Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Методи оптимізації та планування експерименту

Лабораторна робота №3

“ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ

ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ”

Виконав:

студент групи ІВ-83

Поліщук Д.Я.

Перевірив:

ас. Регіда П.Г.

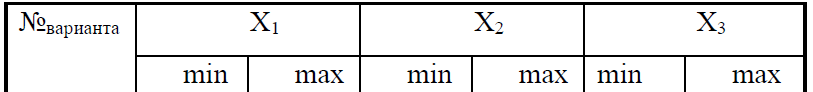
Київ

2020 р.

**Мета:** провести дробовий трьохфакторний експеримент. Скласти матрицю планування, знайти коефіцієнти рівняння регресії, провести 3 статистичні перевірки.

**Номер у списку: 18.**

Варіант завдання: 318.





1. **Лістинг програми:**

/\*\*

\* @author Студент групи ІВ-83 Поліщук Даніїл.

\* Номер у списку групи: 18.

\* Варіант завдання: 318.

\*/

import org.apache.commons.math3.linear.Array2DRowRealMatrix;

import org.apache.commons.math3.linear.LUDecomposition;

import java.security.SecureRandom;

import java.util.ArrayList;

import java.util.Collections;

public class MOPE\_lab3 {

private final static double GTR = 0.7679;

private final static double FTR = 4.1;

private final static double TTR = 2.306;

public static void main(String[] args) {

int X1min = 20;

int X1max = 70;

int X2min = -15;

int X2max = 45;

int X3min = 20;

int X3max = 35;

double X\_average\_max = (X1max + X2max + X3max) / 3;

double X\_average\_min = (X1min + X2min + X3min) / 3;

double Ymin = 200 + X\_average\_min;

double Ymax = 200 + X\_average\_max;

double difference = Ymax - Ymin;

int[] x1\_code = {-1, -1, 1, 1};

int[] x2\_code = {-1, 1, -1, 1};

int[] x3\_code = {-1, 1, 1, -1};

int[] x1 = {20, 20, 70, 70};

int[] x2 = {-15, 45, -15, 45};

int[] x3 = {20, 35, 35, 20};

int[] y1 = new int[4];

int[] y2 = new int[4];

int[] y3 = new int[4];

SecureRandom random = new SecureRandom();

System.out.println("Матриця Y:");

for (int i = 0; i < 4; i++) {

y1[i] = (int) Ymin + random.nextInt((int) (difference + 1));

System.out.print(y1[i] + " ");

y2[i] = (int) Ymin + random.nextInt((int) (difference + 1));

System.out.print(y2[i] + " ");

y3[i] = (int) Ymin + random.nextInt((int) (difference + 1));

System.out.println(y3[i] + " ");

} System.out.println();

double Y1\_average = (y1[0] + y2[0] + y3[0]) / 3;

double Y2\_average = (y1[1] + y2[1] + y3[1]) / 3;

double Y3\_average = (y1[2] + y2[2] + y3[2]) / 3;

double Y4\_average = (y1[3] + y2[3] + y3[3]) / 3;

int mx1 = (x1[0] + x1[1] + x1[2] + x1[3]) / 4;

int mx2 = (x2[0] + x2[1] + x2[2] + x2[3]) / 4;

int mx3 = (x3[0] + x3[1] + x3[2] + x3[3]) / 4;

double my = (Y1\_average + Y2\_average + Y3\_average + Y4\_average) / 4;

double a11 = (x1[0] \* x1[0] + x1[1] \* x1[1] + x1[2] \* x1[2] + x1[3] \* x1[3]) / 4;

double a22 = (x2[0] \* x2[0] + x2[1] \* x2[1] + x2[2] \* x2[2] + x2[3] \* x2[3]) / 4;

double a33 = (x3[0] \* x3[0] + x3[1] \* x3[1] + x3[2] \* x3[2] + x3[3] \* x3[3]) / 4;

