

Міністерство освіти та науки України  
Національний технічний університет України  
«Київський політехнічний інститут ім. Ігоря Сікорського»  
Факультет інформатики і обчислювальної техніки  
Кафедра обчислювальної техніки

#### **ЛАБОРАТОРНА РОБОТА №4**

з дисципліни «Методи оптимізації та планування експерименту» на тему:  
«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ  
ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ ЕФЕКТУ  
ВЗАЄМОДІЇ»

Виконав:  
студент II курсу ФІОТ  
групи ІВ-81 :  
Бухтій О. В.  
Перевірив:  
Регіда П. Г.

Київ-2020

## ЛАБОРАТОРНА РОБОТА №4.

**Мета:** провести повний трьохфакторний експеримент. Знайти рівняння регресії адекватне об'єкту.

### Завдання

|     | X <sub>1</sub> |     | X <sub>2</sub> |     | X <sub>3</sub> |     |
|-----|----------------|-----|----------------|-----|----------------|-----|
|     | min            | max | min            | max | min            | max |
| 107 | 10             | 40  | 25             | 45  | 40             | 45  |

$$Y_{\max} = 243.34$$

$$Y_{\min} = 225$$

### Лістинг програми

```
from prettytable import PrettyTable as prtt
from scipy.stats import f, t
import numpy as np
import functools as ft
import itertools as itr
from math import sqrt

# ~ Дано-----
x1_min = 10
x1_max = 40

x2_min = 25
x2_max = 45

x3_min = 40
x3_max = 45

x_max_list = [x1_max, x2_max, x3_max]
x_min_list = [x1_min, x2_min, x3_min]

k = len(x_max_list)
m = 3
p = 0.95
with_interactions = False

y_max = 200 + sum(x_max_list)/len(x_max_list)
y_min = 200 + sum(x_min_list)/len(x_min_list)
# ~ -----

def fisher_critical(prob, f4, f3):
    return f.ppf(p, f4, f3)

def student_critical(q, f3):
    return t.ppf((1 + (1 - q)) / 2, f3)
```

```

def cochran_critical(q, f1, f2):
    return 1 / (1 + (f2 - 1) / f.ppf(1 - q/f2, f1, (f2 - 1)*f1) )

def print_matrix(x_max_list, k, x_matr, y_matr, y_avg, y_disp):
    adequate_table = prtt()
    group = ["x"+str(n+1) for n in range(len(x_max_list))]
    if with_interractions :
        group_quantity = k+1
    else:
        group_quantity = 1
    header = []
    for j in range(group_quantity):
        header.extend(list(itr.combinations(group,j+1)))
    for i in range(len(y_matr[0])):
        header.append('y'+str(i+1))
    header.append('y_avg')
    header.append('s2{y}') # ~ variance or disp
    # ~ add squares
    final_matrix = np.hstack((x_matr,y_matr,[[round(avg,2)] for avg in y_avg],
    [[round(disp,2)] for disp in y_disp]))
    header = list([ft.reduce(lambda x,y : x+y, i) for i in header])
    adequate_table.field_names = ['x0'] + header
    for row_number in range(len(final_matrix)):
        adequate_table.add_row(final_matrix[row_number])
    print(adequate_table)

def print_equation(x_max_list, k, beta_list, ):
    group = ["x"+str(n+1) for n in range(len(x_max_list))]
    if with_interractions:
        group_quantity = k+1
        header = []
        for j in range(group_quantity):
            header.extend(list(itr.combinations(group,j+1)))
        x_names = list([ft.reduce(lambda x,y : x+y, i) for i in header])
        form = '{:.4f} + '+'*{:.4f} + '.join(x_names)+'*{:.4f} = y'
    else:
        group_quantity = 1
        header = []
        for j in range(group_quantity):
            header.extend(list(itr.combinations(group,j+1)))
        x_names = list([ft.reduce(lambda x,y : x+y, i) for i in header])
        form = '{:.4f} + '+'*{:.4f} + '.join(x_names)+'*{:.4f} = y'
    print(form.format(*beta_list))

stop=0
adequacy = False
while adequacy == False :
    # ~ Normalized matrix
    # ~ factors
    items = [(-1,1) for i in range(k)]
    f_matrix = list(itr.product(*items))
    x0_vector = [[1] for i in range(len(f_matrix))]
    # ~ add zor t
    # ~ with interractions-----
    if with_interractions :
        interractions_matrix1 = []
        group_quantity = k - 1
        for i in f_matrix:
            comb_list = []
            for j in range(group_quantity):
                comb_list.extend(list(itr.combinations(i,j+2)))
            comb_values = [np.prod(k) for k in comb_list]

