Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Новосибирский государственный технический университет»

Кафедра теоретической и прикладной информатики

КУРСОВОЙ ПРОЕКТ

по теме «Исследование скорости сходимости распределения статистики критерия однородности Манна-Уитни к предельному закону»

курса «Компьютерные технологии моделирования и анализа данных»

Выполнил:

студент ФПМИ,

гр. ПММ-01

Кастин В. С.

Проверил:

Профессор каф. ТПИ

Постовалов С. Н.

Дата: 24.05.2021

Новосибирск 2021

**Содержание**

[Введение 3](#_Toc72964173)

[Результаты исследований 4](#_Toc72964174)

[Вывод 14](#_Toc72964175)

[Список использованных источников 15](#_Toc72964176)

[Код программы 16](#_Toc72964177)

# 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 статистические критерии. Одним из этих критериев является U-критерий Манна-Уитни.

Ранговый критерий Манна и Уитни основан на критерии Уилкоксона для независимых выборок. Он является непараметрическим аналогом *t*-критерия для сравнения двух средних значений непрерывных распределений.

U-критерий Манна-Уитни - статистический критерий, используемый для оценки различий между двумя независимыми выборками. Ранговый критерий Ранговый критерий Манна и Уитни основан на критерии Уилкоксона для независимых выборок. Он является непараметрическим аналогом *t*-критерия для сравнения двух средних значений непрерывных распределений.

Данный метод выявления различий между выборками был предложен в 1945 году американским химиком и статистиком Фрэнком Уилкоксоном. В 1947 году он был существенно переработан и расширен Х. Б. Манном и Д. Р. Уитни, по именам которых сегодня обычно и называется.

Для вычисления статистики упорядочивают *m* + *n* значений объединенной выборки, определяют сумму рангов , соответствующую элементам первой выборки, и сумму рангов второй . Вычисляются

,

.

Статистика критерия имеет вид: .

Для достаточно больших выборок (), когда объемы выборок не слишком малы () используется статистика

,

распределение которой приближенно описывается стандартным нормальным законом. Вид критической области – двусторонняя.

# Результаты исследований

Целью исследования является определение объёма выборки n\*, начиная с которого расстояние до предельного закона распределения не превышает 0.01. Для этого необходимо аппроксимировать зависимость расстояния до предельного закона распределения функцией *an-b*.

Для исследования будут использованы нормальное, экспоненциальное распределения, а также распределение Коши.

Для определения n\* необходимо построить регрессию по точкам с разными n, найденными с помощью статистики Колмогорова:



Данная задача будет рассмотрена детальнее на примере стандартного нормального распределения.

С помощью разработанной программы по вычислению статистики по критерию Манна-Уитни были получены выборки статистик объёмом N = 1 300 000, с различными n от 6 до 15.

Далее с помощью критерия Колмогорова было найдено расстояние от эмпирического распределения до предельного:

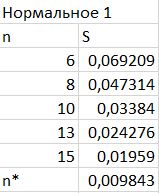


Рисунок 1 - Полученные значения статистики для разных n при стандартном нормальном распределении

По полученным точкам была построена степенная регрессия:

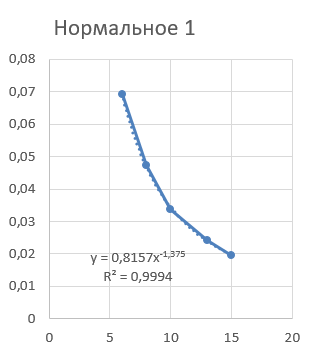


Рисунок 2 - Регрессия для стандартного нормального распределения

Поскольку коэффициент детерминации довольно близок к 1, а точки разбросаны по обе стороны от линии тренда, то можно сделать вывод, что аппроксимация хорошая.

Приравнивая полученное уравнение регрессии к 0.01, было получено n\*, равное 25 (с округлением до целых в большую сторону).

На заключительном этапе необходимо проверить полученный объём выборки n\*, чтобы полученное расстояние до предельного закона не отклонялось от 0.01 не более чем на 0.001 с доверительной вероятностью 0.99.

