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

Методи наукових досліджень

Лабораторна робота №6

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З КВАДРАТИЧНИМИ ЧЛЕНАМИ»

Виконав:
Студент групи ІВ-91
Хандельди О.Р.
Варіант 126

Перевірив:
Ас. Регіда П.Г.

Київ 2021 р.

Мета роботи: Провести трьохфакторний експеримент і отримати адекватну модель – рівняння регресії, використовуючи **рототабельний** композиційний план.

Завдання до лабораторної роботи:

1. Ознайомитися з теоретичними відомостями.
2. Вибрати з таблиці варіантів і записати в протокол інтервали значень x_1, x_2, x_3. Обчислити і записати значення, відповідні кодованим значенням факторів +1; -1;+ l; - l; 0 для $\bar{x}_1$, $\bar{x}_2$, $\bar{x}_3$.
3. Значення функції відгуку знайти за допомогою підстановки в формулу:

$$y_i = f(x_1, x_2, x_3) + random(10)-5,$$

де $f(x_1, x_2, x_3)$ вибирається по номеру в списку в журналі викладача.

4. Провести експерименти і аналізуючи значення статистичних перевірок, отримати адекватну модель рівняння регресії. При розрахунках використовувати натуральні значення факторів.
5. Зробити висновки по виконаній роботі.

126	10	60	15	50	15	20	3,3+7,7*x1+3,8*x2+1,1*x3+2,9*x1*x1+0,5*x2*x2+9,6*x3*x3+4,3*x1*x2+0,1*x1*x3+4,9*x2*x3+3,2*x1*x2*x3

