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 №3

З дисципліни «Методи наукових досліджень»
з теми: “ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ”

ВИКОНАВ:
Студент ІІ курсу ФІОТ
Групи ІВ-91
Дерачиц Віталій Віталійович
Номер заліковки: 9109
Номер у списку: 9

ПЕРЕВІРИВ:
ас. Регіда П. Г.

Київ 2021 р.

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

Завдання на лабораторну роботу

1. Скласти матрицю планування для дробового трьохфакторного експерименту. Провести експеримент в усіх точках факторного простору, повторивши N експериментів, де N – кількість експериментів (рядків матриці планування) в усіх точках факторного простору – знайти значення функції відгуку У. Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі (випадковим чином).

$$y_{max} = 200 + x_{cp\,max};$$

$$y_{min} = 200 + x_{cp\,min}$$

де $x_{cp\,max} = \frac{x_{1max} + x_{2max} + x_{3max}}{3}$, $x_{cp\,min} = \frac{x_{1min} + x_{2min} + x_{3min}}{3}$

2. Знайти коефіцієнти лінійного рівняння регресії. Записати лінійне рівняння регресії.

3. Провести 3 статистичні перевірки.

4. Написати комп'ютерну програму, яка усе це виконує.

Завдання за варіантом:

109	-20	15	10	60	15	35

Програмний код

```
from random import randint
from functools import reduce
from numpy.linalg import det


def naturalize(matrix_of_plan, min_max_arr):
    result = []
    for i in matrix_of_plan:
        result.append(min_max_arr[1]) if i == 1 else
result.append(min_max_arr[0])
    return result


def main():
    x1 = [-20, 15]
    x2 = [10, 60]
    x3 = [15, 35]

    print("x1: ", x1)
    print("x2: ", x2)
    print("x3: ", x3)

    x0_plan_array = [1, 1, 1, 1]
    x1_plan_array = [-1, -1, 1, 1]
    x2_plan_array = [-1, 1, -1, 1]
    x3_plan_array = [-1 * (x1_plan_array[i] * x2_plan_array[i]) for i in
range(len(x1_plan_array))]
```

```
    print("\nx0:", x0_plan_array)
    print("x1:", x1_plan_array)
    print("x2:", x2_plan_array)
    print("x3:", x3_plan_array)

    x1_plan_naturalized = naturalize(x1_plan_array, x1)
    x2_plan_naturalized = naturalize(x2_plan_array, x2)
    x3_plan_naturalized = naturalize(x3_plan_array, x3)

    print('\nx1:', x1_plan_naturalized)
    print('x2:', x2_plan_naturalized)
    print('x3:', x3_plan_naturalized)

    x_avg_max = (max(x1_plan_naturalized) + max(x2_plan_naturalized) +
max(x3_plan_naturalized)) / 3
    x_avg_min = (min(x1_plan_naturalized) + min(x2_plan_naturalized) +
min(x3_plan_naturalized)) / 3

    print("\nx_avg_max = ", x_avg_max)
    print("x_avg_min = ", x_avg_min)

    y_min = int(200 + x_avg_min)
    y_max = int(200 + x_avg_max)

    print("\ny_max = ", y_max)
    print("y_min = ", y_min)

    y1 = [randint(y_min, y_max) for i in range(4)]
    y2 = [randint(y_min, y_max) for i in range(4)]
    y3 = [randint(y_min, y_max) for i in range(4)]

    print("\ny1:", y1)
    print("y2:", y2)
    print("y3:", y3)

    y_avg_array = [(y1[i] + y2[i] + y3[i]) / 3 for i in range(4)]
    print("\nAverage y: ", y_avg_array)

    mx1 = reduce(lambda a, b: a + b, x1_plan_naturalized) / 4
    mx2 = reduce(lambda a, b: a + b, x2_plan_naturalized) / 4
    mx3 = reduce(lambda a, b: a + b, x3_plan_naturalized) / 4
    my = reduce(lambda a, b: a + b, y_avg_array) / 4

    print("\nmx1 = ", mx1)
    print("mx2 = ", mx2)
    print("mx3 = ", mx3)
    print("my = ", my)

    a1 = sum([x1_plan_naturalized[i] * y_avg_array[i] for i in range(4)]) / 4
    a2 = sum([x2_plan_naturalized[i] * y_avg_array[i] for i in range(4)]) / 4
    a3 = sum([x3_plan_naturalized[i] * y_avg_array[i] for i in range(4)]) / 4

    a11 = sum([i * i for i in x1_plan_naturalized]) / 4
    a22 = sum([i * i for i in x2_plan_naturalized]) / 4
    a33 = sum([i * i for i in x3_plan_naturalized]) / 4

    a12 = sum([x1_plan_naturalized[i] * x2_plan_naturalized[i] for i in
range(4)]) / 4
    a13 = sum([x1_plan_naturalized[i] * x3_plan_naturalized[i] for i in
range(4)]) / 4
    a23 = sum([x2_plan_naturalized[i] * x3_plan_naturalized[i] for i in
range(4)]) / 4

    a21 = a12
```

```
    a31 = a13
    a32 = a23

    print("\na1 = ", a1)
    print("a2 = ", a2)