Для этого была смоделирована выборка статистик из N = 2 649 147 и n = n\*. Ниже приведён график проверки полученного результата:

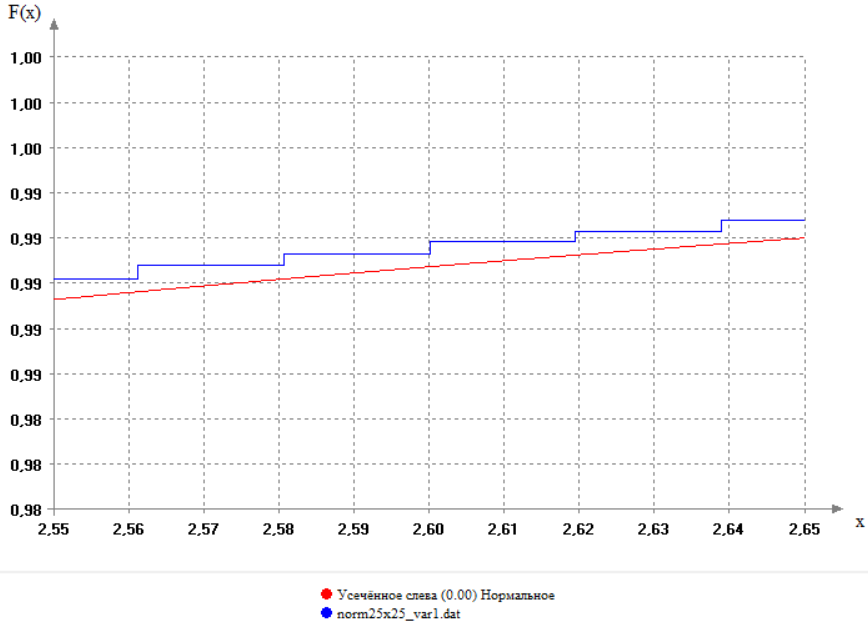


Рисунок 3 - Проверка на согласие критерием Колмогорова при нормальном распределении

Аналогичные исследования были проведены с экспоненциальным распределением и распределением Коши:

Степенная регрессия для распределения Коши:

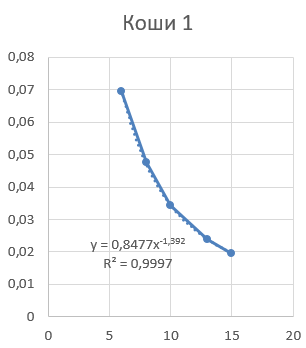


Рисунок 4 - Регрессия для стандартного распределения Коши

Полученное n\* для распределения Коши – 25.

График при проверке на согласие при n\*:

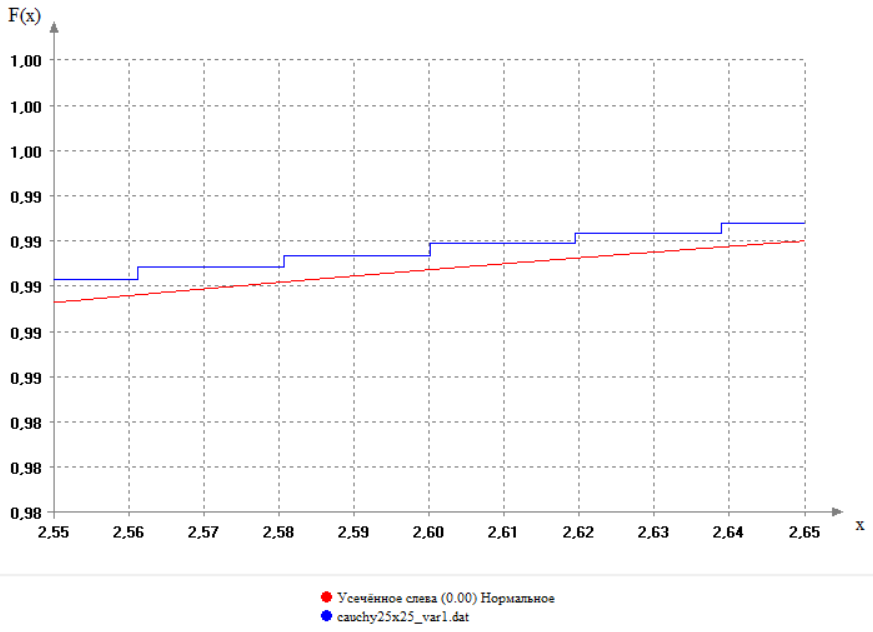


Рисунок 5 - Проверка на согласие критерием Колмогорова при распределении Коши

Степенная регрессия для экспоненциального распределения:

