Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу

Кафедра математичних

методів в інженерії

Лабораторна робота №6

Тема: “ Програмна реалізаціякритерію узгодженості Пірсона”

Виконав ст. гр. ПІ-14-1

Гавриляк Д. Р., Демчук Л. В.

Перевірив:

Незамай Б. С.

м. Івано-Франківськ

2016

**Технічне забезпечення**: ПЕОМ, середовище програмування.

**Завдання:**

**Варіант №8**

Організувати ввід з консолі  цілих чисел в діапазоні від 0 до 20 включно. Запрограмувати перевірку на допустимі значення. Запрограмувати перевірку апроксимації Голдштейна для різних значень  (контрольні значення – додаток А).

Ввести числа вручну випадковим чином. Організувати перевірку програмним чином на основі критерію  на відповідність введеної вибірки нормальному та рівномірному розподілу за двома різними рівнями значущості.

Вивести на екран вибірку та результати перевірок.

**Виконання:**

**Код програми:**

1. Клас «Intervals» для опису інтервалів поділу простору нормального розподілу:
2. using System;
3. using System.Collections.Generic;
4. using System.Linq;
5. using System.Text;
6. using System.Threading.Tasks;
7. namespace lab6
8. {
9. public class Intervals
10. {
11. public double start;
12. public double end;
13. public int mi = 0;
14. }
15. }
16. Клас «Criteria» для обрахунку критерія Пірса та порівняння його з критичним:

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab6

{

public enum Distribution { Normal, Uniform }

public class Criteria

{

public List<Intervals> list = new List<Intervals>(); // інтервали для норм. розподілу

public Dictionary<double, int> unique = new Dictionary<double, int>(); // унікальні значення та їхні частоти для рівном. розподілу

public Distribution type = new Distribution(); // тип розподілу

public double dispertion; // дисперсія

public double average; // середнє значення

public double[] array; // вибірка

public Dictionary<double, double> Laplas = new Dictionary<double, double>();

public string filename;

public int r; // кількість інтервалів для норм. розподілу

public double[] z1; // змінна для функції Лапласа по лівій межі інтервалу

public double[] z2; // змінна для функції Лапласа по правій межі інтервалу

public double[] Laplace\_z1; /\*= new double[5] { -0.5, -0.3264, -0.0753, 0.2123, 0.4049 };\*/ // лаплас для лівих меж

public double[] Laplace\_z2; /\* = new double[5] { -0.3264, -0.0753, 0.2123, 0.4049, 0.5 };\*/ // лаплас для правих меж

public double[] p; // теоретична ймовірність для інтервалу

public double[] ni; // теоретичні частоти для інтервалів

public double criteriaValue = 0; // емпіричний критерій Пірсона

public double criticalNormal; // критичний критерій Пірсона для норм. розп., альфа = 0,01 (0,99 в таблиці)

public double criticalUniform; // критичний критерій Пірсона для рівном. розп., альфа = 0,1 (0,9 в таблиці)

// Конструктор для норм. розподілу

public Criteria(double[] array, Distribution type, int r, string filename)

{

Array.Sort(array);

this.array = new double[array.Length];

for (int i = 0; i < array.Length; i++)

{

this.array[i] = array[i];

}

this.r = r;

this.dispertion = Dispertion(this.array);

this.average = this.array.Average();

this.type = type;

this.filename = filename;

}

// Конструктор для рівномірного розподілу

public Criteria(double[] array, Distribution type)

{

Array.Sort(array);

this.array = new double[array.Length];

for (int i = 0; i < array.Length; i++)

{

this.array[i] = array[i];

}

this.type = type;

}

// Перевірка гіпотези про нормальний розподіл

public void Verify()

{

if (this.type == Distribution.Normal)

{

LoadFromFile(filename);

CreateIntervals();

Evaluate\_Mi();

CorrectIntervals();

Evaluate\_Mi();

Evaluate\_z1\_2();

Evaluate\_Laplace();

Evaluate\_p();

Evaluate\_ni();

Evaluate\_Criteria();

ShowStupin();

Console.WriteLine("Введiть критичний критерiй для обрахованої ступенi свободи: ");

setCritical(Convert.ToDouble(Console.ReadLine()));

Console.WriteLine("Критерiй емпiричний: " + criteriaValue + ", критерiй критичний: " + criticalNormal);

if (criteriaValue > criticalNormal)

Console.WriteLine("Розподiл далекий вiд нормального");

else

Console.WriteLine("Розподiл близький до нормального");

