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

Лабораторна робота №3

«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ
ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ.»

Виконав:
студент групи ІВ-81
Зікратий Д. О.
Залікова книжка № 8114
Перевірив:
Регіда П. Г.

Київ – 2020

Варіант завдання

№варианта	X_1		X_2		X_3	
	min	max	min	max	min	max
113	-15	30	5	40	5	25

Код програми

```
import numpy as np
from scipy.stats import f,t
from random import randrange

#const
x1_min= -15
x1_max= 30
x2_min= 5
x2_max= 40
x3_min= 5
x3_max= 25
x_average_max = (x1_max + x2_max + x3_max)/3
x_average_min = (x1_min + x2_min + x3_min)/3
y_max = 200 + x_average_max
y_min = 200 + x_average_min

def main(m=3,N=4,p=0.95):
    matrix_y = np.random.randint(y_min,y_max, size=(N,m))#[[randrange(y_min,y_max)
for _ in range(m)] for _ in range(N)]
    normalized_matrix_x = np.array([[x1_min, x2_min, x3_min], [x1_min, x2_max,
x3_max], [x1_max, x2_min, x3_max], [x1_max, x2_max, x3_min]])
    matrix_x = [[1, -1, -1, -1], [1, -1, 1, 1], [1, 1, -1, 1], [1, 1, 1, -1]]
    print("Matrix of Y")
    print(matrix_y)
    print("Matrix of normalized X")
    print(normalized_matrix_x)
    y_average_list = [sum(y)/len(y) for y in matrix_y ]
    mx_list=[_/N for _ in np.sum(normalized_matrix_x, axis=0)]
    mx1,mx2,mx3=mx_list
    my=sum(y_average_list)/len(y_average_list)

    a1 = sum([normalized_matrix_x[_][0] * y_average_list[_] for _ in range(N)]) / N
    a2 = sum([normalized_matrix_x[_][1] * y_average_list[_] for _ in range(N)]) / N
    a3 = sum([normalized_matrix_x[_][2] * y_average_list[_] for _ in range(N)]) / N

    a11 = sum([normalized_matrix_x[_][0] ** 2 for _ in range(N)]) / N
    a22 = sum([normalized_matrix_x[_][1] ** 2 for _ in range(N)]) / N
    a33 = sum([normalized_matrix_x[_][2] ** 2 for _ in range(N)]) / N

    a12 = sum([normalized_matrix_x[_][0] * normalized_matrix_x[_][1] for _ in
range(N)]) / N
    a13 = sum([normalized_matrix_x[_][0] * normalized_matrix_x[_][2] for _ in
range(N)]) / N
    a32 = sum([normalized_matrix_x[_][2] * normalized_matrix_x[_][1] for _ in
range(N)]) / N

    a21=a12
    a23=a32
    a31=a13
    det_denominator =
```

```
np.linalg.det([[1,mx1,mx2,mx3],[mx1,a11,a12,a13],[mx2,a12,a22,a32],[mx3,a13,a23,a33]]
)
    b0 =
np.linalg.det([[my,mx1,mx2,mx3],[a1,a11,a12,a13],[a2,a12,a22,a32],[a3,a13,a23,a33]])
/ det_denominator
    b1 =
np.linalg.det([[1,my,mx2,mx3],[mx1,a1,a12,a13],[mx2,a2,a22,a32],[mx3,a3,a23,a33]]) /
det_denominator
    b2 =
np.linalg.det([[1,mx1,my,mx3],[mx1,a11,a1,a13],[mx2,a12,a2,a32],[mx3,a13,a3,a33]]) /
det_denominator
    b3 =
np.linalg.det([[1,mx1,mx2,my],[mx1,a11,a12,a1],[mx2,a12,a22,a2],[mx3,a13,a23,a3]]) /
det_denominator

    print(f"Regression y = {round(b0,2)} + {round(b1,2)}*X1 + {round(b2,2)}*X2 +
{round(b3,2)}*X3")
    y = [b0 + normalized_matrix_x[_][0]*b1 + normalized_matrix_x[_][1]*b2 +
normalized_matrix_x[_][2]*b3 for _ in range(N)]
    print(y)
    print(y_average_list)

    dispersion1 = sum([(_ - y_average_list[0])**2 for _ in matrix_y[0]]) / m
    dispersion2 = sum([(_ - y_average_list[1])**2 for _ in matrix_y[1]]) / m
