Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**ЛАБОРАТОРНА РОБОТА № 2**

з дисципліни «Методи оптимізації та планування експерименту»  
 на тему:

***«****ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ*

*ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ****»***

ВИКОНАЛА:

студентка ІІ курсу ФІОТ

групи ІО-82

Трандашир Юлія

ПЕРЕВІРИВ:

Регіда П. Г.

Київ – 2020

import random, math

import numpy as np

x1\_min = 10

x1\_max = 40

x2\_min = 10

x2\_max = 60

x3\_min = 10

x3\_max = 15

y\_min = 230

y\_max = 238

# class Values:

# def \_\_init\_\_(self):

# self.N = 4

# self.D=[]

X = [[-1.0, -1.0, -1.0],

[-1.0, 1.0, 1.0],

[1.0, -1.0, 1.0],

[1.0, 1.0, -1.0]]

Gt = {1: 0.9065, 2: 0.7679, 3: 0.6841, 4: 0.6287, 5: 0.5892, 6: 0.5598, 7: 0.5365, 8: 0.5175, 9: 0.5017,

10: 0.4884, range(11, 17): 0.4366, range(17, 37): 0.3720, range(37, 145): 0.3093}

t\_tabl = {1: 12.71, 2: 4.303, 3: 3.182, 4: 2.776, 5: 2.571,

6: 2.447, 7: 2.365, 8: 2.306, 9: 2.262, 10: 2.228,

11: 2.201, 12: 2.179, 13: 2.160, 14: 2.145, 15: 2.131,

16: 2.120, 17: 2.110, 18: 2.101, 19: 2.093, 20: 2.086,

21: 2.080, 22: 2.074, 23: 2.069, 24: 2.064, 25: 2.060,

26: 2.056, 27: 2.052, 28: 2.048, 29: 2.045, 30: 2.042}

Ft = {8: {1: 5.3, 2: 4.5, 3: 4.1, 4: 3.8}, 9: {1: 5.1, 2: 4.3, 3: 3.9, 4: 3.6},

10: {1: 5.0, 2: 4.1, 3: 3.7, 4: 3.5}, 11: {1: 4.8, 2: 4.0, 3: 3.6, 4: 3.4},

12: {1: 4.8, 2: 3.9, 3: 3.5, 4: 3.3}, 13: {1: 4.7, 2: 3.8, 3: 3.4, 4: 3.2},

14: {1: 4.6, 2: 3.7, 3: 3.3, 4: 3.1}, 15: {1: 4.4, 2: 3.7, 3: 3.3, 4: 3.1},

16: {1: 4.5, 2: 3.6, 3: 3.2, 4: 3.0}, 17: {1: 4.5, 2: 3.6, 3: 3.2, 4: 3.0},

18: {1: 4.4, 2: 3.6, 3: 3.2, 4: 2.9}, 19: {1: 4.4, 2: 3.5, 3: 3.1, 4: 2.9},

range(20, 22): {1: 4.4, 2: 3.5, 3: 3.1, 4: 2.9}, range(22, 24): {1: 4.3, 2: 3.4, 3: 3.1, 4: 2.8},

range(24, 26): {1: 4.3, 2: 3.4, 3: 3.0, 4: 2.8}, range(26, 28): {1: 4.2, 2: 3.4, 3: 3.0, 4: 2.7},

range(28, 30): {1: 4.2, 2: 3.3, 3: 3.0, 4: 2.7}, range(30, 40): {1: 4.2, 2: 3.3, 3: 2.9, 4: 2.7},

range(40, 60): {1: 4.1, 2: 3.2, 3: 2.9, 4: 2.6}, range(60, 120): {1: 4.0, 2: 3.2, 3: 2.8, 4: 2.5}

}

def func(num):

# матриця У

N = 4

m = num

Y = [[random.randint(y\_min, y\_max) for y in range(m)] for x in range(N)]

print("Y: ")

for i in range(len(Y)):

print(Y[i])

print("-------------------------")

# матриця Х

for i in range(len(X)):

if X[i][0] == -1:

X[i][0] = x1\_min

else:

X[i][0] = x1\_max

if X[i][1] == -1:

X[i][1] = x2\_min

else:

X[i][1] = x2\_max

if X[i][2] == -1:

X[i][2] = x3\_min

else:

X[i][2] = x3\_max

print("X: ")

for i in range(len(X)):

print(X[i])

print("-------------------------")

# середнє по рядках

Ys = []

for i in range(len(Y)):

Ys.append(sum(Y[i]) / m)

print("Ys: ", Ys)

print("-------------------------")

mx1 = (X[0][0] + X[1][0] + X[2][0] + X[3][0]) / N

mx2 = (X[0][1] + X[1][1] + X[2][1] + X[3][1]) / N

mx3 = (X[0][2] + X[1][2] + X[2][2] + X[3][2]) / N

my = sum(Ys) / N

print("mx1= {0} \n"

"mx2= {1} \n"

"mx3= {2} \n"

