Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Методи оптимізації та планування експерименту

Лабораторна робота №2

“ПРОВЕДЕННЯ ДВОФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ”

Виконав:

студент групи ІВ-83

Чорноморець В.О.

Перевірив:

ас. Регіда П.Г.

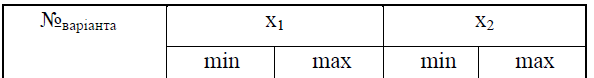
Київ

2020 р.

**Мета:** провести двофакторний експеримент, перевірити однорідність дисперсії за критерієм Романовського, отримати коефіцієнти рівняння регресії, провести натуралізацію рівняння регресії.

Номер у списку: 21

Варіант завдання: 321

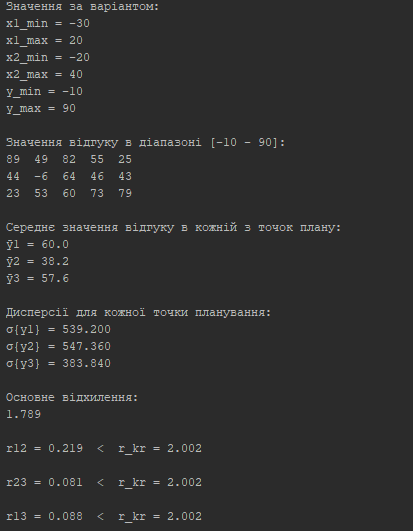


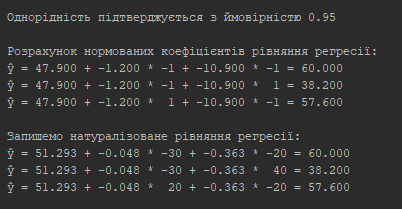


Код програми

from random import randint  
from math import sqrt  
import sys  
# Значення за варіантом  
n\_variant = 21  
y\_min = (20 - n\_variant) \* 10  
y\_max = (30 - n\_variant) \* 10  
x1\_min = -30  
x1\_max = 20  
x2\_min = -20  
x2\_max = 40  
p = 0.95 #дочірня ймовірність  
m = 5  
print('Значення за варіантом:')  
print(f'{x1\_min = }')  
print(f'{x1\_max = }')  
print(f'{x2\_min = }')  
print(f'{x2\_max = }')  
print(f'{y\_min = }')  
print(f'{y\_max = }')  
  
def get\_r\_kr(m):  
 if p == 0.99:  
 table\_values = {2: 1.73, 6: 2.16, 8: 2.43, 10: 2.62, 12: 2.75, 15: 2.9, 20: 3.08}  
 elif p == 0.98:  
 table\_values = {2: 1.72, 6: 2.13, 8: 2.37, 10: 2.54, 12: 2.66, 15: 2.8, 20: 2.96}  
 elif p == 0.95:  
 table\_values = {2: 1.71, 6: 2.10, 8: 2.27, 10: 2.41, 12: 2.52, 15: 2.64, 20: 2.78}  
 elif p == 0.9:  
 table\_values = {2: 1.69, 6: 2, 8: 2.17, 10: 2.29, 12: 2.39, 15: 2.49, 20: 2.62}  
 else:  
 print("Введіть значення довірчої ймовірності з таблиці (0.99, 0.98, 0.95 або 0.9).")  
 sys.exit()  
 for i in range(len(table\_values.keys())):  
 if m == list(table\_values.keys())[i]:  
 return list(table\_values.values())[i]  
 if m > list(table\_values.keys())[i]:  
 less\_than\_m\_key = list(table\_values.keys())[i]  
 less\_than\_m = list(table\_values.values())[i]  
 more\_than\_m\_key = list(table\_values.keys())[i + 1]  
 more\_than\_m = list(table\_values.values())[i + 1]  
 return less\_than\_m + (more\_than\_m - less\_than\_m) \* (m - less\_than\_m\_key) / (  
 more\_than\_m\_key - less\_than\_m\_key)  
  
  
def determinant(matrix):  
 return matrix[0][0] \* matrix[1][1] \* matrix[2][2] + matrix[0][1] \* matrix[1][2] \* matrix[2][0] + matrix[0][2] \* \  
 matrix[1][0] \* matrix[2][1] - matrix[0][2] \* matrix[1][1] \* matrix[2][0] - matrix[0][1] \* matrix[1][0] \* \  
 matrix[2][2] - matrix[0][0] \* matrix[1][2] \* matrix[2][1]  
  
  
def main():  
 global m  
 response\_list1 = [randint(y\_min, y\_max) for i in range(m)]  
 response\_list2 = [randint(y\_min, y\_max) for j in range(m)]  
 response\_list3 = [randint(y\_min, y\_max) for k in range(m)]  
  
