Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**ЛАБОРАТОРНА РОБОТА № 2**

з дисципліни «Методи оптимізації та планування експерименту» на тему

«ПРОВЕДЕННЯ ДВОФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

ВИКОНАВ:

студент ІІ курсу ФІОТ

групи ІО-93

Мудрий Юрій

Варіант: 321

ПЕРЕВІРИВ:

Регіда П. Г.

Київ – 2021

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | -1 | -1 | 166 | 169 | 74 | 79 | 108 | 162 | 89 | 147 | 124,25 |
| 2 | -1 | +1 | 137 | 109 | 127 | 134 | 145 | 125 | 116 | 75 | 121 |
| 3 | +1 | -1 | 143 | 110 | 139 | 92 | 158 | 91 | 109 | 150 | 124 |

m=8 – необхідна кількість дослідів(визначено програмою, детальніше в прикладі роботи програми)

**Приклад роботи програми**

import itertools  
import numpy as np  
from random import \*  
import math  
from functools import \*  
  
  
"""  
constants  
"""  
  
y\_max = (30 - 28)\*10  
y\_min = (20 - 28)\*10  
  
x\_table = [[-1,-1],  
 [-1,+1],  
 [+1,-1]]  
  
p = 0.99  
  
  
x1\_min = -30  
x1\_max = 20  
x2\_min = -20  
x2\_max = 40  
  
naturalized\_x\_table = [[x1\_min, x2\_min],  
 [x1\_min, x2\_max],  
 [x1\_max, x2\_min]]  
  
"""  
Romanovsky criteria start  
"""  
  
  
def romanovsky\_criteria(y1: np.array, y2: np.array, y3: np.array):  
 def sigma\_theta(m):  
 return math.sqrt(abs(2\*(2\*m-2)/(m\*(m-4))))  
  
 def f\_uv(y\_u: np.array, y\_v: np.array):  
 dev\_u = np.var(y\_u)  
 dev\_v = np.var(y\_v)  
 return dev\_u/dev\_v if dev\_u > dev\_v else dev\_v/dev\_u  
  
 def theta\_uv(m: int, fuv: float):  
 return (m-2)/m \* fuv  
  
 def r\_uv(s\_t: float, s\_uv: float):  
 return abs(s\_uv - 1)/s\_t  
  
 def check\_criteria(R, m):  
 romanovsky\_criteria\_table = [[None, 2, 6, 8, 10, 12, 15, 20],  
 [0.99, 1.72, 2.16, 2.43, 2.62, 2.75, 2.90, 3.08],  
 [0.98, 1.72, 2.13, 2.37, 2.54, 2.66, 2.80, 2.96],  
 [0.95, 1.71, 2.10, 2.27, 2.41, 2.52, 2.64, 2.78],  
 [0.90, 1.69, 2.00, 2.17, 2.29, 2.39, 2.49, 2.62]]  
 column = romanovsky\_criteria\_table[0].index(sorted(filter(lambda el: el >= m, romanovsky\_criteria\_table[0][1:]))[0])  
 # in our case equals 1 (p = 0.99)  
 trusted\_probability\_row = 1  
 # defines necessary row depending of given p. Everything works,  
 # but it's unnecessary to add so much functionality in this labwork  
 # global p  
 # trusted\_probabilities = [row[0] for row in romanovsky\_criteria\_table[1:]]  
 # trusted\_probability\_row = trusted\_probabilities.index(min(filter(lambda el: el >= p, trusted\_probabilities)))  
 # trusted\_probability\_row = 1 if trusted\_probability\_row == 0 else trusted\_probability\_row  
 return R < romanovsky\_criteria\_table[trusted\_probability\_row][column]  
  
 global m  
 sTheta = sigma\_theta(m)  
 accordance = True  
 for combination in itertools.combinations((y1,y2,y3), 2):  
 fUV = f\_uv(combination[0], combination[1])  
 sUV = theta\_uv(m, fUV)  
 R = r\_uv(sTheta,sUV)  
 accordance \*= check\_criteria(R,m)  
 return accordance  
  
  
# Romanovsky criteria was tested for these experiment values (all rows' variations are drastically different)  
# to ensure correctness of our romanovsky\_criteria() function:  
#  
# m = 5  
# y\_table = [[-5,10000,1,1,1], [3,4,5,1,2], [4, -16,-20,3,8]]  
# all deviations of rows in this y\_table are drastically different, so romanovsky\_criteria(y\_table) must be false  
# result of test == false  
# everything works correctly  
  
"""  
Romanovsky criteria end  
  
Regression coefficients search start  
"""  
  
  
def experiment():  
 global m  
 return np.array([[randint(y\_min, y\_max) for \_ in range(m)] for \_ in range(3)])  
  
  
def normalized\_regression\_coeffs():  
 def m\_i(arr: np.array):  
 return np.average(arr)  
  
 def a\_i(arr: np.array):: #Обчислення нормованих коефіцієнтів рівняння регресії  
 return sum(arr\*\*2)/len(arr)  
  
 def a\_jj(arr1: np.array, arr2: np.array):  
 return reduce(lambda res, el: res+el[0]\*el[1], list(zip(arr1,arr2)), 0)/len(arr1)  
  