Лістинг програми:

```
from random import randint
from numpy.linalg import det
from copy import deepcopy
from scipy.stats import t


def Naturalize(MatrixOfPlan, MinMaxArr, flag):
    result = []
    for i in range(len(MatrixOfPlan)):
        if i < 8:
            result.append(MinMaxArr[1]) if MatrixOfPlan[i] == 1 else
result.append(MinMaxArr[0])
        else:
            x0 = (max(MinMaxArr) + min(MinMaxArr)) / 2
            dx = x0 - min(MinMaxArr)
            value = None
            if flag == 1:
                value = MatrixOfPlan[i] * dx + x0 if i == 8 or 9 else x0
            elif flag == 2:
                value = MatrixOfPlan[i] * dx + x0 if i == 10 or 11 else x0
            elif flag == 3:
                value = MatrixOfPlan[i] * dx + x0 if i == 12 or 13 else x0
            result.append(value)
    return result


def y_func(x, i):
    return 3.3 + 7.7 * x[0][i] + 3.8 * x[1][i] + 1.1 * x[2][i] + 4.3 * x[3][i] + 0.1
* x[4][i] + 4.9 * x[5][i] +\
           3.2 * x[6][i] + 2.9 * x[7][i] + 0.5 * x[8][i] + 9.6 * x[9][i]


def Cochran(y_arr, y_avg, m):
    # Перевірка однорідності дисперсії за критерієм Кохрена
    print('Перевірка однорідності дисперсії за критерієм Кохрена')
    dispersion = []
    for i in range(len(y_arr[0])):
        current_sum = 0
        for j in range(len(y_arr)):
            current_sum += (y_arr[j][i] - y_avg[i]) ** 2
        dispersion.append(current_sum / len(y_arr))

    print('dispersion:', dispersion)
```

```
    gp = max(dispersion) / sum(dispersion)
    print('Gp =', gp)

    # Рівень значимості q = 0.05
    # f1 = m - 1
    # f2 = N
    print(f'm = {m}')

    gt = 0.3346
    print(f'Gt = {gt}')
    # За таблицею Gт = 0.3346
    if gp < gt:
        print('Дисперсія однорідна')
        return dispersion

    print('Дисперсія неоднорідна')
    return None


def Students(plan1x0, plan1x1, plan1x2, plan1x3, y_avg_arr, dispersion, m):
    # Оцінка значимості коефіцієнтів регресії згідно критерію Стьюдента
    print('Оцінка значимості коефіцієнтів регресії згідно критерію Стьюдента')
    s2b = sum(dispersion) / N
    s2bs_avg = s2b / N * m
    sb = s2bs_avg ** 0.5

    beta_arr = [
        sum([y_avg_arr[i] * plan1x0[i] for i in range(N)]) / N,
        sum([y_avg_arr[i] * plan1x1[i] for i in range(N)]) / N,
        sum([y_avg_arr[i] * plan1x2[i] for i in range(N)]) / N,
        sum([y_avg_arr[i] * plan1x3[i] for i in range(N)]) / N,
        sum([y_avg_arr[i] * plan1x1[i] * plan1x2[i] for i in range(N)]) / N,
        sum([y_avg_arr[i] * plan1x1[i] * plan1x3[i] for i in range(N)]) / N,
        sum([y_avg_arr[i] * plan1x2[i] * plan1x3[i] for i in range(N)]) / N,
        sum([y_avg_arr[i] * plan1x1[i] * plan1x2[i] * plan1x3[i] for i in range(N)])
/ N,
        sum([y_avg_arr[i] * plan1x1[i] ** 2 for i in range(N)]) / N,
        sum([y_avg_arr[i] * plan1x2[i] ** 2 for i in range(N)]) / N,
        sum([y_avg_arr[i] * plan1x3[i] ** 2 for i in range(N)]) / N
    ]

    print('beta:', beta_arr)
    t_arr = [abs(i) / sb for i in beta_arr]
    print('t:', t_arr)

    # f3 = f1*f2 = 2*14 = 28
    f1 = m - 1
    f2 = N
    f3 = f1 * f2

    b_arr = []
    t_table = t.ppf(q=0.975, df=f3)
    print(f't table = {t_table}')
    count = 0
    for i in range(len(t_arr)):
        if t_arr[i] > t_table:
            b_arr.append(t_arr[i])
        else:
            print(f'Коефіцієнт b{i} приймаємо не значним')
            b_arr.append(0)
            count += 1
```

```
    if not count:
        print('Усі коефіцієнти рівняння значимі')

    return b_arr, s2b


def Fisher(b_arr, s2b, y_avg, y_res, m):
    # Критерій Фішера
    print('Перевірка адекватності моделі за критерієм Фішера')

    d = len([i for i in b_arr if i != 0])  # кількість значимих коефіцієнтів