 average1 = sum(response\_list1) / len(response\_list1)  
 average2 = sum(response\_list2) / len(response\_list2)  
 average3 = sum(response\_list3) / len(response\_list3)  
  
 dispersion1 = sum((i - average1) \*\* 2 for i in response\_list1) / len(response\_list1)  
 dispersion2 = sum((i - average2) \*\* 2 for i in response\_list2) / len(response\_list2)  
 dispersion3 = sum((i - average3) \*\* 2 for i in response\_list3) / len(response\_list3)  
  
 major\_deviation = sqrt((4 \* m - 4) / (m \* m - 4 \* m))  
  
 f12 = dispersion1 / dispersion2 if dispersion1 >= dispersion2 else dispersion2 / dispersion1  
 f23 = dispersion2 / dispersion3 if dispersion2 >= dispersion3 else dispersion3 / dispersion2  
 f13 = dispersion1 / dispersion3 if dispersion1 >= dispersion3 else dispersion3 / dispersion1  
  
 t12 = (m - 2) / m \* f12  
 t23 = (m - 2) / m \* f23  
 t13 = (m - 2) / m \* f13  
  
 r12 = abs(t12 - 1) / major\_deviation  
 r23 = abs(t23 - 1) / major\_deviation  
 r13 = abs(t13 - 1) / major\_deviation  
  
 r\_kr = get\_r\_kr(m)  
  
 print(f'\nЗначення відгуку в діапазоні [{y\_min} - {y\_max}]:')  
 print(\*response\_list1, sep='\t')  
 print(\*response\_list2, sep='\t')  
 print(\*response\_list3, sep='\t')  
 print('\nСереднє значення відгуку в кожній з точок плану:')  
 print("ȳ1 = " + str(average1))  
 print("ȳ2 = " + str(average2))  
 print("ȳ3 = " + str(average3))  
 print('\nДисперсії для кожної точки планування:')  
 print("σ{y1} = " + "{:.3f}".format(dispersion1))  
 print("σ{y2} = " + "{:.3f}".format(dispersion2))  
 print("σ{y3} = " + "{:.3f}".format(dispersion3))  
 print('\nОсновне відхилення:')  
 print(f'{major\_deviation:.3f}')  
 print(f'\n{r12 = :.3f} ', '<' if r12 < r\_kr else '>', f' {r\_kr = :.3f}')  
 print(f'\n{r23 = :.3f} ', '<' if r23 < r\_kr else '>', f' {r\_kr = :.3f}')  
 print(f'\n{r13 = :.3f} ', '<' if r13 < r\_kr else '>', f' {r\_kr = :.3f}')  
  
 if r12 < r\_kr and r23 < r\_kr and r13 < r\_kr:  
 print('\nОднорідність підтверджується з ймовірністю ' + str(p))  
  
 normalized\_x1\_x2 = [  
 [-1, -1],  
 [-1, 1],  
 [1, -1]  
 ]  
  
 mx\_list = [sum(i) / len(i) for i in list(zip(normalized\_x1\_x2[0], normalized\_x1\_x2[1], normalized\_x1\_x2[2]))]  
 my = sum([average1, average2, average3]) / len([average1, average2, average3])  
 a1 = sum(i[0] \*\* 2 for i in normalized\_x1\_x2) / len(normalized\_x1\_x2)  
 a2 = sum(i[0] \* i[1] for i in normalized\_x1\_x2) / len(normalized\_x1\_x2)  
 a3 = sum(i[1] \*\* 2 for i in normalized\_x1\_x2) / len(normalized\_x1\_x2)  
 a11 = sum(  
 normalized\_x1\_x2[i][0] \* [average1, average2, average3][i] for i in range(len(normalized\_x1\_x2))) / len(  
 normalized\_x1\_x2)  
 a22 = sum(  
 normalized\_x1\_x2[i][1] \* [average1, average2, average3][i] for i in range(len(normalized\_x1\_x2))) / len(  
 normalized\_x1\_x2)  
 matrix\_b = [  
 [1, mx\_list[0], mx\_list[1]],  
 [mx\_list[0], a1, a2],  
 [mx\_list[1], a2, a3]  
 ]  
 matrix\_b1 = [  
 [my, mx\_list[0], mx\_list[1]],  
 [a11, a1, a2],  
 [a22, a2, a3]  
 ]  
 matrix\_b2 = [  
 [1, my, mx\_list[1]],  
 [mx\_list[0], a11, a2],  
 [mx\_list[1], a22, a3]  
 ]  
 matrix\_b3 = [  
 [1, mx\_list[0], my],  
 [mx\_list[0], a1, a11],  
 [mx\_list[1], a2, a22]  
 ]  
 b0 = determinant(matrix\_b1) / determinant(matrix\_b)  
 b1 = determinant(matrix\_b2) / determinant(matrix\_b)  
 b2 = determinant(matrix\_b3) / determinant(matrix\_b)  
  