 global x\_table  
 global y\_table  
 y\_vals = np.array([np.average(i) for i in y\_table])  
 x1\_vals = np.array([i[0] for i in x\_table])  
 x2\_vals = np.array([i[1] for i in x\_table])  
 m\_x1 = m\_i(x1\_vals)  
 m\_x2 = m\_i(x2\_vals)  
 m\_y = m\_i(y\_vals)  
 a1 = a\_i(x1\_vals)  
 a2 = a\_jj(x1\_vals, x2\_vals)  
 a3 = a\_i(x2\_vals)  
 a11 = a\_jj(x1\_vals, y\_vals)  
 a22 = a\_jj(x2\_vals, y\_vals)  
 coeffs\_matrix = [[1, m\_x1, m\_x2],  
 [m\_x1, a1, a2],  
 [m\_x2, a2, a3]]  
 vals\_matrix = [m\_y, a11, a22]  
 b\_coeffs = list(map(lambda num: round(num, 2), np.linalg.solve(coeffs\_matrix, vals\_matrix)))  
 return b\_coeffs  
  
  
def assert\_normalized\_regression():  
 global b\_coeffs  
 global x\_table  
 global y\_table  
 y\_average\_experim\_vals = np.array([np.average(i) for i in y\_table])  
 print("\nПеревірка правильності знаходження коефіцієнтів рівняння регресії: ")  
 print("Середні експериментальні значення y для кожного рядка матриці планування: " +  
 ", ".join(map(str, y\_average\_experim\_vals)))  
 y\_theoretical = [b\_coeffs[0] + x\_table[i][0]\*b\_coeffs[1] + x\_table[i][1]\*b\_coeffs[2] for i in range(len(x\_table))]  
 print("Теоретичні значення y для кожного рядка матриці планування: ".ljust(74) + ", ".join(map(str, y\_theoretical)))  
 for i in range(len(x\_table)):  
 try:  
 assert round(y\_theoretical[i], 2) == round(y\_average\_experim\_vals[i],2)  
 except:  
 print("Неправильні результати пошуку коефіцієнтів рівняння регресії")  
 return  
 print("Правильні результати пошуку коефіцієнтів рівняння регресії")  
  
  
"""  
Regression coefficients search end  
"""  
  
  
def naturalized\_regression(b\_coeffs: list):  
 v = globals()  
 global x1\_max  
 global x1\_min  
 global x2\_max  
 global x2\_min  
 x1 = abs(x1\_max-x1\_min)/2  
 x2 = abs(x2\_max-x2\_min)/2  
 x10 = (x1\_max+x1\_min)/2  
 x20 = (x2\_max+x2\_min)/2  
 a0 = b\_coeffs[0]-b\_coeffs[1]\*x10/x1 - b\_coeffs[2]\*x20/x2  
 a1 = b\_coeffs[1]/x1  
 a2 = b\_coeffs[2]/x2  
 return [a0, a1, a2]  
  
  
def assert\_naturalized\_regression():  
 global y\_table  
 global naturalized\_x\_table  
 global a\_coeffs  
 y\_average\_experim\_vals = np.array([np.average(i) for i in y\_table])  
 print("\nПеревірка натуралізації коефіцієнтів рівняння регресії:")  
 print("Середні експериментальні значення y для кожного рядка матриці планування: " +  
 ", ".join(map(str, y\_average\_experim\_vals)))  
 y\_theoretical = [a\_coeffs[0] + naturalized\_x\_table[i][0]\*a\_coeffs[1]+ naturalized\_x\_table[i][1]\*a\_coeffs[2] for i in range(len(naturalized\_x\_table))]  
 print("Теоретичні значення y для кожного рядка матриці планування: ".ljust(74) + ", ".join(  
 map(str, y\_theoretical)))  
 for i in range(len(naturalized\_x\_table)):  
 try:  
 assert round(y\_theoretical[i],2) == round(y\_average\_experim\_vals[i],2)  
 except:  
 print("Неправильні результати натуралізації")  
 return  
 print("Правильні результати натуралізації")  
  
  
m = 5  
y\_table = experiment()  
  
while not romanovsky\_criteria(\*y\_table):  
 m += 1  
 y\_table = experiment()  
  
labels\_table = ["x1", "x2"] + ["y{}".format(i+1) for i in range(m)]  
rows\_table = [naturalized\_x\_table[i] + list(y\_table[i]) for i in range(3)]  
rows\_normalized\_table = [x\_table[i] + list(y\_table[i]) for i in range(3)]  
  
print("Матриця планування:")  
print((" "\*4).join(labels\_table))  
print("\n".join([" ".join(map(lambda j: "{:<+5}".format(j), rows\_table[i])) for i in range(len(rows\_table))]))  
print("\t")  
  
print("Нормована матриця планування:")  
print((" "\*4).join(labels\_table))  
print("\n".join([" ".join(map(lambda j: "{:<+5}".format(j), rows\_normalized\_table[i])) for i in range(len(rows\_normalized\_table))]))  
print("\t")  
  
b\_coeffs = normalized\_regression\_coeffs()  
print("Рівняння регресії для нормованих факторів: y = {0} {1:+}\*x1 {2:+}\*x2".format(\*b\_coeffs))  
assert\_normalized\_regression()  
a\_coeffs = naturalized\_regression(b\_coeffs)  
print("\nРівняння регресії для натуралізованих факторів: y = {0} {1:+}\*x1 {2:+}\*x2".format(\*a\_coeffs))  
assert\_naturalized\_regression()

**Контрольні запитання**

1. Що таке регресійні поліноми і де вони застосовуються?

Регресійний поліном – це рівняння регресії виду



використовується в ТПЕ для оцінки результатів вимірів.

1. Визначення однорідності дисперсії.

Однорідність дисперсій – властивість, коли дисперсії вимірювання функцій відгуку є однаковими, або близькими.

1. Що називається повним факторним експериментом?

ПФЕ – експеримент, в якому використовуються всі можливі комбінації рівнів факторів.