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 наукових досліджень»
За темою:
«ПРОВЕДЕННЯ ДВОФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

ВИКОНАВ:
Студент ІІ курсу ФІОТ
Групи ІВ-91
Гришин О.С.
Номер у списку - 07

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.

Завдання:

1. Записати лінійне рівняння регресії.
2. Обрати тип двофакторного експерименту і скласти матрицю планування для
нього з використанням додаткового нульового фактору (xo=1).
3. Провести експеримент в усіх точках повного факторного простору (знайти значення функції відгуку y). Значення функції відгуку задати випадковим чином у відповідності до варіанту у діапазоні ymin ÷ ymax
ymax = (30 - Nваріанту)*10,
ymin = (20 - Nваріанту)*10.
Варіанти обираються по номеру в списку в журналі викладача.

107	-5	15	-15	35

Програмний код

```
if __name__ == '__main__':

    new_r = Romanovsky(107, -5, 15, -15, 35)

    new_r.run()
```

```
from math import sqrt
from random import randint

import pandas as pd
from numpy.linalg import det


class Romanovsky:
    m = 5
    average_y = []
    dispersion_y = []
    f_uv = []
    sigma_uv = []
    r_uv = []
    deviation = 0
    r_value = 0
    romanovsky_table = {(2, 3, 4): 1.72, (5, 6, 7): 2.13, (8, 9): 2.37, (10,
11): 2.54,
                        (12, 13): 2.66, (14, 15, 16, 17): 2.8, (18, 19, 20):
2.96}
    x1 = [-1, -1, 1]
    x2 = [-1, 1, -1]

    def __init__(self, var, x1_min, x1_max, x2_min, x2_max):
        self.x2_max = x2_max
        self.x2_min = x2_min
        self.x1_max = x1_max
        self.x1_min = x1_min
        self.var = var

        self.y_min = (20 - var) * 10
        self.y_max = (30 - var) * 10
```

```
        self.nx1 = [x1_min if self.x1[i] == -1 else x1_max for i in range(3)]
        self.nx2 = [x2_min if self.x2[i] == -1 else x2_max for i in range(3)]

        self.y_1 = [randint(self.y_min, self.y_max) for _ in range(self.m)]
        self.y_2 = [randint(self.y_min, self.y_max) for _ in range(self.m)]
        self.y_3 = [randint(self.y_min, self.y_max) for _ in range(self.m)]
        self.y_lists = [self.y_1, self.y_2, self.y_3]

    @staticmethod
    def make_df(f_names, rows):
        n_df = {i: [] for i in f_names}
        for row in rows:
            for i, val in enumerate(n_df):
                n_df[val].append(row[i])

        df = pd.DataFrame(data=n_df)
        print(df)

    def __dispersion_calc(self, y_list, y_avg) -> float:
        return sum([(i - y_avg) ** 2 for i in y_list]) / self.m

    def get_avarage_y(self) -> list:
        return [sum(self.y_1) / self.m, sum(self.y_2) / self.m, sum(self.y_3)
/ self.m]

    def get_dispersion_y(self) -> list:
        return [round(self.__dispersion_calc(self.y_lists[i],
self.get_avarage_y()[i]), 4) for i in range(3)]

    def get_deviation(self) -> float:
        return sqrt((2 * (2 * self.m - 2)) / self.m * (self.m - 4))

    def get_f_uv(self):
        uv = [[self.dispersion_y[0], self.dispersion_y[1]],
[self.dispersion_y[1], self.dispersion_y[2]],
              [self.dispersion_y[2], self.dispersion_y[0]]]
        return [round(max(uv[i]) / min(uv[i]), 4) for i in range(3)]

    def get_sigma(self):
        return [round(((self.m - 2) / self.m * f), 4) for f in self.f_uv]

    def get_r_uv(self):
        return [round((abs(sigma - 1) / self.deviation), 4) for sigma in
self.sigma_uv]

    def romanovsky_criterion(self) -> bool:
        self.average_y = self.get_avarage_y()

        self.dispersion_y = self.get_dispersion_y()

        self.deviation = self.get_deviation()

        self.f_uv = self.get_f_uv()

        self.sigma_uv = self.get_sigma()

        self.r_uv = self.get_r_uv()

        for key in self.romanovsky_table.keys():
            if self.m in key:
                self.r_value = self.romanovsky_table[key]
                break
        return max(self.r_uv) <= self.r_value
```

```
    def run(self):

        while not self.romanovsky_criterion():
            for i in self.y_lists:
                i.append((randint(self.y_min, self.y_max)))
            self.m += 1

        mx1, mx2, my = sum(self.x1) / 3, sum(self.x2) / 3,
sum(self.average_y) / 3
        a1 = sum([i ** 2 for i in self.x1]) / 3
        a2 = sum([self.x1[i] * self.x2[i] for i in range(3)]) / 3
        a3 = sum([i ** 2 for i in self.x2]) / 3

        a11 = sum([self.x1[i] * self.average_y[i] for i in range(3)]) / 3
        a22 = sum([self.x2[i] * self.average_y[i] for i in range(3)]) / 3

        determinant = det([[1, mx1, mx2], [mx1, a1, a2], [mx2, a2, a3]])
        b0 = det([[my, mx1, mx2], [a11, a1, a2], [a22, a2, a3]]) /
determinant
        b1 = det([[1, my, mx2], [mx1, a11, a2], [mx2, a22, a3]]) /
determinant
        b2 = det([[1, mx1, my], [mx1, a1, a11], [mx2, a2, a22]]) /
determinant

        delta_x1 = abs(self.x1_max - self.x1_min) / 2
        delta_x2 = abs(self.x2_max - self.x2_min) / 2
        x_10 = (self.x1_max + self.x1_min) / 2
        x_20 = (self.x2_max + self.x2_min) / 2

        nb0 = b0 - b1 * (x_10 / delta_x1) - b2 * (x_20 / delta_x2)
        nb1 = b1 / delta_x1
        nb2 = b2 / delta_x2

        f_names = ['X1', 'X2', *[f"Y{i}" for i in range(1, self.m + 1)]]
        rows = [[self.x1[i], self.x2[i], *self.y_lists[i]] for i in
range(len(self.y_lists))]
        self.make_df(f_names, rows)

        print('\n')

        f_names = ['AVG Y', 'Dispersion Y', 'F_uv', 'σ_uv', 'R_uv']
        rows = [[self.average_y[i], self.dispersion_y[i], self.f_uv[i],
self.sigma_uv[i], self.r_uv[i]] for i in
                range(len(self.y_lists))]
        self.make_df(f_names, rows)

        print('\n')

        f_names = ['NX1', 'NX2', 'AVG Y', 'Experimental']
        rows = [[self.nx1[i], self.nx2[i], self.average_y[i], round(nb0 + a1
* self.nx1[i] + a2 * self.nx2[i], 4)] for i
                in
                range(len(self.y_lists))]
        self.make_df(f_names, rows)

        print('\n')

        print('Рівняння')
        print(f"y = {round(b0, 4)} + {round(b1, 4)}*x1 + {round(b2, 4)}*x2")

        print('Нормоване')
        print(f"y = {round(nb0, 3)} + {round(nb1, 3)}*nx1 + {round(nb2,
3)}*nx2")
```

```
        print(f"Відхилення: {self.deviation}")
        print(f"Критерій Романовського: {self.r_value}")
```

Результати роботи програми

```
   X1  X2   Y1   Y2   Y3   Y4   Y5
0  -1  -1 -818 -833 -840 -818 -811
1  -1   1 -816 -797 -844 -778 -812
2   1  -1 -820 -796 -851 -853 -776

   AVG Y  Dispersion Y    F_uv    σ_uv    R_uv
0 -824.0        115.60  4.1301  2.4781  0.8263
1 -809.4        477.44  1.9097  1.1458  0.0815
2 -819.2        911.76  7.8872  4.7323  2.0864

   NX1  NX2  AVG Y  Experimental
0   -5  -15 -824.0     -818.4200
1   -5   35 -809.4     -835.0867
2   15  -15 -819.2     -798.4200

Рівняння
y = -814.3 + 2.4*x1 + 7.3*x2
Нормоване
y = -818.42 + 0.24*nx1 + 0.292*nx2
Відхилення: 1.7888543819998317
Критерій Романовського: 2.13
```

Контрольні запитання:

1. **В теорії планування експерименту найважливішою частиною є оцінка результатів вимірів. При цьому використовують апроксимуючі поліноми, за допомогою яких ми можемо описати нашу функцію. В ТПЕ ці поліноми отримали спеціальну назву - регресійні поліноми, а їх знаходження та аналіз - регресійний аналіз.**

2. **Обирають так названу «довірчу ймовірність» p – ймовірність, з якою вимагається підтвердити гіпотезу про однорідність дисперсій. У відповідності до p і кількості дослідів m обирають з таблиці критичне значення критерію. Кожне експериментальне значення Ruv критерію Романовського порівнюється з Rкр. (значення критерію Романовського за різних довірчих ймовірностей p) і якщо для усіх кожне Ruv < Rкр., то гіпотеза про однорідність дисперсій підтверджується з ймовірністю p.**

3. **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**