}

if (this.type == Distribution.Uniform)

{

UniqueValues();

Evaluate\_p();

Evaluate\_ni();

Evaluate\_Criteria();

ShowStupin();

Console.WriteLine("Введiть критичний критерiй для обрахованої ступенi свободи: ");

setCritical(Convert.ToDouble(Console.ReadLine()));

Console.WriteLine("Критерiй емпiричний: " + criteriaValue + ", критерiй критичний: " + criticalUniform );

if (criteriaValue > criticalUniform)

Console.WriteLine("Розподiл далекий вiд рiвномiрного");

else

Console.WriteLine("Розподiл близький до рiвномiрного");

}

}

// Визначення інтервалів

public void CreateIntervals()

{

double minValue = array.Min();

double maxValue = array.Max();

double interval = (maxValue - minValue) / (double)r; // величина інтервалу

for (double i = minValue; i < maxValue; i += interval)

{

if (i == maxValue) break;

Intervals obj = new Intervals();

obj.start = i;

obj.end = i + interval;

list.Add(obj);

}

}

// Коригування інтервалів

public void CorrectIntervals()

{

int permission = 0;

do

{

Evaluate\_Mi();

for (int i = 0; i < list.Count; i++)

{

if (list.ElementAt(i).mi < 5)

{

if (i == list.Count - 1)

{

list.ElementAt(i - 1).end = list.ElementAt(i).end;

list.RemoveAt(i);

}

else

{

list.ElementAt(i+1).start = list.ElementAt(i).start;

list.RemoveAt(i);

}

permission = 0;

break;

}

else

permission = 1;

}

} while (permission == 0);

}

// Обчислення частот вибіркових елементів

public void Evaluate\_Mi()

{

for(int i =0; i < list.Count; i++)

{

list.ElementAt(i).mi = 0;

}

for (int i = 0; i < list.Count; i++)

{

for (int j = 0; j < array.Length; j++)

{

if (list.ElementAt(i).start <= array[j] && array[j] <= list.ElementAt(i).end)

list.ElementAt(i).mi++;

}

}

}

// Обчислення дисперсії

public double Dispertion(double[] array)

{

double average = array.Average();

double sum = 0.0;

for (int i = 0; i < array.Length; i++)

sum += Math.Pow((array[i] - average), 2);

return sum / array.Length;

}

// Обчислення вхідних значень для ф-цій Лапласа

public void Evaluate\_z1\_2()

{

z1 = new double[list.Count];

z2 = new double[list.Count];

for (int i = 0; i < list.Count; i++)

{

z1[i] = Math.Round((list.ElementAt(i).start - average) / Math.Sqrt(dispertion), 2);

z2[i] = Math.Round((list.ElementAt(i).end - average) / Math.Sqrt(dispertion), 2);

}

}

// Обчислення значень функції Лапласа

public void Evaluate\_Laplace()

{

Laplace\_z1 = new double[z1.Length];

Laplace\_z2 = new double[z2.Length];

for(int i = 0; i < list.Count; i++)

{

if(z1[i] < 0)

Laplace\_z1[i] = -1\*Laplas[-z1[i]];

else

Laplace\_z1[i] = Laplas[z1[i]];

if (z2[i] < 0)

Laplace\_z2[i] = -1 \* Laplas[-z2[i]];

else

Laplace\_z2[i] = Laplas[z2[i]];

}

Laplace\_z1[0] = -0.5;

Laplace\_z2[z2.Length-1] = 0.5;

}

// Обчилення теоретичних ймовірностей для норм. розподілу

public void Evaluate\_p()

{

if (this.type == Distribution.Normal)

{

p = new double[list.Count];

for (int i = 0; i < p.Length; i++)

{

p[i] = Laplace\_z2[i] - Laplace\_z1[i];

}

}

if (this.type == Distribution.Uniform)

{

p = new double[unique.Count];

for (int i = 0; i < p.Length; i++)

{

p[i] = 1.0 / unique.Count;

}

}

}

// Обчислення теоретичних частот

public void Evaluate\_ni()

{

ni = new double[p.Length];

for (int i = 0; i < p.Length; i++)

{

ni[i] = array.Length \* p[i];

}

}

// Обчислення критерію Пірсона

public void Evaluate\_Criteria()

{

if (this.type == Distribution.Normal)

{

for (int i = 0; i < list.Count; i++)

{

criteriaValue += Math.Pow((list.ElementAt(i).mi - ni[i]), 2) / ni[i];