    print(f'd = {d}')
    s2_ad = m * sum([(y_res[i] - y_avg[i]) ** 2 for i in range(N)]) / N - d
    fp = s2_ad / s2b
    print(f'Fp = {fp}')

    print(f'Ft = {3}')
    # Fт = 3
    if fp > 3:
        print('Рівняння регресії неадекватно оригіналу при рівні значимості 0.05')
    else:
        print('Рівняння регресії адекватно оригіналу при рівні значимості 0.05')


def main(m):
    # Кількість факторів
    k = 3

    x1 = [10, 60]
    x2 = [15, 50]
    x3 = [15, 20]

    # Величина зоряного плеча
    l = round(k ** 0.5, 2)

    # Матриця планування з нормованих значень
    plan1x0 = [1 for _ in range(N)]
    plan1x1 = [-1, -1, 1, 1, -1, -1, 1, 1, l, -l, 0, 0, 0, 0]
    plan1x2 = [-1, 1, -1, 1, -1, 1, -1, 1, 0, 0, l, -l, 0, 0]
    plan1x3 = [-1, 1, 1, -1, 1, -1, -1, 1, 0, 0, 0, 0, l, -l]
    print('x1:', plan1x1)
    print('x2:', plan1x2)
    print('x3:', plan1x3)
    print('-' * 100)

    # Матриця планування з натуралізованих значень
    plan2x1 = Naturalize(plan1x1, x1, 1)
    plan2x2 = Naturalize(plan1x2, x2, 2)
    plan2x3 = Naturalize(plan1x3, x3, 3)

    # Мультиплікативні значення факторів
    plan2x4 = [plan2x1[i] * plan2x2[i] for i in range(len(plan2x1))]
    plan2x5 = [plan2x1[i] * plan2x3[i] for i in range(len(plan2x1))]
    plan2x6 = [plan2x2[i] * plan2x3[i] for i in range(len(plan2x1))]
    plan2x7 = [plan2x1[i] * plan2x2[i] * plan2x3[i] for i in range(len(plan2x1))]

    # Квадратичні значення факторів
    plan2x8 = [plan2x1[i] ** 2 for i in range(len(plan2x1))]
    plan2x9 = [plan2x2[i] ** 2 for i in range(len(plan2x1))]
    plan2x10 = [plan2x3[i] ** 2 for i in range(len(plan2x1))]
```

```
    print(f'x1: {plan2x1}')
    print(f'x2: {plan2x2}')
    print(f'x3: {plan2x3}')
    print(f'x4: {plan2x4}')
    print(f'x5: {plan2x5}')
    print(f'x6: {plan2x6}')
    print(f'x7: {plan2x7}')
    print(f'x8: {plan2x8}')
    print(f'x9: {plan2x9}')
    print(f'x10: {plan2x10}')
    print()

    x_matrix = [plan2x1, plan2x2, plan2x3, plan2x4, plan2x5, plan2x6, plan2x7,
plan2x8, plan2x9, plan2x10]

    y_arr = [[y_func(x_matrix, i) + randint(0, 10) - 5 for i in range(N)] for _ in
range(m)]
    for i in range(len(y_arr)):
        print(f'y{i + 1}: {y_arr[i]}')

    y_avg = []
    for i in range(len(y_arr[0])):
        current_sum = 0
        for j in range(len(y_arr)):
            current_sum += y_arr[j][i]
        y_avg.append(current_sum / len(y_arr))
    print('y average:', y_avg)
    print('-' * 100)

    mx1 = sum(plan2x1) / len(plan2x1)
    mx2 = sum(plan2x2) / len(plan2x2)
    mx3 = sum(plan2x3) / len(plan2x3)
    mx4 = sum(plan2x4) / len(plan2x4)
    mx5 = sum(plan2x5) / len(plan2x5)
    mx6 = sum(plan2x6) / len(plan2x6)
    mx7 = sum(plan2x7) / len(plan2x7)
    mx8 = sum(plan2x8) / len(plan2x8)
    mx9 = sum(plan2x9) / len(plan2x9)
    mx10 = sum(plan2x10) / len(plan2x10)
    my = sum(y_avg) / len(y_avg)

    a1 = sum([y_avg[i] * plan2x1[i] for i in range(len(plan2x1))]) / len(plan2x1)
    a11 = mx8
    a12 = mx4
    a13 = mx5
    a14 = sum([plan2x1[i] * plan2x4[i] for i in range(len(plan2x1))]) / len(plan2x1)
    a15 = sum([plan2x1[i] * plan2x5[i] for i in range(len(plan2x1))]) / len(plan2x1)
    a16 = sum([plan2x1[i] * plan2x6[i] for i in range(len(plan2x1))]) / len(plan2x1)
    a17 = sum([plan2x1[i] * plan2x7[i] for i in range(len(plan2x1))]) / len(plan2x1)
    a18 = sum([plan2x1[i] * plan2x8[i] for i in range(len(plan2x1))]) / len(plan2x1)
    a19 = sum([plan2x1[i] * plan2x9[i] for i in range(len(plan2x1))]) / len(plan2x1)

    a2 = sum([y_avg[i] * plan2x2[i] for i in range(len(plan2x1))]) / len(plan2x2)
    a21 = a12
    a22 = mx9
    a23 = mx6
    a24 = sum([plan2x2[i] * plan2x4[i] for i in range(len(plan2x2))]) / len(plan2x2)
