

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»  
**ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З**  
**ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ**

ВИКОНАВ:  
Студентка II курсу ФІОТ  
Групи ІВ-91  
Богомол В.Ю. - 9101

ПЕРЕВІРИВ:  
асистент  
Регіда П.Г.

Київ 2021 р.

**Мета:**

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

**Варіант завдання:**

Варіант	X <sub>1</sub>		X <sub>2</sub>		X <sub>3</sub>	
	min	max	min	max	min	max
101	-10	50	20	60	50	55

**Лістинг програми:**

```
from random import *
import numpy as np
from numpy.linalg import solve
from scipy.stats import f, t
from functools import partial

class FractionalExperiment:
    """Проведення дробового трьохфакторного експерименту"""

    def __init__(self, n, m):
        self.n = n
        self.m = m
        self.x_min = (-10 + 20 + 50) / 3
        self.x_max = (50 + 60 + 55) / 3
        self.y_max = round(200 + self.x_max)
        self.y_min = round(200 + self.x_min)
        self.x_norm = [[1, -1, -1, -1],
                        [1, -1, 1, 1],
                        [1, 1, -1, 1],
                        [1, 1, 1, -1],
                        [1, -1, -1, 1],
                        [1, -1, 1, -1],
                        [1, 1, -1, -1],
                        [1, 1, 1, 1]]
        self.x_range = [(-10, 50), (20, 60), (50, 55)]
        self.y = np.zeros(shape=(self.n, self.m))
        self.y_new = []
        for i in range(self.n):
            for j in range(self.m):
                self.y[i][j] = randint(self.y_min, self.y_max)
        self.y_av = [round(sum(i) / len(i), 2) for i in self.y]
        self.x_norm = self.x_norm[:len(self.y)]
        self.x = np.ones(shape=(len(self.x_norm), len(self.x_norm[0])))
        for i in range(len(self.x_norm)):
            for j in range(1, len(self.x_norm[i])):
                if self.x_norm[i][j] == -1:
                    self.x[i][j] = self.x_range[j - 1][0]
                else:
                    self.x[i][j] = self.x_range[j - 1][1]
        self.f1 = m - 1
        self.f2 = n
        self.f3 = self.f1 * self.f2
        self.q = 0.05

    def regression(self, x, b):
        """Підстановка коефіцієнтів у рівняння регресії"""
```

```

        y = sum([x[i] * b[i] for i in range(len(x))])
        return y

    def count_koefs(self):
        """Розрахунок коефіцієнтів рівняння регресії"""
        mx1 = sum(self.x[:, 1]) / self.n
        mx2 = sum(self.x[:, 2]) / self.n
        mx3 = sum(self.x[:, 3]) / self.n
        my = sum(self.y_av) / self.n
        a12 = sum([self.x[i][1] * self.x[i][2] for i in range(len(self.x))]) /
        self.n
        a13 = sum([self.x[i][1] * self.x[i][3] for i in range(len(self.x))]) /
        self.n
        a23 = sum([self.x[i][2] * self.x[i][3] for i in range(len(self.x))]) /
        self.n
        a11 = sum([i ** 2 for i in self.x[:, 1]]) / self.n
        a22 = sum([i ** 2 for i in self.x[:, 2]]) / self.n
        a33 = sum([i ** 2 for i in self.x[:, 3]]) / self.n
        a1 = sum([self.y_av[i] * self.x[i][1] for i in range(len(self.x))]) /
        self.n
        a2 = sum([self.y_av[i] * self.x[i][2] for i in range(len(self.x))]) /
        self.n
        a3 = sum([self.y_av[i] * self.x[i][3] for i in range(len(self.x))]) /
        self.n

        X = [[1, mx1, mx2, mx3], [mx1, a11, a12, a13], [mx2, a12, a22, a23],
        [mx3, a13, a23, a33]]
        Y = [my, a1, a2, a3]
        B = [round(i, 2) for i in solve(X, Y)]
        print('\nРівняння регресії')
        print(f'y = {B[0]} + {B[1]}*x1 + {B[2]}*x2 + {B[3]}*x3')

        return B

    def dispersion(self):
        """Розрахунок дисперсії"""
        res = []
        for i in range(self.n):
            s = sum([(self.y_av[i] - self.y[i][j]) ** 2 for j in
            range(self.m)]) / self.m
            res.append(s)
        return res

    def kohren(self):
        """Перевірка однорідності дисперсій за критерієм Кохрена"""
        q1 = self.q / self.f1
        fisher_value = f.ppf(q=1 - q1, dfn=self.f2, dfd=(self.f1 - 1) *
        self.f2)
        G_cr = fisher_value / (fisher_value + self.f1 - 1)
        s = self.dispersion()
        Gp = max(s) / sum(s)
        return Gp, G_cr

    def student(self):
        """Перевірка значущості коефіцієнтів за критерієм Стьюдента"""

        def bs():
            res = [sum(1 * y for y in self.y_av) / self.n]
            for i in range(3): # 4 - ксть факторів
                b = sum(j[0] * j[1] for j in zip(self.x[:, i], self.y_av)) /
                self.n
                res.append(b)
            return res