    print("a3 = ", a3)

    print("\na11 = ", a11)
    print("a22 = ", a22)
    print("a33 = ", a33)

    print("\na12 = ", a12)
    print("a13 = ", a13)
    print("a23 = ", a23)

    print("\na21 = ", a21)
    print("a31 = ", a31)
    print("a32 = ", a32)

    b0 = det([[my, mx1, mx2, mx3],
              [a1, a11, a12, a13],
              [a2, a21, a22, a23],
              [a3, a31, a32, a33]]) / det([[1, mx1, mx2, mx3],
                                           [mx1, a11, a12, a13],
                                           [mx2, a21, a22, a23],
                                           [mx3, a31, a32, a33]])
    b1 = det([[1, my, mx2, mx3],
              [mx1, a1, a12, a13],
              [mx2, a2, a22, a23],
              [mx3, a3, a32, a33]]) / det([[1, mx1, mx2, mx3],
                                           [mx1, a11, a12, a13],
                                           [mx2, a21, a22, a23],
                                           [mx3, a31, a32, a33]])
    b2 = det([[1, mx1, my, mx3],
              [mx1, a11, a1, a13],
              [mx2, a21, a2, a23],
              [mx3, a31, a3, a33]]) / det([[1, mx1, mx2, mx3],
                                           [mx1, a11, a12, a13],
                                           [mx2, a21, a22, a23],
                                           [mx3, a31, a32, a33]])
    b3 = det([[1, mx1, mx2, my],
              [mx1, a11, a12, a1],
              [mx2, a21, a22, a2],
              [mx3, a31, a32, a3]]) / det([[1, mx1, mx2, mx3],
                                           [mx1, a11, a12, a13],
                                           [mx2, a21, a22, a23],
                                           [mx3, a31, a32, a33]])

    print("\ny = b0 + b1*x1 + b2*x2 + b3*x3")
    print(f"y = {b0} + {b1}*x1 + {b2}*x2 + b3*x3")

    for i in range(4):
        y = b0 + b1 * x1_plan_naturalized[i] + b2 * x2_plan_naturalized[i] +
b3 * x3_plan_naturalized[i]

    dispersion = [((y1[i] - y_avg_array[i]) ** 2 + (y2[i] - y_avg_array[i])
** 2 + (y3[i] - y_avg_array[i]) ** 2) / 3
                  for i in
                  range(4)]

    print("\ndispersion: ", dispersion)

    gp = max(dispersion) / sum(dispersion)
```

```
    print("\nКоефіцієнт Gp = ", gp)

    m = 3
    # f1=m-1=2, f2=N=4, q=0.05 => Gт=0.7679 за таблицею
    if gp > 0.7679:
        print("\nДисперсія неоднорідна!")
        exit()

    print("\nGp < 0.7679 => Дисперсія однорідна")

    # ПЕРЕВІРКА ЗНАЧУЩОСТІ КОЕФІЦІЄНТІВ ЗА КРИТЕРІЄМ СТЬЮДЕНТА
    s2b = sum(dispersion) / 4
    s2bs_avg = s2b / (4 * m)
    sb = s2bs_avg ** (1 / 2)

    beta0 = sum([y_avg_array[i] * x0_plan_array[i] for i in range(4)]) / 4
    beta1 = sum([y_avg_array[i] * x1_plan_array[i] for i in range(4)]) / 4
    beta2 = sum([y_avg_array[i] * x2_plan_array[i] for i in range(4)]) / 4
    beta3 = sum([y_avg_array[i] * x3_plan_array[i] for i in range(4)]) / 4

    beta_array = [beta0, beta1, beta2, beta3]

    print("\nbeta: ", beta_array)

    t_array = [abs(beta_array[i]) / sb for i in range(4)]

    print("\nt: ", t_array)
    print()

    # f3 = f1*f2 = 2*4 = 8 => за таблицею значення t-критерію = 2.306
    d = 0
    indexes = []
    for i, v in enumerate(t_array):
        if t_array[i] > 2.306:
            indexes.append(i)
            d += 1
        else:
            print(f"Коефіцієнт b{i} = {v} є статистично незначущим і його
слід виключити з рівняння регресії.")

    b_array = [b0, b1, b2, b3]

    b_result = [b_array[indexes[0]] for i in range(4)]

    # ПЕРЕВІРКА АДЕКВАТНОСТІ ЗА КРИТЕРІЄМ ФІШЕРА
    s2_ad = m * sum([(y_avg_array[i] - b_result[i]) ** 2 for i in range(4)])
/ (4 - d)
    fp = s2_ad / s2b

    print("\nFp = ", fp)

    # q = 0.05, f3 = f1*f2 = 8, f4 = N - d = 3 => Fт = 4.1
    if fp < 4.1:
        print("Fp < Fт. "
              "Отримана математична модель з прийнятим рівнем статистичної
значимості q адекватна оригіналу")
    else:
        print("Fp > Fт. "
            "Отримана математична модель з прийнятим рівнем статистичної
значимості q не адекватна оригіналу")


if __name__ == '__main__':
    main()
```

