Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Методи наукових досліджень
Лабораторна робота №3
«Проведення трьохфакторного експерименту з використанням лінійного рівняння регресії»

Виконала:
студентка групи ІВ-93
Баранчук І. М.
Варіант: 01
Перевірив:
Регіда П.Г

Київ - 2021 р.

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

Індивідуальне завдання:

301	-10	50	20	60	-10	5

Лістинг коду програми:

```
import random
import math


# Лабораторна робота №3
# Виконала Баранчук Іванна  Варіант - 301

class Lab3:
    x1_min = -10
    x1_max = 50
    x2_min = 20
    x2_max = 60
    x3_min = -10
    x3_max = 5

    gt = 0.7679
    tf = 2.306
    ft = 4.5
    m = 3
    n = 4

    def __init__(self):
        self.y_max = 200 + (self.x1_max + self.x2_max + self.x3_max) / 3
        self.y_min = 200 + (self.x1_min + self.x2_min + self.x3_min) / 3

        self.x1 = [random.randint(self.x1_min, self.x1_max + 1) for i in range(4)]
        self.x2 = [random.randint(self.x2_min, self.x2_max + 1) for i in range(4)]
        self.x3 = [random.randint(self.x3_min, self.x3_max + 1) for i in range(4)]
        self.y1 = [random.randint(int(self.y_min), int(self.y_max) + 1) for i in range(4)]
        self.y2 = [random.randint(int(self.y_min), int(self.y_max) + 1) for i in range(4)]
        self.y3 = [random.randint(int(self.y_min), int(self.y_max) + 1) for i in range(4)]

        self.plan_matrix = [[1, 1, 1, 1],
                            [-1, -1, 1, 1],
                            [-1, 1, -1, 1],
                            [-1, 1, 1, -1]]

        self.average_y = [0, 0, 0, 0]
        for i in range(0, len(self.x1)):
            self.average_y[i] = (self.y1[i] + self.y2[i] + self.y3[i]) / 3
        print(f"Average Y: {self.average_y}")

        self.calculate_and_print()

    @staticmethod
    def func_mx(arr, main_arr):
        main_arr.append(sum(arr) / len(arr))

    @staticmethod
    def func_aii(arr, main_arr):
        main_arr.append((arr[0] ** 2 + arr[1] ** 2 + arr[2] ** 2 + arr[3] ** 2) /
len(arr))
```

```
    @staticmethod
    def func_aij(arr1, arr2, main_arr):
        main_arr.append((arr1[0] * arr2[0] + arr1[1] * arr2[1] + arr1[2] * arr2[2] +
arr1[3] * arr2[3]) / len(arr1))

    def func_a(self, arr, main_arr):
        main_arr.append(
            (arr[0] * self.average_y[0] + arr[1] * self.average_y[1] + arr[2] *
self.average_y[2] + arr[3] * self.average_y[3]) / len(arr))

    def calculate_and_print(self):
        mx = []
        Lab3.func_mx(self.x1, mx)
        Lab3.func_mx(self.x2, mx)
        Lab3.func_mx(self.x3, mx)
        print(f"\nMX: {mx}")

        my = (self.average_y[0] + self.average_y[1] + self.average_y[2] +
self.average_y[3]) / len(self.average_y)
        print(f"MY: {my}")

        a = []
        self.func_a(self.x1, a)
        self.func_a(self.x2, a)
        self.func_a(self.x3, a)
        print(f"\nA: {a}")

        a11 = []
        Lab3.func_aii(self.x1, a11)
        print(f"A11: {a11}")

        a22 = []
        Lab3.func_aii(self.x2, a22)
        print(f"A22: {a22}")

        a33 = []
        Lab3.func_aii(self.x3, a33)
        print(f"A33: {a33}")

        a12 = a21 = []
        Lab3.func_aij(self.x1, self.x2, a12)
        print(f"A12 = A21: {a12}")

        a13 = a31 = []
        Lab3.func_aij(self.x1, self.x3, a13)
        print(f"A13 = A31: {a13}")

        a23 = a32 = []
        Lab3.func_aij(self.x2, self.x3, a23)
        print(f"A23 = A32: {a23}")

        r01 = [1, mx[0], mx[1], mx[2]]
        r02 = [mx[0], a11[0], a12[0], a13[0]]
        r03 = [mx[1], a12[0], a22[0], a32[0]]
        r04 = [mx[2], a13[0], a23[0], a33[0]]
        temp_0 = [r01, r02, r03, r04]

        determinant = 1 * a11[0] * a22[0] * a33[0] + mx[0] * a12[0] * a32[0] * mx[2] +
mx[1] * a13[0] * mx[1] * a13[0] + mx[
            2] * mx[0] * a12[0] * a23[0] - \
                 (mx[2] * a12[0] * a12[0] * mx[2] + a13[0] * a22[0] * a13[0] * 1 + a23[0]
* a32[0] * mx[0] * mx[0] +
                  a33[0] * mx[1] * a11[0] * mx[1])

        r11 = [my, mx[0], mx[1], mx[2]]
        r12 = [a[0], a11[0], a12[0], a13[0]]
        r13 = [a[1], a12[0], a22[0], a32[0]]
```

```
        r14 = [a[2], a13[0], a23[0], a33[0]]
        temp_1 = [r11, r12[0], r13[0], r14[0]]

        determinant_1 = my * a11[0] * a22[0] * a33[0] + a[0] * a12[0] * a32[0] * mx[2] +
a[1] * a13[0] * mx[1] * a13[0] + a[
            2] * mx[0] * a12[0] * a23[0] - \
                  (a[2] * a12[0] * a12[0] * mx[2] + a13[0] * a22[0] * a13[0] * my + a23[0]
* a32[0] * a[0] * mx[0] +
                   a33[0] * a[1] * a11[0] * mx[1])

        r21 = [1, my, mx[1], mx[2]]