double a1 = (x1[0] \* Y1\_average + x1[1] \* Y2\_average + x1[2] \* Y3\_average + x1[3] \* Y4\_average) / 4;

double a2 = (x2[0] \* Y1\_average + x2[1] \* Y2\_average + x2[2] \* Y3\_average + x2[3] \* Y4\_average) / 4;

double a3 = (x3[0] \* Y1\_average + x3[1] \* Y2\_average + x3[2] \* Y3\_average + x3[3] \* Y4\_average) / 4;

double a12 = (x1[0] \* x2[0] + x1[1] \* x2[1] + x1[2] \* x2[2] + x1[3] \* x2[3]) / 4; //a12=a21

double a13 = (x1[0] \* x3[0] + x1[1] \* x3[1] + x1[2] \* x3[2] + x1[3] \* x3[3]) / 4; //a13=a31

double a23 = (x2[0] \* x3[0] + x2[1] \* x3[1] + x2[2] \* x3[2] + x2[3] \* x3[3]) / 4; //a23=a32

double[][] v0 = {{1, mx1, mx2, mx3},

{mx1, a11, a12, a13},

{mx2, a12, a22, a23},

{mx3, a13, a23, a33}};

double[][] v1 = {{my, mx1, mx2, mx3},

{a1, a11, a12, a13},

{a2, a12, a22, a23},

{a3, a13, a23, a33}};

double[][] v2 = {{1, my, mx2, mx3},

{mx1, a1, a12, a13},

{mx2, a2, a22, a23},

{mx3, a3, a23, a33}};

double[][] v3 = {{1, mx1, my, mx3},

{mx1, a11, a1, a13},

{mx2, a12, a2, a23},

{mx3, a13, a3, a33}};

double[][] v4 = {{1, mx1, mx2, my},

{mx1, a11, a12, a1},

{mx2, a12, a22, a2},

{mx3, a13, a23, a3}};

LUDecomposition A = new LUDecomposition(new Array2DRowRealMatrix(v0));

LUDecomposition B = new LUDecomposition(new Array2DRowRealMatrix(v1));

LUDecomposition C = new LUDecomposition(new Array2DRowRealMatrix(v2));

LUDecomposition E = new LUDecomposition(new Array2DRowRealMatrix(v3));

LUDecomposition D = new LUDecomposition(new Array2DRowRealMatrix(v4));

double b0 = B.getDeterminant() / A.getDeterminant();

double b1 = C.getDeterminant() / A.getDeterminant();

double b2 = E.getDeterminant() / A.getDeterminant();

double b3 = D.getDeterminant() / A.getDeterminant();

double[] array = new double[]{b0,b1,b2,b3};

System.out.println("Наші коефіцієнти:");

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

System.out.print("b"+i+": ");

System.out.printf("%.3f",array[i]);

System.out.println();

} System.out.println();

System.out.println("Наші рівняння регресії:\ny = b0 + b1\*x1 + b2\*x2 + b3\*x3");

double y111 = b0 + b1 \* x1[0] + b2 \* x2[0] + b3 \* x3[0];

double y112 = b0 + b1 \* x1[1] + b2 \* x2[1] + b3 \* x3[1];

double y113 = b0 + b1 \* x1[2] + b2 \* x2[2] + b3 \* x3[2];

double y114 = b0 + b1 \* x1[3] + b2 \* x2[3] + b3 \* x3[3];

double[] array1 = new double[]{y111,y112,y113,y114};

for(int i = 0; i < array1.length; i++){

System.out.println("y = "+Math.round(b0)+" + "+b1+"\*"+x1[i]+" + "+b2+"\*"+x2[i]+" + "+b3+"\*"+x3[i]+" = " + array1[i]);

} System.out.println();

/\*\*

\* Критерій Кохрена

\*/

double sigmaY1 = countDispersia(y1, y2, y3, Y1\_average, 0);

double sigmaY2 = countDispersia(y1, y2, y3, Y2\_average, 1);

double sigmaY3 = countDispersia(y1, y2, y3, Y3\_average, 2);

double sigmaY4 = countDispersia(y1, y2, y3, Y4\_average, 3);

double sigma = (sigmaY1 + sigmaY2 + sigmaY3 + sigmaY4);