 print('\nРозрахунок нормованих коефіцієнтів рівняння регресії:')  
  
 for i in normalized\_x1\_x2:  
 print(  
 f'ŷ = {b0:.3f} + {b1:.3f} \* {i[0]:2} + {b2:.3f} \* {i[1]:2}'  
 f' = {b0 + b1 \* i[0] + b2 \* i[1]:.3f}')  
  
 x10 = (x1\_max + x1\_min) / 2  
 x20 = (x2\_max + x2\_min) / 2  
 delta\_x1 = (x1\_max - x1\_min) / 2  
 delta\_x2 = (x2\_max - x2\_min) / 2  
  
 a\_0 = b0 - b1 \* (x10 / delta\_x1) - b2 \* (x20 / delta\_x2)  
 a\_1 = b1 / delta\_x1  
 a\_2 = b2 / delta\_x2  
  
 print('\nЗапишемо натуралізоване рівняння регресії:')  
 print(  
 f'ŷ = {a\_0:.3f} + {a\_1:.3f} \* {x1\_min:3} + {a\_2:.3f} \* {x2\_min:3}'  
 f' = {a\_0 + a\_1 \* x1\_min + a\_2 \* x2\_min:.3f}')  
 print(  
 f'ŷ = {a\_0:.3f} + {a\_1:.3f} \* {x1\_min:3} + {a\_2:.3f} \* {x2\_max:3}'  
 f' = {a\_0 + a\_1 \* x1\_min + a\_2 \* x2\_max:.3f}')  
 print(  
 f'ŷ = {a\_0:.3f} + {a\_1:.3f} \* {x1\_max:3} + {a\_2:.3f} \* {x2\_min:3}'  
 f' = {a\_0 + a\_1 \* x1\_max + a\_2 \* x2\_min:.3f}')  
  
 else:  
 print('\nОднорідність не підтвердилася, підвищуємо m на 1\n')  
 m += 1  
 main()  
  
  
main()

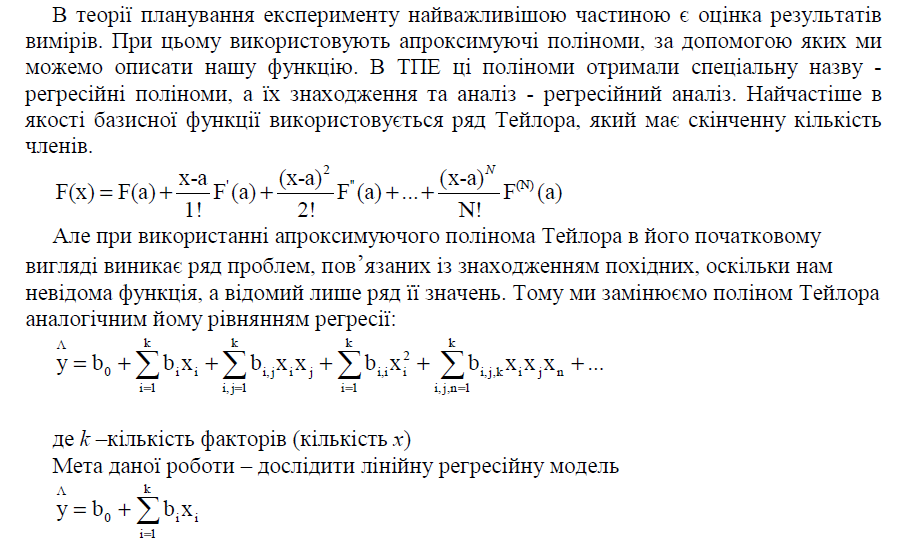
Результат виконання програми





**Теоретичні відомості**

**Що таке регресійні поліноми і де вони застосовуються?**



**Визначення однорідності дисперсії.**

Однорідність дисперсії означає, що серед усіх дисперсій нема таких, які б значно перевищували одна одну. Перевірка однорідності проводиться за допомогою різніх статистичних критеріїв.

**Що називається повним факторним експериментом**

Для знаходження коефіцієнтів у лінійному рівнянні регресії застосовують повний факторний експеримент (ПФЕ). Якщо в багатофакторному експерименті використані всі можливі комбінації рівнів факторів, то такий експеримент називається повним факторним експериментом.