        r22 = [mx[0], a[0], a12[0], a13[0]]
        r23 = [mx[1], a[1], a22[0], a32[0]]
        r24 = [mx[2], a[2], a23[0], a33[0]]
        temp_2 = [r21, r22, r23, r24]

        determinant_2 = 1 * a[0] * a22[0] * a33[0] + my * a12[0] * a32[0] * mx[2] + mx[1]
* a13[0] * mx[1] * a[2] + mx[2] * \
                  mx[0] * a[1] * a23[0] - \
                  (mx[2] * a[1] * a12[0] * mx[2] + a[2] * a22[0] * a13[0] * 1 + a23[0] *
a32[0] * my * mx[0] + a33[0] *
                   mx[1] * a[0] * mx[1])

        r31 = [1, mx[0], my, mx[2]]
        r32 = [mx[0], a11[0], a[0], a13[0]]
        r33 = [mx[1], a12[0], a[1], a32[0]]
        r34 = [mx[2], a13[0], a[2], a33[0]]
        temp_3 = [r31, r32, r33, r34]

        determinant_3 = 1 * a11[0] * a[1] * a33[0] + mx[0] * a[0] * a32[0] * mx[2] + my *
a13[0] * mx[1] * a13[0] + mx[2] * \
                  mx[0] * a12[0] * a[2] - \
                  (mx[2] * a12[0] * a[0] * mx[2] + a13[0] * a[1] * a13[0] * 1 + a[2] *
a32[0] * mx[0] * mx[0] + a33[0] *
                   mx[1] * a11[0] * my)

        r41 = [1, mx[0], mx[1], my]
        r42 = [mx[0], a11[0], a12[0], a[0]]
        r43 = [mx[1], a12[0], a22[0], a[1]]
        r44 = [mx[2], a13[0], a23[0], a[2]]
        temp_4 = [r41, r42, r43, r44]

        determinant_4 = 1 * a11[0] * a22[0] * a[2] + mx[0] * a12[0] * a[1] * mx[2] + mx[1]
* a[0] * mx[1] * a13[0] + my * mx[
            0] * a12[0] * a23[0] - \
                  (mx[2] * a12[0] * a12[0] * my + a13[0] * a22[0] * a[0] * 1 + a23[0] *
a[1] * mx[0] * mx[0] + a[2] *
                   mx[1] * a11[0] * mx[1])

        b = [0, 0, 0, 0]
        b[0] = determinant_1 / determinant
        b[1] = determinant_2 / determinant
        b[2] = determinant_3 / determinant
        b[3] = determinant_4 / determinant

        print("\n")
        for i in range(4):
            print(f"b{i}: {b[i]}")
        print("\n")

        y = []
        for i in range(len(self.x1)):
            y.append(b[0] + b[1] * self.x1[i] + b[2] * self.x2[i] + b[3] * self.x3[i])
            print(f"Y{i + 1}: {y[i]}")

        dispersion = [0, 0, 0, 0]
        for i in range(0, len(dispersion)):
```

```
            dispersion[i] = ((self.y1[i] - self.average_y[i]) ** 2 + (self.y2[i] -
self.average_y[i]) ** 2 + (self.y3[i] - self.average_y[i]) ** 2) / 3
        print(f"Dispersion: {dispersion}")

        print('\nПеревірка однорідності дисперсії за критерієм Кохрена:')
        gp = max(dispersion) / sum(dispersion)
        if gp < self.gt:
            print("Дисперсія однорідна")
        else:
            print('Дисперсія не однорідна. Потрібно збільшити m')
            return

        sb = sum(dispersion) / len(dispersion)
        sb2_uniform = sb / (self.n * self.m)
        sb_uniform = math.sqrt(sb2_uniform)

        beta = [0, 0, 0, 0]
        beta[0] = (self.average_y[0] * 1 + self.average_y[1] * 1 + self.average_y[2] * 1 +
self.average_y[3] * 1) / self.n
        beta[1] = (self.average_y[0] * (-1) + self.average_y[1] * (-1) + self.average_y[2]
* 1 + self.average_y[3] * 1) / self.n
        beta[2] = (self.average_y[0] * (-1) + self.average_y[1] * 1 + self.average_y[2] *
(-1) + self.average_y[3] * 1) / self.n
        beta[3] = (self.average_y[0] * (-1) + self.average_y[1] * 1 + self.average_y[2] *
1 + self.average_y[3] * (-1)) / self.n
        print(f"\nBeta: {beta}")

        t = []
        for i in range(len(beta)):
            t.append(abs(beta[i]) / sb_uniform)
        print(f"t0: {t}")

        print('\nОцінка значимості коефіцієнтів регресії згідно критерію Стьюдента:')
        d = 0  # кількість значимих коефіцієнтів
        temp = [0, 0, 0, 0]
        n = 4
        tf = 2.306

        for i in range(0, n):
            if t[i] <= tf:
                temp[i] = 0
            else:
                temp[i] = b[i]
                d += 1

        y_2 = []
        for i in range(0, n):
            y_2.append(temp[0] + temp[1] * self.x1[i] + temp[2] * self.x2[i] + temp[3] *
self.x3[i])
        print(f"y_2: {y_2}")

        # Критерій Фішера
        print("\nКритерій Фішера")
        s_ad = []
        tmp = []
        kof = self.m / (n - d)
        for i in range(len(y_2)):
            tmp.append((y_2[i] - self.average_y[i]) ** 2)
        s_ad.append(kof * sum(tmp))
        print(f"S_ad: {s_ad}")

        fp = s_ad[0] / sb2_uniform
        print(f"Fp: {fp}\n")
        if fp > self.ft:
            print("Рівняння регресії неадекватно оригіналу при рівні значимості 0.05")
        else:
            print("Рівняння регресії адекватно оригіналу при рівні значимості 0.05")
```

```
        print(f"y = {b[0]} + {b[1]} * x1 + {b[2]} * x2 + {b[3]} * x3")

Lab3()
```