ArrayList<Double> collection = new ArrayList<Double>();

collection.add(sigmaY1);

collection.add(sigmaY2);

collection.add(sigmaY3);

collection.add(sigmaY4);

double Gkr = Collections.max(collection) / sigma;

int N = 4, m = 3;

int f1 = m - 1;

int f2 = N;

//q = 0.05

if (cr\_cohr(Gkr)) {

System.out.println("За критерієм Кохрена дисперсія однорідна\n");

} else {

System.out.println("За критерієм Кохрена дисперсія неоднорідна\n");

}

/\*\*

\* Критерій Стьюдента

\*/

double sigmaV = sigma / N;

double sigmaBB = (sigmaV \* sigmaV) / (N \* m);

double sigmaB = Math.sqrt(sigmaBB);

double B0 = (Y1\_average + Y2\_average + Y3\_average + Y4\_average) / 4;

double B1 = (-Y1\_average - Y2\_average + Y3\_average + Y4\_average) / 4;

double B2 = (-Y1\_average + Y2\_average - Y3\_average + Y4\_average) / 4;

double B3 = (-Y1\_average + Y2\_average + Y3\_average - Y4\_average) / 4;

double t0 = Math.abs(B0) / sigmaB;

double t1 = Math.abs(B1) / sigmaB;

double t2 = Math.abs(B2) / sigmaB;

double t3 = Math.abs(B3) / sigmaB;

int f3 = f1 \* f2;

double b00 = 0, b11 = 0, b22 = 0, b33 = 0;

int d = 0;

if (t0 < TTR) {

System.out.println("Коефіцент b0 = "+b0+" не значимий за критерієм Стьюдента");

} else {

b00 = t0;

d++;

}

if (t1 < TTR) {

System.out.println("Коефіцент b1 = "+b1+" не значимий за критерієм Стьюдента");

} else {

b11 = t1;

d++;

}

if (t2 < TTR) {

System.out.println("Коефіцент b2 = "+b2+" не значимий за критерієм Стьюдента");

} else {

b22 = t2;

d++;

}

if (t3 < TTR) {

System.out.println("Коефіцент b3 = "+b3+" не значимий за критерієм Стьюдента");

} else {

b33 = t3;

d++;

}

System.out.println();

double y121 = b00 + b11 \* x1[0] + b22 \* x2[0] + b33 \* x3[0];

double y122 = b00 + b11 \* x1[1] + b22 \* x2[1] + b33 \* x3[1];

double y123 = b00 + b11 \* x1[2] + b22 \* x2[2] + b33 \* x3[2];

double y124 = b00 + b11 \* x1[3] + b22 \* x2[3] + b33 \* x3[3];

/\*\*

\* Критерій Фішера

\*/

int f4 = N - d;

double sigmaAd = ((y121 - Y1\_average) \* (y121 - Y1\_average) + (y122 - Y2\_average) \* (y122 - Y2\_average)

+ (y123 - Y3\_average) \* (y123 - Y3\_average) + (y124 - Y4\_average) \* (y124 - Y4\_average)) / (m / f4);

double Fkr = sigmaAd / sigmaBB;

if (cr\_fisher(Fkr)) {

System.out.println("За критерієм Фішера рівняння неадекватно оригіналу");

} else {

System.out.println("За критерієм Фішера рівняння адекватно оригіналу");

}

}

private static boolean cr\_cohr(double a) {

return a < GTR;

}

private static boolean cr\_fisher(double a) {

return a > FTR;

}

private static double countDispersia(int[] a, int[] b, int[] c, double y, int i) {

return ((a[i] - y) \* (a[i] - y) + (b[i] - y) \* (b[i] - y) + (c[i] - y) \* (c[i] - y)) / 3;

}

}