```

```

S_kv = self.dispersion()
s_kv_aver = sum(S_kv) / self.n

# статистична оцінка дисперсії
s_Bs = (s_kv_aver / self.n / self.m) ** 0.5
Bs = bs()
ts = [abs(B) / s_Bs for B in Bs]
return ts

def fisher(self, d):
    """Перевірка адекватності за критерієм Фішера"""
    S_ad = self.m / (self.n - d) * sum([(self.y_new[i] - self.y_av[i]) **
2 for i in range(len(self.y))])
    S_kv = self.dispersion()
    S_kv_aver = sum(S_kv) / self.n
    F_p = S_ad / S_kv_aver
    return F_p

def check(self):
    """Проведення статистичних перевірок"""
    student = partial(t.ppf, q=1 - 0.025)
    t_student = student(df=self.f3)

    print('\nПеревірка за критерієм Кохрена')
    Gp, G_kr = self.kohren()
    print(f'Gp = {Gp}')
    if Gp < G_kr:
        print(f'З ймовірністю {1-self.q} дисперсії однорідні.')
    else:
        print("Необхідно збільшити кількість дослідів")
        self.m += 1
        FractionalExperiment(self.n, self.m)

    ts = self.student()
    print('\nПеревірка значущості коефіцієнтів за критерієм Стьюдента')
    print('Критерій Стьюдента:\n', ts)
    res = [t for t in ts if t > t_student]
    B = self.count_koefs()
    final_k = [B[ts.index(i)] for i in ts if i in res]
    print('Коефіцієнти {} статистично незначущі, тому ми виключаємо їх з
рівняння.'.format(
        [i for i in B if i not in final_k]))

    for j in range(self.n):
        self.y_new.append(self.regression([self.x[j][ts.index(i)] for i
in ts if i in res], final_k))

    print(f'\nЗначення "y" з коефіцієнтами {final_k}')
    print(self.y_new)

    d = len(res)
    f4 = self.n - d
    F_p = self.fisher(d)

    fisher = partial(f.ppf, q=1 - 0.05)
    f_t = fisher(dfn=f4, dfd=self.f3) # табличне знач
    print('\nПеревірка адекватності за критерієм Фішера')
    print('Fp =', F_p)
    print('F_t =', f_t)
    if F_p < f_t:
        print('Математична модель адекватна експериментальним даним')
    else:
        print('Математична модель не адекватна експериментальним даним')

```

```
experiment = FractionalExperiment(7, 8)
experiment.check()
```

### Контрольні запитання:

1. Що називається дробовим факторним експериментом?  
Дробовим факторним експериментом називається експеримент з використанням частини повного факторного експерименту.
2. Для чого потрібно розрахункове значення Кохрена?  
Розрахункове значення Кохрена використовують для перевірки однорідності дисперсій.
3. Для чого перевіряється критерій Стюдента?  
За допомогою критерію Стюдента перевіряється значущість коефіцієнтів рівняння регресії.
4. Чим визначається критерій Фішера і як його застосовувати?  
Критерій Фішера використовують при перевірці отриманого рівняння регресії досліджуваному об'єкту.

### Результат виконання роботи:

```
C:\Users\vbogo\AppData\Local\Programs\Python\Python38\python.exe C:/Users/vbogo/PycharmProjects/MND3/main.py

Перевірка за критерієм Кохрена
Gr = 0.2356291355428581
З ймовірністю 0.95 дисперсії однорідні.

Перевірка значущості коефіцієнтів за критерієм Стюдента
Критерій Стюдента:
[170.85548247973128, 170.85548247973128, 2679.5002660882105, 6335.323416789904]

Рівняння регресії
y = 235.14 + -0.01*x1 + -0.04*x2 + 0.1*x3
Коефіцієнти [-0.01] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [235.14, 235.14, -0.04, 0.1]
[474.47999999999996, 473.38, 474.97999999999996, 472.88, 474.97999999999996, 472.88, 474.47999999999996]

Перевірка адекватності за критерієм Фішера
Fr = 9449.49899508464
F_t = 2.7939488515842408
Математична модель не адекватна експериментальним даним

Process finished with exit code 0
```

### Висновок:

В даній лабораторній роботі я провів дробовий трьохфакторний експеримент з трьома статистичними перевірками і отримав коефіцієнти рівняння регресії.