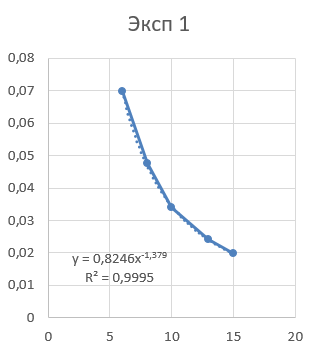


Рисунок 6 - Регрессия для стандартного экспоненциального распределения

Полученное n\* для экспоненциального распределения – 25.

График при проверке на согласие при n\*:

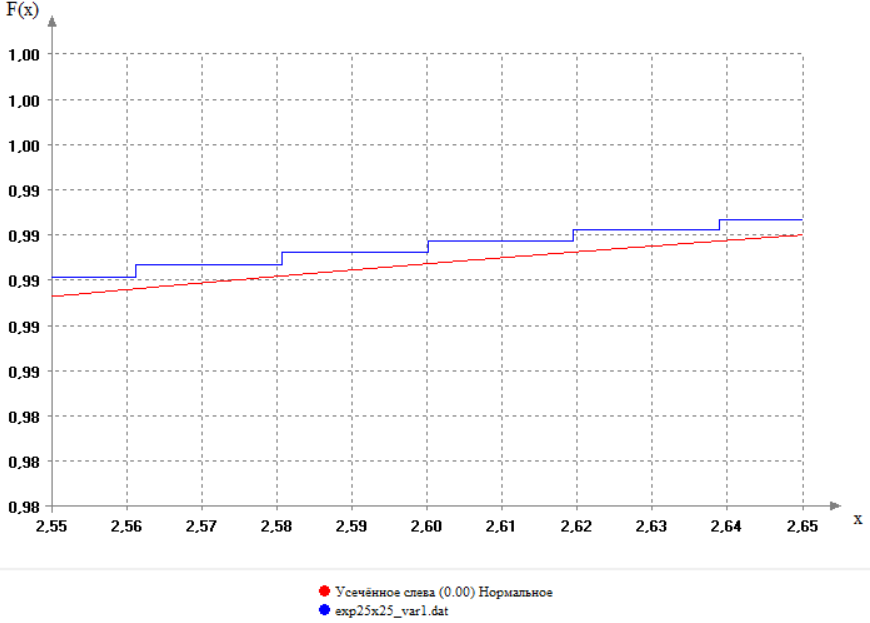


Рисунок 7 - Проверка на согласие критерием Колмогорова при экспоненциальном распределении

Далее приведены степенные регрессии для нормального распределения и распределения Коши, но с разными дисперсиями у выборок, где отличия в 2, 5 и 10 раз, объёмы выборок n от 6 до 15:

Нормальное распределение:

Разница дисперсий в 2 раза:

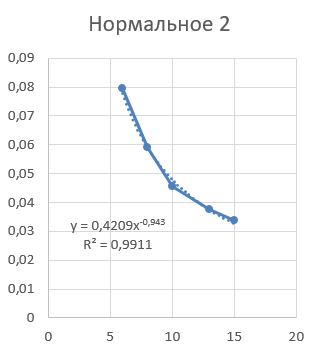


Рисунок 8 - Регрессия для стандартного нормального распределения с разницей дисперсий в 2 раза

Разница дисперсий в 5 раз:

