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 IВ-81

Зікратий Д. О.

Залікова книжка № 8114

Перевірив:

[Регіда П. Г.](http://rozklad.kpi.ua/Schedules/ViewSchedule.aspx?v=3616fe25-c15f-4d3e-986b-deb3928e21b8)

Київ – 2020

Варіант завдання





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