    dispersion3 = sum([(_ - y_average_list[2])**2 for _ in matrix_y[2]]) / m
    dispersion4 = sum([(_ - y_average_list[3])**2 for _ in matrix_y[3]]) / m
    dispersion_list = [dispersion1,dispersion2,dispersion3,dispersion4]

    Gp = max(dispersion_list)/sum(dispersion_list)
    f1 = m-1,
    f2 = N

    Gt=(1 / (1 + (f2 - 1) / f.ppf(1 - (1 - p) / f2, f1, (f2 - 1) * f1)))[0]

    if Gp < Gt:
        print(f"Homogeneous dispersion with {p} probability:\t{round(Gp,3)} < {Gt}")
    else:
        print(f"Inhomogeneous dispersion with {p} probability:\t{round(Gp,3)} >
{Gt}")
        return False


    s_b = sum(dispersion_list) / N
    s2_b_s = s_b / (N * m)
    s_b_s = s2_b_s**(0.5)

    beta0 = sum([matrix_x[_][0] * y_average_list[_] for _ in range(len(matrix_x))]) /
N
    beta1 = sum([matrix_x[_][1] * y_average_list[_] for _ in range(len(matrix_x))]) /
N
    beta2 = sum([matrix_x[_][2] * y_average_list[_] for _ in range(len(matrix_x))]) /
N
    beta3 = sum([matrix_x[_][3] * y_average_list[_] for _ in range(len(matrix_x))]) /
N

    t0 = abs(beta0) / s_b_s
    t1 = abs(beta1) / s_b_s
    t2 = abs(beta2) / s_b_s
    t3 = abs(beta3) / s_b_s
    f3 = f1 * f2
    t_table = t.ppf((1 + p) / 2, f3)[0]
    b00 = b0 if t0 > t_table else 0
```

```
    b11 = b1 if t1 > t_table else 0
    b22 = b2 if t2 > t_table else 0
    b33 = b3 if t3 > t_table else 0
    b_list = np.array([b00, b11, b22, b33])
    print(f"Regression y = {round(b00, 2)} + {round(b11, 2)}*X1 + {round(b22, 2)}*X2
+ {round(b33, 2)}*X3")


    ch11 = b00 + b11 * normalized_matrix_x[0][0] + b22 * normalized_matrix_x[0][1] +
b33 * normalized_matrix_x[0][2]
    ch22 = b00 + b11 * normalized_matrix_x[1][0] + b22 * normalized_matrix_x[1][1] +
b33 * normalized_matrix_x[1][2]
    ch33 = b00 + b11 * normalized_matrix_x[2][0] + b22 * normalized_matrix_x[2][1] +
b33 * normalized_matrix_x[2][2]
    ch44 = b00 + b11 * normalized_matrix_x[3][0] + b22 * normalized_matrix_x[3][1] +
b33 * normalized_matrix_x[3][2]
    ch_list = [ch11, ch22, ch33, ch44]


    d = len(b_list[np.array(b_list) != 0])
    f4 = N - d

    s2_ad = m / f4 * sum([(ch_list[_] - y_average_list[_]) ** 2 for _ in
range(len(y_average_list))])
    fp = s2_ad / s2_b_s
    ft = f.ppf(p, f4, f3)[0]

    if fp > ft:
        print(f"Equation is not adequate to the original with probability -
{p}:\t{round(fp,3)} > {round(ft,3)}")
    else:
        print(f"Equation is adequate to the original with probability -
{p}:\t{round(fp,3)} < {round(ft,3)}")
    return True



if __name__=="__main__":
    m=3
    result = False
    while not result:
        result = main(m=m)
        print(f"Experiment done with m ={m}")
        if not result:
            m += 1
```

Результати

```
Matrix of Y
[[226 230 200]
 [220 216 226]
 [230 217 224]
 [213 222 203]]
Matrix of normalized X
[[-15   5   5]
 [-15  40  25]
 [ 30   5  25]
 [ 30  40   5]]
Regression y = 217.18 + -0.03*X1 + -0.13*X2 + 0.32*X3
Homogeneous dispersion with 0.95 probability:   0.627 < 2500
Regression y = 217.18 + 0*X1 + 0*X2 + 0*X3
Equation is adequate to the original with probability - 0.95:   13.054 < 19.164
Experiment done with m =3
```