}

}

if(this.type == Distribution.Uniform)

{

for(int i = 0; i < unique.Count; i++)

{

criteriaValue += Math.Pow(unique.ElementAt(i).Value - ni[i], 2) / ni[i];

}

}

}

// Пошук унікальних значень для рівном. розподілу та їхні частоти

public void UniqueValues()

{

int step = 0;

for (int i = 0; i < array.Length; i += step)

{

step = 0;

for (int j = 0; j < array.Length; j++)

{

if (array[i] == array[j])

step++;

}

unique.Add(array[i], step);

}

}

public void LoadFromFile(string filename)

{

StreamReader reader = File.OpenText(filename);

double x;

double y;

while (!reader.EndOfStream)

{

string[] line = reader.ReadLine().Split('\t');

x = Convert.ToDouble(line[0]);

y = Convert.ToDouble(line[1]);

Laplas.Add(x, y);

}

}

public void setCritical(double k)

{

if(this.type == Distribution.Normal)

this.criticalNormal = k;

if (this.type == Distribution.Uniform)

this.criticalUniform = k;

}

public void ShowStupin()

{

Console.WriteLine("Обрахований ступiнь свободи: ");

if (this.type == Distribution.Normal)

Console.WriteLine(list.Count - 2 - 1);

if (this.type == Distribution.Uniform)

Console.WriteLine(unique.Count - 0 - 1);

}

}

}

1. Клас «Program» для початку роботи програми:
2. using System;
3. using System.Collections.Generic;
4. using System.Linq;
5. using System.Text;
6. using System.Threading.Tasks;
7. namespace lab6
8. {
9. class Program
10. {
11. static void Main(string[] args)
12. {
14. double[] array2 = new double[28];
15. for (int i = 0; i < array2.Length; i++)
16. {
17. bool result;
18. do
19. {
20. Console.WriteLine("Enter the integer value: ");
21. string value = Console.ReadLine();
22. int number;
23. result = Int32.TryParse(value, out number);
24. if (result)
25. {
26. array2[i] = Convert.ToDouble(number);
27. }
28. else
29. {
30. Console.WriteLine("Not integer! Retry please.");
31. }
32. } while (!result);
33. }
34. Console.WriteLine("-----");
35. Console.WriteLine("Створена вибiрка:");
36. Console.WriteLine();
37. for (int i = 0; i < array2.Length; i++)
38. {
39. Console.Write(array2[i] + " ");
40. }
41. Console.WriteLine();
42. Distribution type = Distribution.Normal;
43. int r = 5;// кількість інтервалів для норм. розподілу
44. Criteria cr = new Criteria(array2, type, r, "Laplas.txt");
45. cr.Verify();
46. Console.WriteLine("------");
47. type = Distribution.Uniform;
48. Criteria cr2 = new Criteria(array2, type);
49. cr2.Verify();
50. Console.ReadLine();
51. }
52. }

}

**Результати виконання**

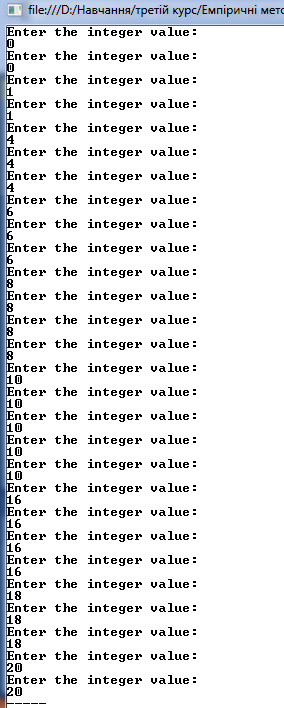


Рис. 1 – Введення даних для вибірки

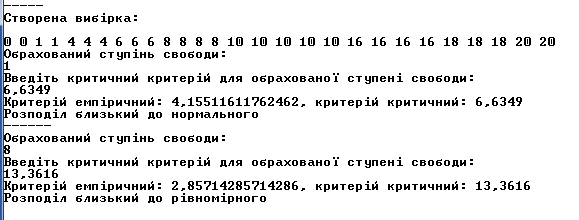


Рис. 2 – Результати перевірок

**Висновки:** виконуючи дану лабораторну роботу, ми навчились програмно реалізовуватикритерію узгодженості Пірсона, який дозволяє спростити або підтвердити гіпотезу про те, що емпіричний розподіл подібний до рівномірного, нормального та інших.