```

```

        interactions_matrix1.append(comb_values)
# ~ -----
if with_interactions :
    norm_matrix = np.hstack((x0_vector, f_matrix,
interactions_matrix1))
else:
    norm_matrix = np.hstack((x0_vector, f_matrix))

# ~ Naturalized matrix
col_list = []
for i in range(len(f_matrix[0])):
    col = [row[i] for row in f_matrix]
    col = list(map(lambda x : x_max_list[i] if x==1 else x_min_list[i],
col))
    col_list.append(col)
nf_matrix = list(zip(*col_list)) # ~ naturalized factors
# ~ add zor t

x0_vector = [[1] for i in range(len(nf_matrix))]
# ~ with interactions-----
if with_interactions :
    print("here")
    interactions_matrix2 = []
    group_quantity = k - 1
    for i in nf_matrix:
        comb_list = []
        for j in range(group_quantity):
            comb_list.extend(list(itertools.combinations(i,j+2)))
        comb_values = [np.prod(k) for k in comb_list]
        interactions_matrix2.append(comb_values)
# ~ -----
if with_interactions :
    natur_matrix = np.hstack((x0_vector, nf_matrix,
interactions_matrix2))
else:
    natur_matrix = np.hstack((x0_vector, nf_matrix))
# ~
-----

escape = False
while escape == False:
    y_matr = np.random.randint(y_min, y_max, (len(natur_matrix),m))
    y_avg = [sum(i)/len(i) for i in y_matr]
    y_disp = [] # ~ i.e. variance
    for i in range(len(natur_matrix)):
        tmp_disp = 0
        for j in range(m):
            tmp_disp += ((y_matr[i][j] - y_avg[i]) ** 2) / m
        y_disp.append(tmp_disp)
    f1 = m - 1
    f2 = len(natur_matrix)
    q = 1 - p
# ~ Cochran test
    Gp = max(y_disp) / sum(y_disp)
    Gt = cochrans_critical(q, f1, f2)
    # ~ print(Gt, Gp)
    if Gt > Gp:
        escape = True
        form = 'Cochran`s test passed with significance level {:.4f} :
Gt > Gp'
    else:
        m += 1
        form = 'Cochran`s test failed with significance level {:.4f} :

```

```

Gt < Gp'
    print(form.format(q))
    print('Gt = {}\nGp = {}'.format(Gt, Gp))
# ~
-----
    if with_interactions :
        beta_list_norm = list(np.linalg.solve(norm_matrix, y_avg))
        beta_list_natur = list(np.linalg.solve(natur_matrix, y_avg))
    else:
        beta_list_norm = list(np.linalg.solve(norm_matrix[1:len(x_max_list)
+2], y_avg[1:len(x_max_list)+2]))
        beta_list_natur =
list(np.linalg.solve(natur_matrix[1:len(x_max_list)+2], y_avg[1:len(x_max_list)
+2]))

    print_matrix(x_max_list, k, norm_matrix, y_matr, y_avg, y_disp)
    print('Equation with normalized coefficients : ')
    print_equation(x_max_list, k, beta_list_norm)
    print_matrix(x_max_list, k, natur_matrix, y_matr, y_avg, y_disp)
    print('Equation with naturalized coefficients : ')
    print_equation(x_max_list, k, beta_list_natur)

# ~
-----

    print('\nStudent`s test')
    N = len(beta_list_norm)
    f3 = f1 * f2
    S2b = sum(y_disp)**2 / (N * N * m)
    Sb = sqrt(S2b)
    betast_list = []
    for i in range(len(norm_matrix[0])):
        x_list_tmp = norm_matrix[:,i]
        beta_tmp = ((sum([np.prod(i) for i in list(zip(x_list_tmp,
y_avg))])) / N)
        betast_list.append(beta_tmp)

    T_list = [abs(beta)/Sb for beta in betast_list]
    T = student_critical(q, f3)
    print('T = '+str(T)+ '\nT_list = '+str(list(map(lambda x : round(x, 2),
T_list))))
    for i in range(len(T_list)):
        if T_list[i] < T :
            T_list[i] = 0
            beta_list_natur[i] = 0
    print('Fixed beta_list = '+ str(list(map(lambda x :round(x, 2),
beta_list_natur))))
    print('Equation without insignificant coefficients : ')
    print_equation(x_max_list, k, beta_list_natur)

    print("\nFisher`s test")
    equation_y_list = []
    for i in range(len(natur_matrix[0])):
        x_list_tmp = natur_matrix[i]
        y_tmp = sum([np.prod(i) for i in list(zip(x_list_tmp,
beta_list_natur))])
        equation_y_list.append(y_tmp)

    beta_list_natur = list(filter(lambda i : (i != 0), beta_list_natur))
    d = len(beta_list_natur)
    f4 = N - d

```

```

S2ad = m * (sum([(i[0]-i[1])**2 for i in list(zip(equation_y_list,
y_avg))])) / f4
Fp = S2ad / S2b
Ft = fisher_critical(p, f4, f3)
print('Fp = ' + str(Fp) + "\nFt = " + str(Ft))
if Fp > Ft:
    print("      The regression equation is inadequate at the
significance level {:.2f}".format(q))
    with_interactions = True
else:
    print("      The regression equation is adequate at the significance
level {:.2f}".format(q))
    adequacy = True