    a25 = sum([plan2x2[i] * plan2x5[i] for i in range(len(plan2x2))]) / len(plan2x2)
    a26 = sum([plan2x2[i] * plan2x6[i] for i in range(len(plan2x2))]) / len(plan2x2)
    a27 = sum([plan2x2[i] * plan2x7[i] for i in range(len(plan2x2))]) / len(plan2x2)
    a28 = sum([plan2x2[i] * plan2x8[i] for i in range(len(plan2x2))]) / len(plan2x2)
    a29 = sum([plan2x2[i] * plan2x9[i] for i in range(len(plan2x2))]) / len(plan2x2)
```

```
a3 = sum([y_avg[i] * plan2x3[i] for i in range(len(plan2x3))]) / len(plan2x3)
a31 = a13
a32 = a23
a33 = mx10
a34 = sum([plan2x3[i] * plan2x4[i] for i in range(len(plan2x3))]) / len(plan2x3)
a35 = sum([plan2x3[i] * plan2x5[i] for i in range(len(plan2x3))]) / len(plan2x3)
a36 = sum([plan2x3[i] * plan2x6[i] for i in range(len(plan2x3))]) / len(plan2x3)
a37 = sum([plan2x3[i] * plan2x7[i] for i in range(len(plan2x3))]) / len(plan2x3)
a38 = sum([plan2x3[i] * plan2x8[i] for i in range(len(plan2x3))]) / len(plan2x3)
a39 = sum([plan2x3[i] * plan2x9[i] for i in range(len(plan2x3))]) / len(plan2x3)

a4 = sum([y_avg[i] * plan2x4[i] for i in range(len(plan2x4))]) / len(plan2x4)
a41 = a14
a42 = a24
a43 = a34
a44 = sum([plan2x4[i] ** 2 for i in range(len(plan2x4))]) / len(plan2x4)
a45 = sum([plan2x4[i] * plan2x5[i] for i in range(len(plan2x4))]) / len(plan2x4)
a46 = sum([plan2x4[i] * plan2x6[i] for i in range(len(plan2x4))]) / len(plan2x4)
a47 = sum([plan2x4[i] * plan2x7[i] for i in range(len(plan2x4))]) / len(plan2x4)
a48 = sum([plan2x4[i] * plan2x8[i] for i in range(len(plan2x4))]) / len(plan2x4)
a49 = sum([plan2x4[i] * plan2x9[i] for i in range(len(plan2x4))]) / len(plan2x4)

a5 = sum([y_avg[i] * plan2x5[i] for i in range(len(plan2x5))]) / len(plan2x5)
a51 = a15
a52 = a25
a53 = a35
a54 = a45
a55 = sum([plan2x5[i] ** 2 for i in range(len(plan2x5))]) / len(plan2x5)
a56 = sum([plan2x5[i] * plan2x6[i] for i in range(len(plan2x5))]) / len(plan2x5)
a57 = sum([plan2x5[i] * plan2x7[i] for i in range(len(plan2x5))]) / len(plan2x5)
a58 = sum([plan2x5[i] * plan2x8[i] for i in range(len(plan2x5))]) / len(plan2x5)
a59 = sum([plan2x5[i] * plan2x9[i] for i in range(len(plan2x5))]) / len(plan2x5)

a6 = sum([y_avg[i] * plan2x6[i] for i in range(len(plan2x6))]) / len(plan2x6)
a61 = a16
a62 = a26
a63 = a36
a64 = a46
a65 = a56
a66 = sum([plan2x6[i] ** 2 for i in range(len(plan2x6))]) / len(plan2x6)
a67 = sum([plan2x6[i] * plan2x7[i] for i in range(len(plan2x6))]) / len(plan2x6)
a68 = sum([plan2x6[i] * plan2x8[i] for i in range(len(plan2x6))]) / len(plan2x6)
a69 = sum([plan2x6[i] * plan2x9[i] for i in range(len(plan2x6))]) / len(plan2x6)

a7 = sum([y_avg[i] * plan2x7[i] for i in range(len(plan2x7))]) / len(plan2x7)
a71 = a17
a72 = a27
a73 = a37
a74 = a47
a75 = a57
a76 = a67
a77 = sum([plan2x7[i] ** 2 for i in range(len(plan2x7))]) / len(plan2x7)
a78 = sum([plan2x7[i] * plan2x8[i] for i in range(len(plan2x7))]) / len(plan2x7)
a79 = sum([plan2x7[i] * plan2x9[i] for i in range(len(plan2x7))]) / len(plan2x7)

a8 = sum([y_avg[i] * plan2x8[i] for i in range(len(plan2x8))]) / len(plan2x8)
a81 = a18
a82 = a28
a83 = a38
a84 = a48
a85 = a58
```

```
    a86 = a68
    a87 = a78
    a88 = sum([plan2x8[i] ** 2 for i in range(len(plan2x8))]) / len(plan2x8)
    a89 = sum([plan2x8[i] * plan2x9[i] for i in range(len(plan2x8))]) / len(plan2x8)

    a9 = sum([y_avg[i] * plan2x9[i] for i in range(len(plan2x9))]) / len(plan2x9)
    a91 = a19
    a92 = a29
    a93 = a39
    a94 = a49
    a95 = a59
    a96 = a69
    a97 = a79
    a98 = a89
    a99 = sum([plan2x9[i] ** 2 for i in range(len(plan2x9))]) / len(plan2x9)

    a10 = sum([y_avg[i] * plan2x10[i] for i in range(len(plan2x10))]) / len(plan2x10)
    a101 = sum([plan2x10[i] * plan2x1[i] for i in range(len(plan2x10))]) /
len(plan2x10)
    a102 = sum([plan2x10[i] * plan2x2[i] for i in range(len(plan2x10))]) /