"my= {3} ".format(mx1, mx2, mx3, my))

print("-------------------------")

a1 = ((X[0][0] \* Ys[0]) + (X[1][0] \* Ys[1]) + (X[2][0] \* Ys[2]) + (X[3][0] \* Ys[3])) / N

a2 = ((X[0][1] \* Ys[0]) + (X[1][1] \* Ys[1]) + (X[2][1] \* Ys[2]) + (X[3][1] \* Ys[3])) / N

a3 = ((X[0][2] \* Ys[0]) + (X[1][2] \* Ys[1]) + (X[2][2] \* Ys[2]) + (X[3][2] \* Ys[3])) / N

print("a1= {0} \n"

"a2= {1} \n"

"a3= {2} ".format(a1, a2, a3))

print("-------------------------")

a11 = (X[0][0] \*\* 2 + X[1][0] \*\* 2 + X[2][0] \*\* 2 + X[3][0] \*\* 2) / N

a22 = (X[0][1] \*\* 2 + X[1][1] \*\* 2 + X[2][1] \*\* 2 + X[3][1] \*\* 2) / N

a33 = (X[0][2] \*\* 2 + X[1][2] \*\* 2 + X[2][2] \*\* 2 + X[3][2] \*\* 2) / N

print("a11= {0} \n"

"a22= {1} \n"

"a33= {2} ".format(a11, a22, a33))

print("-------------------------")

a12 = a21 = ((X[0][0] \* X[0][1]) + (X[1][0] \* X[1][1]) + (X[2][0] \* X[2][1]) + (X[3][0] \* X[3][1])) / N

a13 = a31 = ((X[0][0] \* X[0][2]) + (X[1][0] \* X[1][2]) + (X[2][0] \* X[2][2]) + (X[3][0] \* X[3][2])) / N

a23 = a32 = ((X[0][1] \* X[0][2]) + (X[1][1] \* X[1][2]) + (X[2][1] \* X[2][2]) + (X[3][1] \* X[3][2])) / N

print("a12=a21= {0} \n"

"a13=a31= {1} \n"

"a23=a32= {2} ".format(a12, a13, a23))

print("-------------------------")

# розрахунок детермінантів матриці

znamen = np.array([[1, mx1, mx2, mx3],

[mx1, a11, a12, a13],

[mx2, a12, a22, a32],

[mx3, a13, a23, a33]], dtype='float')

ch0 = np.array([[my, mx1, mx2, mx3],

[a1, a11, a12, a13],

[a2, a12, a22, a32],

[a3, a13, a23, a33]], dtype='float')

ch1 = np.array([[1, my, mx2, mx3],

[mx1, a1, a12, a13],

[mx2, a2, a22, a32],

[mx3, a3, a23, a33]], dtype='float')

ch2 = np.array([[1, mx1, my, mx3],

[mx1, a11, a1, a13],

[mx2, a12, a2, a32],

[mx3, a13, a3, a33]], dtype='float')

ch3 = np.array([[1, mx1, mx2, my],

[mx1, a11, a12, a1],

[mx2, a12, a22, a2],

[mx3, a13, a23, a3]], dtype='float')

b0 = np.linalg.det(ch0) / np.linalg.det(znamen)

b1 = np.linalg.det(ch1) / np.linalg.det(znamen)

b2 = np.linalg.det(ch2) / np.linalg.det(znamen)

b3 = np.linalg.det(ch3) / np.linalg.det(znamen)

b = []

b.append(b0)

b.append(b1)

b.append(b2)

b.append(b3)

print("b0= {0} \n"

"b1= {1} \n"

"b2= {2} \n"

"b3= {3}".format(b0, b1, b2, b3))

print("-------------------------")

# нормоване рівняння

print("Y={} + {}\*x1 + {}\*x2 + {}\*x3".format(b0, b1, b2, b3))

print("-------------------------")

# перевірка

print("Перевірка: ")

# for i in range(len(X)):

# for i in range(3):

print(b0 + b1 \* X[0][0] + b2 \* X[0][1] + b3 \* X[0][2], "==", Ys[0])

print(b0 + b1 \* X[1][0] + b2 \* X[1][1] + b3 \* X[1][2], "==", Ys[1])

print(b0 + b1 \* X[2][0] + b2 \* X[2][1] + b3 \* X[2][2], "==", Ys[2])

print(b0 + b1 \* X[3][0] + b2 \* X[3][1] + b3 \* X[3][2], "==", Ys[3])

print("Результат збігається з середніми значеннями")

print("-------------------------")

# перевірка однорідності за Кохреном

# дисперсії по рядках

D = []

Summa = 0

for i in range(N):

for j in range(m):

Summa += pow((Y[i][j] - Ys[i]), 2)

D.append(1 / m \* Summa)

Summa = 0

print("D: ", D)

print("-------------------------")

Gp = max(D) / sum(D)

print("Gp= ", Gp)

f1 = m - 1

f2 = N = 4

if Gp <= Gt[f1]:

print("Дисперсія однорідна")

print(Gp, "<=", Gt[f1])

else:

print("Дисперсія не однорідна")

m += 1

print("M:", m)

return func(m)