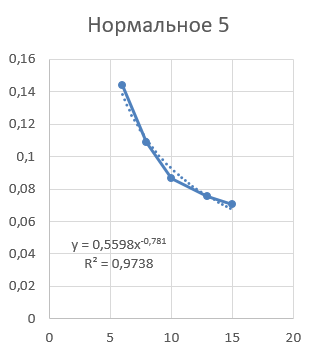


Рисунок 9 - Регрессия для стандартного нормального распределения с разницей дисперсий в 5 раз

Разница дисперсий в 10 раз:

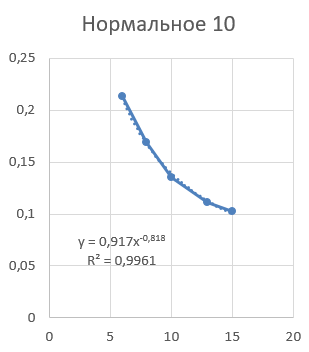


Рисунок 10 - Регрессия для стандартного нормального распределения с разницей дисперсий в 10 раз

Распределение Коши:

Разница дисперсий в 2 раза:

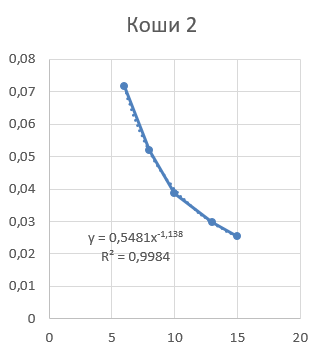


Рисунок 11 - Регрессия для стандартного распределения Коши с разницей дисперсий в 2 раза

Разница дисперсий в 5 раз:

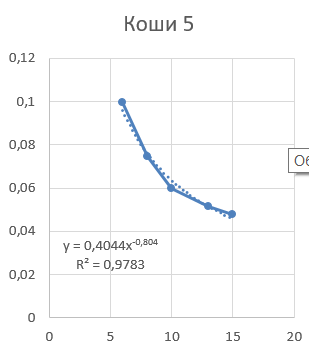


Рисунок 12 - Регрессия для стандартного распределения Коши с разницей дисперсий в 5 раз

Разница дисперсий в 10 раз:

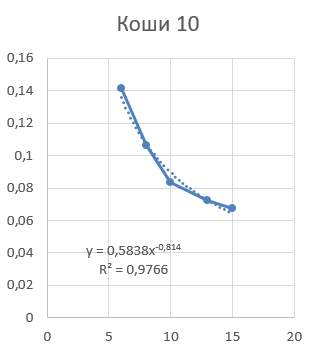


Рисунок 13 - Регрессия для стандартного распределения Коши с разницей дисперсий в 10 раза

Далее приведен скриншот работы программы:

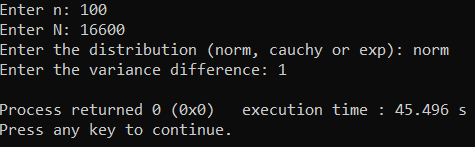


Рисунок 14 – Скриншот работы программы

# Вывод

В ходе работы была изучена методика исследования скорости сходимости распределения статистики критерия к предельному с использованием компьютерных технологий.

Для этого была разработана программа для моделирования выборки статистик критерия Манна-Уитни. Были смоделированы выборки статистик разных значений n, и для каждого значения n было найдено расстояние.

Также Была аппроксимирована зависимость расстояния до предельного закона распределения с помощью степенной регрессии.

Для каждого из трёх стандартных распределений был найден объём n\*, начиная с которого расстояние до предельного закона распределения отклоняется от 0.01 не более чем на 0.001 с доверительной вероятностью 0.99.

В конце, с помощью ISW были проверены полученные результаты, и в итоге с помощью графиков было подтверждено, что при полученных объёмах n\* расстояние до предельного закона распределения отклоняется не более чем на 0.001.

В конце было установлено, что чем сильнее разность дисперсий, тем большее расстояние между эмпирическим и предельным распределением и тем меньше скорость сходимости в случае критерия Манна-Уитни.

# Список использованных источников

1. Статистический анализ данных, моделирование и исследование вероятностных закономерностей. Компьютерный подход : [монография] / Б. Ю. Лемешко [и др.]. - Новосибирск, 2011. - 887 с. : ил., табл.
2. U-критерий Манна — Уитни [Электронный ресурс] // WikipediA. (ru.wikipedia.org/wiki/U-критерий\_Манна\_ —\_Уитни#:~:text=Mann-Whitney%20U-test),значении%20параметра%20между%20малыми%20выборками.) Дата обращения: 24.05.2021.

# Код программы

#include <iostream>

#include <ctime>

#include <random>

#include <vector>

#include <algorithm>

#include <stdlib.h>

#include <math.h>

#include <fstream>

#include <string>

#include <chrono>

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

// Для генерации распределений внутри функций

**static** std::random\_device rd;

**static** std::mt19937 rng{rd()};

**double** **statisticsForNLittle**(vector<**double**> &sample1, vector<**double**> &sample2)

{

// Заполнение общего вектора для дальнейшего нахождения рангов

vector<vector<**double**>> R;

R.resize(sample1.size() + sample2.size());

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

R[i].resize(**2**);

**for**(**int** i = **0**; i < sample1.size(); i++)

R[i][**0**] = sample1[i];

**for**(**int** i = sample1.size(), j = **0**; i < R.size(); i++, j++)

R[i][**0**] = sample2[j];

sort(R.begin(), R.end());

**int** count = **0**;

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

// Расчёт ранга последних элементов при условии, что они равны друг другу

**for**(**int** i = R.size()-**1**; R[i][**0**] == R[i-**1**][**0**]; i--)

count++;

// Если в конце вектора есть элементы, равные друг другу, находится их ранг

**if**(count > **0**)

{

// Находится сумма рангов этих элементов

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

sum += i+**1**;

// Среднее арифмитическое (count + 1, т.к. если одинаковых элемента 2, то count, согласно коду выше, будет равен 1, и из-за этого предпоследний элемент не инициализируется)

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

R[i][**1**] = sum / (count + **1**);

}

// Иначе последний элемент получает самое высокое значение ранга

**else** R[R.size()-**1**][**1**] = R.size();

// Временная переменная для определения индекса элемента, до которого нужно инициализировать оставшиеся элементы

**int** tmp = count;

count = **0**;

sum = **0**;

// Инициализация рангов всех элементов кроме последнего (последних)

**for**(**int** i = **0**; i < R.size() - **1** - tmp; i++)

{

// Если текущий элемент не равен предыдущему (о чём говорит count) и не равен следующему элементам, то присваивается ранг по порядку

**if**(count == **0** && R[i][**0**] != R[i+**1**][**0**])

R[i][**1**] = i+**1**;

// Иначе

**else**

{

// Если текущий элемент равен следующему, увеличивается счётчик

**if**(R[i][**0**] == R[i+**1**][**0**])

count++;

// Иначе...

**else**

{

// Счётчик увеличивается в последний раз для определения всех одинаковых элементов

count++;

**for**(**int** j = **0**; j < count; j++)

sum += i-j;

**for**(**int** j = **0**; j < count; j++)

R[i-j][**1**] = sum / count;

sum = **0**;

count = **0**;

}

}

}

vector<**double**> vTmp;

vTmp.resize(R.size());

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

{

vTmp[i] = R[i][**0**];

}

// Ранг 1-й выборки

**int** R1 = **0**;

// Ранг 2-й выборки

**int** R2 = **0**;

// Переменные для нахождения индекса элемента по значению

vector<**double**>::iterator it;

**int** index = **0**;

// Расчёт 1-го ранга

**for**(**int** i = **0**; i < sample1.size(); i++)

{

// Поиск элемента по значению

it = find(vTmp.begin(), vTmp.end(), sample1[i]);

// Нахождение индекса элемента

index = distance(vTmp.begin(), it);

// Если элемент есть в общем векторе, ранг увеличивается

**if**(index < vTmp.size())

R1 += R[index][**1**];

}

//cout << "R1 = " << R1 << endl;

// Расчёт 2-го ранга

**for**(**int** i = **0**; i < sample2.size(); i++)

{

it = find(vTmp.begin(), vTmp.end(), sample2[i]);

index = distance(vTmp.begin(), it);

**if**(index < vTmp.size())

R2 += R[index][**1**];

}

//cout << R1 + R2 << endl;

//cout << "R2 = " << R2 << endl;

**int** m = sample1.size();

**int** n = sample2.size();

**double** U1 = m \* n + ((m \* (m - **1**)) / **2**) - R1;

//cout << "U1 = " << U1 << endl;

**double** U2 = m \* n + ((n \* (n - **1**)) / **2**) - R2;

//cout << "U2 = " << U2 << endl;

**return** min(U1, U2);

}

**double** **MannWhitneyUtest**(vector<**double**> &sample1, vector<**double**> &sample2)

{

**int** m = sample1.size();

**int** n = sample2.size();

**if**(m + n > **60** && m >= **8** && n >= **8**)

{

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

**for**(**int** i = **0**; i < sample1.size(); i++)

{

**for**(**int** j = **0**; j < sample2.size(); j++)

{

**if**(sample2[j] < sample1[i])

sum += **1**;

**else** **if**(sample2[j] == sample1[i])

sum += **0.5**;

}

}

**return** ((abs(sum - ((m \* n) / **2**))) / sqrt((m \* n \* (m + n + **1**)) / **12**));

}

**else** **return** statisticsForNLittle(sample1, sample2);

}

vector<**double**> normRGenerate(**int** n, **double** D = **1**)

{

normal\_distribution<**double**> normR(**0**, D);

vector<**double**> sample;

sample.resize(n);

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

sample[i] = normR(rng);

**return** sample;

}

vector<**double**> cauchyRGenerate(**int** n, **double** D = **1**)

{

cauchy\_distribution<**double**> cauchyR(**0**, D);

vector<**double**> sample;

sample.resize(n);

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

sample[i] = cauchyR(rng);

**return** sample;

}

vector<**double**> expRGenerate(**int** n, **double** D = **1**)

{

exponential\_distribution<**double**> expR(D);

vector<**double**> sample;

sample.resize(n);

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

sample[i] = expR(rng);

**return** sample;

}

vector<**double**> MonteKarloMethod(**int** N, **int** m, **int** n, string typeOfRaspr = "norm", **int** varDif = **1**)

{

vector<**double**> statSample;

vector<**double**> sample1;

vector<**double**> sample2;

**if**(typeOfRaspr == "norm")

{

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

{

sample1 = normRGenerate(m);

sample2 = normRGenerate(n, varDif);

statSample.push\_back(MannWhitneyUtest(sample1, sample2));

}

}

**else** **if**(typeOfRaspr == "cauchy")

{

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

{

sample1 = cauchyRGenerate(m);

sample2 = cauchyRGenerate(n, varDif);

statSample.push\_back(MannWhitneyUtest(sample1, sample2));

}

}

**else**

{

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

{

sample1 = expRGenerate(m);

sample2 = expRGenerate(n, varDif);

statSample.push\_back(MannWhitneyUtest(sample1, sample2));

}

}

**return** statSample;

}

**void** samplesGenerate(**int** N, **int** n, **int** m, string typeOfRaspr = "norm", **int** varDif = **1**)

{

vector<**double**> sample;

sample = MonteKarloMethod(N, m, n, typeOfRaspr, varDif);

//sort(sample.begin(), sample.end());

string sampleName = typeOfRaspr + to\_string(n) + "x" + to\_string(m) + "\_var" + to\_string(varDif) + ".dat";

ofstream **sampleStream**(sampleName);

sampleStream << sampleName << endl;

sampleStream << "0 " << N << endl;

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

{

sampleStream << sample[i] << endl;

}

}

**int** main()

{

**int** n, m, N;

//n = 130;

//m = 130;

//N = 2000;

cout << "Enter n: ";

cin >> n;

m = n;

cout << "Enter N: ";

cin >> N;

string dist;

cout << "Enter the distribution (norm, cauchy or exp): ";

cin >> dist;

**int** varDif;

cout << "Enter the variance difference: ";

cin >> varDif;

samplesGenerate(N, n, m, dist, varDif);

**return** **0**;

}