len(plan2x10)
    a103 = sum([plan2x10[i] * plan2x3[i] for i in range(len(plan2x10))]) /
len(plan2x10)
    a104 = sum([plan2x10[i] * plan2x4[i] for i in range(len(plan2x10))]) /
len(plan2x10)
    a105 = sum([plan2x10[i] * plan2x5[i] for i in range(len(plan2x10))]) /
len(plan2x10)
    a106 = sum([plan2x10[i] * plan2x6[i] for i in range(len(plan2x10))]) /
len(plan2x10)
    a107 = sum([plan2x10[i] * plan2x7[i] for i in range(len(plan2x10))]) /
len(plan2x10)
    a108 = sum([plan2x10[i] * plan2x8[i] for i in range(len(plan2x10))]) /
len(plan2x10)
    a109 = sum([plan2x10[i] * plan2x9[i] for i in range(len(plan2x10))]) /
len(plan2x10)
    a1010 = sum([plan2x10[i] ** 2 for i in range(len(plan2x10))]) / len(plan2x10)

    main_matrix = [[1, mx1, mx2, mx3, mx4, mx5, mx6, mx7, mx8, mx9, mx10],
                   [mx1, a11, a21, a31, a41, a51, a61, a71, a81, a91, a101],
                   [mx2, a12, a22, a32, a42, a52, a62, a72, a82, a92, a102],
                   [mx3, a13, a23, a33, a43, a53, a63, a73, a83, a93, a103],
                   [mx4, a14, a24, a34, a44, a54, a64, a74, a84, a94, a104],
                   [mx5, a15, a25, a35, a45, a55, a65, a75, a85, a95, a105],
                   [mx6, a16, a26, a36, a46, a56, a66, a76, a86, a96, a106],
                   [mx7, a17, a27, a37, a47, a57, a67, a77, a87, a97, a107],
                   [mx8, a18, a28, a38, a48, a58, a68, a78, a88, a98, a108],
                   [mx9, a19, a29, a39, a49, a59, a69, a79, a89, a99, a109],
                   [mx10, a101, a102, a103, a104, a105, a106, a107, a108, a109,
a1010]]

    column_to_change = [my, a1, a2, a3, a4, a5, a6, a7, a8, a9, a10]
    main_determinant = det(main_matrix)

    matrices = []
    for i in range(len(main_matrix[0])):
        new_matrix = deepcopy(main_matrix)
        for j in range(len(main_matrix)):
            new_matrix[j][i] = column_to_change[j]
        matrices.append(new_matrix)

    print('Знаходження коефіцієнтів рівняння регресії')
    b_list = []
```

```
    for i in range(len(matrices)):
        b_list.append(det(matrices[i]) / main_determinant)
    print(f'b: {b_list}')

    print('Підстановка отриманих коефіцієнтів у рівняння регресії')
    y_list = []
    for i in range(len(plan2x1)):
        y = b_list[0] + b_list[1] * plan2x1[i] + b_list[2] * plan2x2[i] + b_list[3] *
plan2x3[i] + \
            b_list[4] * plan2x4[i] + b_list[5] * plan2x5[i] + b_list[6] * plan2x6[i]
+ b_list[7] * plan2x7[i] + \
            b_list[8] * plan2x8[i] + b_list[9] * plan2x9[i] + b_list[10] *
plan2x10[i]
        y_list.append(y)
        print(f'y = {y}; y avg = {y_avg[i]}')
    print('-' * 100)

    dispersion = Cochran(y_arr, y_avg, m)
    print('-' * 100)
    if dispersion:
        t_arr, s2b = Students(plan1x0, plan1x1, plan1x2, plan1x3, y_avg, dispersion,
m)

        b_arr = []
        for i in range(len(b_list)):
            b = b_list[i] if t_arr[i] != 0 else 0
            b_arr.append(b)
        print('-' * 100)

        print('Підстановка коефіцієнтів у спрощене рівняння регресії')
        y_res = []
        for i in range(N):
            y = b_arr[0] + b_arr[1] * plan2x1[i] + b_arr[2] * plan2x2[i] + b_arr[3] *
plan2x3[i] + \
                b_arr[4] * plan2x4[i] + b_arr[5] * plan2x5[i] + b_arr[6] * plan2x6[i]
+ b_arr[7] * plan2x7[i] +\
                b_arr[8] * plan2x8[i] + b_arr[9] * plan2x9[i] + b_arr[10] *
plan2x10[i]
            print(f'y = {y}; y avg = {y_avg[i]}')
            y_res.append(y)

        print('-' * 100)
        Fisher(b_arr, s2b, y_avg, y_res, m)
    else:
        print('Збільшуємо кількість дослідів')
        main(m+1)
        exit()


if __name__ == '__main__':
    N = 14
    main(m=3)
```

Результат виконання програми: