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»

ПРОВЕДЕННЯ ДВОФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ

ВИКОНАВ:
Студент ІІ курсу ФІОТ
Групи ІО-93
Сукач А.В.

ПЕРЕВІРИВ:
асистент
Регіда П.Г.

Київ 2021 р.

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

Варіант завдання:

Варіант	X_1		X_2	
	min	max	min	max
325	-20	15	25	45

$Y_{max}=(30-325)*10 = -2950$
$Y_{min}=(20-325)*10 = -3050$

Лістинг програми:

```
import itertools
import numpy as np
from random import *
import math
from functools import *

"""
constants
"""

y_max = (30 - 325)*10          # -2950
y_min = (20 - 325)*10          # -3050

x_table = [[-1,-1],
           [-1,+1],
           [+1,-1]]

p = 0.99

x1_min = -20
x1_max = 15
x2_min = 25
x2_max = 45

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])
        trusted_probability_row = 1
        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 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):
        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?
   Регресійні поліноми – це апроксимуючі поліноми, за допомогою яких ми можемо описати функцію. Застосовуються в теорії планування експерименту.

2. Визначення однорідності дисперсії.
   Опираючись на вимоги регресивного аналізу достовірне оброблення та використання вихідних даних експериментальних досліджень можливе лише тоді, коли дисперсії вимірювання функцій відгуку в кожній точці експерименту є однаковими. Дана властивість називається однорідністю дисперсії.

3. Що називається повним факторним експериментом?
   ПФЕ – багатофакторний експеримент в якому використовуються всі можливі комбінації рівні факторів. $N_{ПФЕ} = 2^k$ або 3^k або 5^k.

Результат виконання роботи:

```
/usr/local/bin/python3.9 "/Users/artem/Учеба/МОПЕ/МОРЕ programs/MOPE_2.py"
Матриця планування:
x1    x2    y1    y2    y3    y4    y5
-20   +25   -2964 -2955 -2953 -2984 -3033
-20   +45   -3040 -2976 -3028 -2960 -3032
+15   +25   -2995 -3019 -2986 -3005 -2984

Нормована матриця планування:
x1    x2    y1    y2    y3    y4    y5
-1    -1    -2964 -2955 -2953 -2984 -3033
-1    +1    -3040 -2976 -3028 -2960 -3032
+1    -1    -2995 -3019 -2986 -3005 -2984

Рівняння регресії для нормованих факторів: y = -3002.5 -10.0*x1 -14.7*x2

Перевірка правильності знаходження коефіцієнтів рівняння регресії:
Середні експериментальні значення y для кожного рядка матриці планування: -2977.8, -3007.2, -2997.8
Теоретичні значення y для кожного рядка матриці планування:               -2977.8, -3007.2, -2997.8
Правильні результати пошуку коефіцієнтів рівняння регресії

Рівняння регресії для натуралізованих факторів: y = -2952.4785714285717 -0.5714285714285714*x1 -1.47*x2

Перевірка натуралізації коефіцієнтів рівняння регресії:
Середні експериментальні значення y для кожного рядка матриці планування: -2977.8, -3007.2, -2997.8
Теоретичні значення y для кожного рядка матриці планування:               -2977.8, -3007.2000000000003, -2997.8
Правильні результати натуралізації

Process finished with exit code 0
```

Висновок:

В даній лабораторній роботі я провів двофакторний експеримент з перевіркою дисперсій на однорідність за критерієм Романовського і отримав коефіцієнти рівняння регресії. Також провів натуралізацію рівняння регресії.