Результати роботи програми:

```
Average Y: [218.66666666666666, 208.33333333333334, 225.0, 218.0]

MX: [17.25, 45.5, -6.5]
MY: 217.5

A: [3762.75, 9941.833333333332, -1427.1666666666665]
A11: [673.25]
A22: [2246.5]
A33: [51.5]
A12 = A21: [814.5]
A13 = A31: [-103.75]
A23 = A32: [-291.5]


b0: 205.03377558358403
b1: 1.8663058633414074
b2: 0.27335718731196845
b3: -23.904162237104316


Y1: 496.79409710686804
Y2: 286.58712103812786
Y3: 434.70636109289956
Y4: 402.0818539224882
Dispersion: [96.88888888888887, 38.22222222222222, 114.0, 68.66666666666667]

Перевірка однорідності дисперсії за критерієм Кохрена:
Дисперсія однорідна


Beta: [217.5, 4.0, -4.333333333333329, -0.8333333333333286]
t0: [84.5314109698841, 1.5546006615151098, 1.6841507166413672, 0.32387513781564603]

Оцінка значимості коефіцієнтів регресії згідно критерію Стьюдента:
y_2: [205.03377558358403, 205.03377558358403, 205.03377558358403, 205.03377558358403]

Критерій Фішера
S_ad: [763.5158936906713]
Fp: 115.32827485117832

Рівняння регресії неадекватно оригіналу при рівні значимості 0.05
y = 205.03377558358403 + 1.8663058633414074 * x1 + 0.27335718731196845 * x2 + -23.904162237104316 * x3
```

Відповіді на контрольні питання:

1. З чого складається план експерименту?

Сукупність усіх точок плану - векторів Xi (для i = 1, 2, . . . , N) утворює план експерименту. Таким чином, план експерименту описується матрицею, яка містить N рядків і K стовбців. Кожен рядок матриці означає точку плану експерименту, а стовпчик – фактор експерименту.

2. Що називається спектром плану?

Сукупність усіх точок плану, що відрізняються рівнем хоча б одного фактора (різних строк матриці планування), називається спектром плану. Матриця, отримана із усіх різних строк плану називається матрицею спектра плану.

3. Чим відрізняються активні та пасивні експерименти?

Експерименти поділяють на пасивні та активні (керовані). В пасивному експерименті існують контрольовані, але некеровані вхідні параметри – ми немаємо можливості втручатись в хід проведення експерименту, і виступаємо в ролі пасивного користувача. В активному – існують керовані і контрольовані вхідні параметри – ми самі являємось адміністраторами нашої системи.

4. Чим характеризується об'єкт досліджень? Дайте визначення факторному простору.

Об'єкт досліджень розглядається як «чорний ящик». Аналізуються деякі властивості та якості, які можуть описуватися числовими значеннями. Вектор X1…XK представляє собою групу контрольованих та керованих величин, котрі можуть змінюватись необхідним чином при проведенні експерименту, Цю групу характеристик X1…XK також називають факторами або керованими впливами. Факторний простір - простір незалежних змінних(факторів), діапазон значень факторів.