```

## Результат виконання

```

Terminal -
File Edit View Terminal Tabs Help
Cochran's test passed with significance level 0.0500 : Gt > Gp
Gt = 0.5156874570365015
Gp = 0.3023622047244095
+-----+
| x0 | x1 | x2 | x3 | x1x2 | x1x3 | x2x3 | x1x2x3 | y1 | y2 | y3 | y_avg | s2{y} |
+-----+
| 1.0 | -1.0 | -1.0 | -1.0 | 1.0 | 1.0 | 1.0 | -1.0 | 232.0 | 236.0 | 235.0 | 234.33 | 2.89 |
| 1.0 | -1.0 | -1.0 | 1.0 | 1.0 | -1.0 | -1.0 | 1.0 | 241.0 | 225.0 | 233.0 | 233.0 | 42.67 |
| 1.0 | -1.0 | 1.0 | -1.0 | -1.0 | 1.0 | -1.0 | 1.0 | 238.0 | 233.0 | 226.0 | 232.33 | 24.22 |
| 1.0 | -1.0 | 1.0 | 1.0 | -1.0 | -1.0 | 1.0 | -1.0 | 232.0 | 234.0 | 232.0 | 232.67 | 0.89 |
| 1.0 | 1.0 | -1.0 | -1.0 | -1.0 | -1.0 | 1.0 | 1.0 | 242.0 | 233.0 | 240.0 | 238.33 | 14.89 |
| 1.0 | 1.0 | -1.0 | 1.0 | -1.0 | 1.0 | -1.0 | -1.0 | 242.0 | 236.0 | 230.0 | 236.0 | 24.0 |
| 1.0 | 1.0 | 1.0 | -1.0 | 1.0 | -1.0 | -1.0 | -1.0 | 239.0 | 234.0 | 226.0 | 233.0 | 28.67 |
| 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 238.0 | 237.0 | 241.0 | 238.67 | 2.89 |
+-----+
Equation with normalized coefficients :
234.7917 + x1*1.7083 + x2*-0.6250 + x3*0.2917 + x1x2*-0.0417 + x1x3*0.5417 + x2x3*1.2083 + x1x2x3*0.7917 = y
+-----+
| x0 | x1 | x2 | x3 | x1x2 | x1x3 | x2x3 | x1x2x3 | y1 | y2 | y3 | y_avg | s2{y} |
+-----+
| 1.0 | 10.0 | 25.0 | 40.0 | 250.0 | 400.0 | 1000.0 | 10000.0 | 232.0 | 236.0 | 235.0 | 234.33 | 2.89 |
| 1.0 | 10.0 | 25.0 | 45.0 | 250.0 | 450.0 | 1125.0 | 11250.0 | 241.0 | 225.0 | 233.0 | 233.0 | 42.67 |
| 1.0 | 10.0 | 45.0 | 40.0 | 450.0 | 400.0 | 1800.0 | 18000.0 | 238.0 | 233.0 | 226.0 | 232.33 | 24.22 |
| 1.0 | 10.0 | 45.0 | 45.0 | 450.0 | 450.0 | 2025.0 | 20250.0 | 232.0 | 234.0 | 232.0 | 232.67 | 0.89 |
| 1.0 | 40.0 | 25.0 | 40.0 | 1000.0 | 1600.0 | 1000.0 | 40000.0 | 242.0 | 233.0 | 240.0 | 238.33 | 14.89 |
| 1.0 | 40.0 | 25.0 | 45.0 | 1000.0 | 1800.0 | 1125.0 | 45000.0 | 242.0 | 236.0 | 230.0 | 236.0 | 24.0 |
| 1.0 | 40.0 | 45.0 | 40.0 | 1800.0 | 1600.0 | 1800.0 | 72000.0 | 239.0 | 234.0 | 226.0 | 233.0 | 28.67 |
| 1.0 | 40.0 | 45.0 | 45.0 | 1800.0 | 1800.0 | 2025.0 | 81000.0 | 238.0 | 237.0 | 241.0 | 238.67 | 2.89 |
+-----+
Equation with naturalized coefficients :
237.6667 + x1*2.6500 + x2*0.1333 + x3*-0.0889 + x1x2*-0.0900 + x1x3*-0.0594 + x2x3*-0.0044 + x1x2x3*0.0021 = y
Student's test
T = 2.1199052992210112
T list = [23.06, 0.17, 0.06, 0.03, 0.0, 0.05, 0.12, 0.08]
Fixed beta list = [237.67, 0, 0, 0, 0, 0, 0, 0]
Equation without insignificant coefficients :
237.6667 + x1*0.0000 + x2*0.0000 + x3*0.0000 + x1x2*0.0000 + x1x3*0.0000 + x2x3*0.0000 + x1x2x3*0.0000 = y
Fisher test
Fp = 0.46420591126895266
Ft = 2.6571966002210865
The regression equation is adequate at the significance level 0.05

(program exited with code: 0)
Press return to continue

```

## **Висновки**

Підчас виконання лабораторної роботи було реалізовано завдання . Отримані результати збігаються , отже, експеримент було поставлено правильно.