}

**else**

{

//Если случайная величина не попадает в интервал, то значение сохраняется

i--;

//Увеличение j до следующего интервала

j++;

//Увеличение h до следующего интервала

h += d / k;

}

}

//Исправление бага с последним элементом в цикле выше

kLength[k-**1**]++;

//Обновление значения шага

h = d / k;

//Шаг 1.3: вероятность попадания в интервал

//Вектор для хранения вероятности

vector <**double**> pInterval;

pInterval.resize(k);

**for**(**int** i = **0**; i < k; i++)

{

pInterval[i] = kLength[i] / sample.size();

}

//Вектор для хранения значений граничных точек

vector <**double**> xInterval;

xInterval.resize(k);

//cout << "Boundary points: " << endl;

xInterval[**0**] = sample[**0**] + h;

**for**(**int** i = **1**; i < k-**1**; i++)

{

xInterval[i] = xInterval[i-**1**] + h;

//if(i == 1)

//cout << xInterval[i-1] << endl;

//cout << xInterval[i] << endl;

}

//cout << endl;

//Шаг 2: Хи-квадрат Пирсона

//Шаг 2.1: нахождение ожидаемой частоты попадания в интервал

//Вектор для хранения вероятности попадания в интервал

vector <**double**> pT;

pT.resize(k);

**double** sum = **0**;

//cout << "Expected frequency of hitting the interval: " << endl;

**switch**(r)

{

**case** 'e':

**for**(**int** i = **0**; i < k; i++)

{

**if**(i == **0**)

pT[i] = expRaspr(xInterval[i]);

**else** **if**(i > **0** && i < k - **1**)

pT[i] = expRaspr(xInterval[i]) - sum;

**else** pT[i] = **1** - sum;

//cout << pT[i] << endl;

sum += pT[i];

}

**break**;

**case** 'r':

**for**(**int** i = **0**; i < k; i++)

{

**if**(i == **0**)

pT[i] = relRaspr(xInterval[i]);

**else** **if**(i > **0** && i < k - **1**)

pT[i] = relRaspr(xInterval[i]) - sum;

**else** pT[i] = **1** - sum;

//cout << pT[i] << endl;

sum += pT[i];

}

**break**;

}

//cout << endl;

//Шаг 2.2: нахождение статистики

**double** pSquare = **0**;

**for**(**int** i = **0**; i < k; i++)

{

pSquare += pow((pInterval[i] - pT[i]), **2**) / pT[i];

}

**return** sample.size() \* pSquare;

}

**double** monteKarlo(**double** xPSquare, **int** n, **int** k, **char** r, **int** N = **16600**)

{

**double** m = **0**;

**double** yPSquare;

cout << "Waiting...**\n\n**";

**switch**(r)

{

**case** 'e':

**for**(**int** i = **0**; i < N; i++)

{

vector <**double**> y = modelExpRaspr(n);

yPSquare = pirsonSquare(y, k, r);

**if**(yPSquare > xPSquare)

m = m + **1**;

}

**break**;

**case** 'r':

**for**(**int** i = **0**; i < N; i++)

{

vector <**double**> y = modelRayleighRaspr(n);

yPSquare = pirsonSquare(y, k, r);

**if**(yPSquare > xPSquare)

m = m + **1**;

}

}

**return** m/N;

}

**void** pValueTab(**double** pSquare, **int** k, **double** aValue = **0.05**)

{

vector <**double**> pValue {**0.01**, **0.025**, **0.05**, **0.95**, **0.975**, **0.99**};

vector <**double**> pStatic7 {**18.5**, **16.0**, **14.1**, **2.17**, **1.69**, **1.24**};

vector <**double**> pStatic9 {**21.7**, **19.0**, **16.9**, **3.33**, **2.7**, **2.09**};

vector <**double**> pStatic13 {**27.7**, **24.7**, **22.4**, **5.89**, **5.01**, **4.11**};

**double** pR;

**double** pL;

**switch**(k)

{

**case** **10**:

**for**(**int** i = **0**; i < pValue.size(); i++)

{

**if**(pSquare >= pStatic9[i])

{

pR = pValue[i];

**if**(i != **0**)

pL = pValue[i-**1**];

**else** pL = **0**;

**break**;

}

}

**break**;

**case** **8**:

**for**(**int** i = **0**; i < pValue.size(); i++)

{

**if**(pSquare >= pStatic7[i])

{

pR = pValue[i];

**if**(i != **0**)

pL = pValue[i-**1**];

**else** pL = **0**;

**break**;

}

}

**break**;

**case** **14**:

**for**(**int** i = **0**; i < pValue.size(); i++)

{

**if**(pSquare >= pStatic13[i])

{

pR = pValue[i];

**if**(i != **0**)

pL = pValue[i-**1**];

**else** pL = **0**;

**break**;

}

}

**break**;

}

cout << "p-value: " << pL << " <= p-value <= " << pR << endl;

**if** (pR <= aValue)

cout << "The hypothesis is rejected**\n\n**";

**else** cout << "The hypothesis is accepted**\n\n**";

}

**void** pValueMMK(**double** pValue, **double** aValue = **0.05**)

{

**if** (pValue > aValue)

cout << "The hypothesis is accepted";

**else** cout << "The hypothesis is rejected";

}

**int** main()

{

**int** n;

cout << "Enter n: ";

cin >> n;

cout << "**\n**";

//Шаг выборки

**int** k;

cout << "Enter k: ";

cin >> k;

cout << "**\n**";

**char** r;

cout << "Choose distribution (r or e): ";

cin >> r;

cout << "**\n**";

vector <**double**> sample;

sample.resize(n);

string sampleName;

cout << "Enter file name for the sample: ";

cin >> sampleName;

cout << endl;

//Поиск файла с выборкой

ifstream **Y**(sampleName);

**if**(!Y)

{

cout << "File not founded.**\n\n**";

**return** **1**;

}

//Заполнение массива выборкой

**for** (**int** i = **0**, j = **0**; i < n; i++)

{

Y >> sample[i];

}

**double** xPSquare = pirsonSquare(sample, k, r);

cout << "Pirson Square is " << xPSquare << "**\n\n**";

pValueTab(xPSquare, k);

**double** pValue = monteKarlo(xPSquare, n, k, r);

cout << "MMK p-value is " << pValue << "**\n**";

pValueMMK(pValue);

cout << endl;

**return** **0**;

}