Результати роботи програми

```
C:\Users\derac\Anaconda3\python.exe C:/Users/derac/PycharmProjects/MND_lab3/main.py
x1:  [-20, 15]
x2:  [10, 60]
x3:  [15, 35]

x0: [1, 1, 1, 1]
x1: [-1, -1, 1, 1]
x2: [-1, 1, -1, 1]
x3: [-1, 1, 1, -1]

x1: [-20, -20, 15, 15]
x2: [10, 60, 10, 60]
x3: [15, 35, 35, 15]

x_avg_max =  36.666666666666664
x_avg_min =  1.666666666666667

y_max =  236
y_min =  201

y1: [216, 229, 209, 226]
y2: [230, 226, 201, 212]
y3: [209, 222, 226, 230]

Average y:  [218.33333333333334, 225.66666666666666, 212.0, 222.66666666666666]

mx1 =  -2.5
mx2 =  35.0
mx3 =  25.0
my =  219.66666666666666

a1 =  -590.0
a2 =  7800.833333333334
a3 =  5483.333333333333

a11 =  312.5
a22 =  1850.0
a33 =  725.0

a12 =  -87.5
a13 =  -62.5
a23 =  875.0
```

```
a21 =  -87.5
a31 =  -62.5
a32 =  875.0

y = b0 + b1*x1 + b2*x2 + b3*x3
y = 215.11666666666594 + -0.13333333333333328*x1 + 0.18000000000000035*x2 + b3*x3

dispersion:  [76.22222222222221, 8.222222222222223, 108.66666666666667, 59.55555555555555]

Коефіцієнт Gp =  0.43007915567282323

Gp < 0.7679 => Дисперсія однорідна

beta:  [219.66666666666666, -2.3333333333333357, 4.499999999999993, -0.8333333333333357]

t:  [95.7438314801051, 1.017005797209008, 1.9613683231887964, 0.363216356146075]

Коефіцієнт b1 = 1.017005797209008 є статистично незначущим і його слід виключити з рівняння регресії.
Коефіцієнт b2 = 1.9613683231887964 є статистично незначущим і його слід виключити з рівняння регресії.
Коефіцієнт b3 = 0.363216356146075 є статистично незначущим і його слід виключити з рівняння регресії.

Fp =  2.9820404573443042
Fp < Ft. Отримана математична модель з прийнятим рівнем статистичної значимості q адекватна оригіналу

Process finished with exit code 0
```

Контрольні запитання:

1. **Що називається дробовим факторним експериментом?**
   Дробовий факторний експеримент – частина ПФЕ, який мінімізує число дослідів, за рахунок тієї інформації, яка не дуже істотна для побудови моделі
2. **Для чого потрібно розрахункове значення Кохрена?**
   Значення Кохрена використовують для перевірки однорідності дисперсії
3. **Для чого перевіряється критерій Стьюдента?**
   Критерій Стьюдента перевіряє значущість коефіцієнтів рівняння
4. **Чим визначається критерій Фішера і як його застосовувати?**
   Критерій Фішера використовують при перевірці отриманого рівняння регресії досліджуваному об’єкту