##########################

print("############################")

# критерій Стьюдента

S2\_b = sum(D) / N

S2\_betta = S2\_b / (N \* m)

S\_betta = math.sqrt(S2\_betta)

print("S2\_b= {0} \n"

"S2\_betta= {1} \n"

"S\_betta= {2}".format(S2\_b, S2\_betta, S\_betta))

print("-------------------------")

Xs = [[1.0, -1.0, -1.0, -1.0],

[1.0, -1.0, 1.0, 1.0],

[1.0, 1.0, -1.0, 1.0],

[1.0, 1.0, 1.0, -1.0]]

betta = []

betta0 = (Ys[0] \* Xs[0][0] + Ys[1] \* Xs[1][0] + Ys[2] \* Xs[2][0] + Ys[3] \* Xs[3][0]) / N

betta1 = (Ys[0] \* Xs[0][1] + Ys[1] \* Xs[1][1] + Ys[2] \* Xs[2][1] + Ys[3] \* Xs[3][1]) / N

betta2 = (Ys[0] \* Xs[0][2] + Ys[1] \* Xs[1][2] + Ys[2] \* Xs[2][2] + Ys[3] \* Xs[3][2]) / N

betta3 = (Ys[0] \* Xs[0][3] + Ys[1] \* Xs[1][3] + Ys[2] \* Xs[2][3] + Ys[3] \* Xs[3][3]) / N

betta.append(betta0)

betta.append(betta1)

betta.append(betta2)

betta.append(betta3)

print("betta0= {0} \n"

"betta1= {1} \n"

"betta2= {2} \n"

"betta3= {3}".format(betta0, betta1, betta2, betta3))

print("-------------------------")

t = []

for i in range(len(betta)):

t.append(abs(betta[i]) / S\_betta)

print("t: ", t)

print("-------------------------")

f3 = f1 \* f2

for i in range(len(t)):

if t[i] < t\_tabl[f3]:

b[i] = 0

print(t[i], "<", t\_tabl[f3])

y = []

y.append(b0 + b1 \* X[0][0] + b2 \* X[0][1] + b3 \* X[0][2])

y.append(b0 + b1 \* X[1][0] + b2 \* X[1][1] + b3 \* X[1][2])

y.append(b0 + b1 \* X[2][0] + b2 \* X[2][1] + b3 \* X[2][2])

y.append(b0 + b1 \* X[3][0] + b2 \* X[3][1] + b3 \* X[3][2])

print("y: ", y)

for i in range(len(y)):

print(y[i], "==", Ys[i])

print("-------------------------")

# критерій Фішера

d = 0

for i in range(len(b)):

if b[i] != 0:

d += 1

f4 = N - d

Sum = 0

for i in range(len(y)):

Sum += pow((y[i] - Ys[i]), 2)

S\_ad = (m / (N - d)) \* Sum

print("S\_ad= ", S\_ad)

Fp = S\_ad / S2\_b

print("Fp=", Fp)

if Fp > Ft[f3][f4]:

print("Рівняння регресії неадекватно оригіналу при рівні значимості 0.05")

else:

print("Рівняння регресії адекватно оригіналу при рівні значимості 0.05")

func(3)

Контрольні запитання

1. Що називається дробовим факторним експериментом?

У деяких випадках немає необхідності проводити повний факторний експеримент (ПФЕ). Якщо буде використовуватися лінійна регресія, то можливо зменшити кількість рядків матриці ПФЕ до кількості коефіцієнтів регресійної моделі. Кількість дослідів слід скоротити, використовуючи для планування так звані регулярні дробові репліки від повного факторного експерименту, що містять відповідну кількість дослідів і зберігають основні властивості матриці планування – це означає дробовий факторний експеримент (ДФЕ).

2. Для чого потрібно розрахункове значення Кохрена?

Для перевірки дисперсії на однорідність.

3. Для чого перевіряється критерій Стьюдента?

Для перевірки значущості коефіцієнтів регресії. Тобто, Якщо виконується нерівність ts< tтабл, то приймається нуль-гіпотеза, тобто вважається, що знайдений коефіцієнт βs є статистично незначущим і його слід виключити з рівняння регресії. Якщо ts > tтабл то гіпотеза не підтверджується, тобто βs – значимий коефіцієнт і він залишається в рівнянні регресії.

4. Чим визначається критерій Фішера і як його застосовувати?

Отримане рівняння регресії необхідно перевірити на адекватність досліджуваному об'єкту. Для цієї мети необхідно оцінити, наскільки відрізняються середні значення у вихідної величини, отриманої в точках факторного простору, і значення у, отриманого з рівняння регресії в тих самих точках факторного простору. Для цього використовують дисперсію адекватності. Адекватність моделі перевіряють за F-критерієм Фішера, який дорівнює відношенню дисперсії адекватності до дисперсії відтворюваності.