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 №3

“ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ”

Виконав:

студент групи ІВ-83

Дровнін П. А.

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: 10.

Варіант завдання: 310.





1. Лістинг програми:

import numpy as np

x1\_min = -25

x1\_max = -5

x2\_min = 10

x2\_max = 60

x3\_min = -5

x3\_max = 60

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

m, n = 3, 4

def main(m, n):

print("\nMatrix of code values")

norm\_x = np.array([

[+1, -1, -1, -1],

[+1, -1, +1, +1],

[+1, +1, -1, +1],

[+1, +1, +1, -1]

])

for i in range(len(norm\_x)):

print("{}.".format(i + 1), end="")

for j in range(len(norm\_x[i])):

print("{:4}".format(norm\_x[i][j]), end="")

print()

print("\nX-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]

])

for i in range(len(x)):

print("{}.".format(i + 1), end="")

for j in range(len(x[i])):

print("{:4}".format(x[i][j]), end="")

print()

print("\nY-matrix:")

y = np.random.randint(y\_min, y\_max, size=(n, m))

print("\nAverage of the response features:")

y\_av = np.sum(y, axis=1) / len(y[0])

y\_1, y\_2, y\_3, y\_4 = y\_av

print(f"y\_1 = {y\_1:.2f}\ny\_2 = {y\_2:.2f}\ny\_3 = {y\_3:.2f}\ny\_4 = {y\_4:.2f}")

mx\_1, mx\_2, mx\_3 = [i / len(x) for i in np.sum(x, axis=0)]

my = sum(y\_av) / len(y\_av)

a\_1 = sum([x[i][0] \* y\_av[i] for i in range(len(x))]) / len(x)

a\_2 = sum([x[i][1] \* y\_av[i] for i in range(len(x))]) / len(x)

a\_3 = sum([x[i][2] \* y\_av[i] for i in range(len(x))]) / len(x)

a\_11 = sum([x[i][0] \*\* 2 for i in range(len(x))]) / len(x)

a\_22 = sum([x[i][1] \*\* 2 for i in range(len(x))]) / len(x)

a\_33 = sum([x[i][2] \*\* 2 for i in range(len(x))]) / len(x)

a\_12 = sum([x[i][0] \* x[i][1] for i in range(len(x))]) / len(x)

a\_13 = sum([x[i][0] \* x[i][2] for i in range(len(x))]) / len(x)

a\_23 = a\_32 = sum([x[i][1] \* x[i][2] for i in range(len(x))]) / len(x)

det = np.linalg.det(

[[1, mx\_1, mx\_2, mx\_3], [mx\_1, a\_11, a\_12, a\_13], [mx\_2, a\_12, a\_22, a\_32], [mx\_3, a\_13, a\_23, a\_33]])

det\_0 = np.linalg.det(

[[my, mx\_1, mx\_2, mx\_3], [a\_1, a\_11, a\_12, a\_13], [a\_2, a\_12, a\_22, a\_32], [a\_3, a\_13, a\_23, a\_33]])

det\_1 = np.linalg.det(

[[1, my, mx\_2, mx\_3], [mx\_1, a\_1, a\_12, a\_13], [mx\_2, a\_2, a\_22, a\_32], [mx\_3, a\_3, a\_23, a\_33]])

det\_2 = np.linalg.det(

[[1, mx\_1, my, mx\_3], [mx\_1, a\_11, a\_1, a\_13], [mx\_2, a\_12, a\_2, a\_32], [mx\_3, a\_13, a\_3, a\_33]])

det\_3 = np.linalg.det(

[[1, mx\_1, mx\_2, my], [mx\_1, a\_11, a\_12, a\_1], [mx\_2, a\_12, a\_22, a\_2], [mx\_3, a\_13, a\_23, a\_3]])

b\_0 = det\_0 / det

b\_1 = det\_1 / det

b\_2 = det\_2 / det

b\_3 = det\_3 / det

b = [b\_0, b\_1, b\_2, b\_3]

print(f"\nThe normalized regression equation: y = {b\_0:.5f} + {b\_1:.5f} \* x1 + {b\_2:.5f} \* x2 + {b\_3:.5f} \* x3\n")

print("Audit:")

y\_1\_exp = b\_0 + b\_1 \* x[0][0] + b\_2 \* x[0][1] + b\_3 \* x[0][2]

y\_2\_exp = b\_0 + b\_1 \* x[1][0] + b\_2 \* x[1][1] + b\_3 \* x[1][2]

y\_3\_exp = b\_0 + b\_1 \* x[2][0] + b\_2 \* x[2][1] + b\_3 \* x[2][2]

y\_4\_exp = b\_0 + b\_1 \* x[3][0] + b\_2 \* x[3][1] + b\_3 \* x[3][2]

print(f"y\_1 = {b\_0:.3f} + {b\_1:.3f} \* {x[0][0]} + {b\_2:.3f} \* {x[0][1]} + {b\_3:.3f} \* {x[0][2]} = {y\_1\_exp:.3f}"

f"\ny\_2 = {b\_0:.3f} + {b\_1:.3f} \* {x[1][0]} + {b\_2:.3f} \* {x[1][1]} + {b\_3:.3f} \* {x[1][2]} = {y\_2\_exp:.3f}"

f"\ny\_3 = {b\_0:.3f} + {b\_1:.3f} \* {x[2][0]} + {b\_2:.3f} \* {x[2][1]} + {b\_3:.3f} \* {x[2][2]} = {y\_3\_exp:.3f}"

f"\ny\_4 = {b\_0:.3f} + {b\_1:.3f} \* {x[3][0]} + {b\_2:.3f} \* {x[3][1]} + {b\_3:.3f} \* {x[3][2]} = {y\_4\_exp:.3f}")

print("\n[ Kohren's test ]")

f\_1 = m - 1

f\_2 = n

s\_1 = sum([(i - y\_1) \*\* 2 for i in y[0]]) / m

s\_2 = sum([(i - y\_2) \*\* 2 for i in y[1]]) / m

s\_3 = sum([(i - y\_3) \*\* 2 for i in y[2]]) / m

s\_4 = sum([(i - y\_4) \*\* 2 for i in y[3]]) / m

s\_array = np.array([s\_1, s\_2, s\_3, s\_4])

gP = max(s\_array) / sum(s\_array)

table = {3: 0.6841, 4: 0.6287, 5: 0.5892, 6: 0.5598, 7: 0.5365, 8: 0.5175, 9: 0.5017, 10: 0.4884,

range(11, 17): 0.4366, range(17, 37): 0.3720, range(37, 145): 0.3093}

gT = table.get(m)

if (gP < gT):

print(f"The variance is homogeneous: Gp = {gP:.5} < Gt = {gT}")

else:

print(f"The variance is not homogeneous Gp = {gP:.5} < Gt = {gT}")

m = m + 1

main(m + 1, n, q)

return

print("\n[ Student's test ]")

s2\_B = s\_array.sum() / n

s2\_beta\_S = s2\_B / (n \* m)

s\_beta\_S = pow(s2\_beta\_S, 1 / 2)

beta\_0 = sum([norm\_x[i][0] \* y\_av[i] for i in range(len(norm\_x))]) / n

beta\_1 = sum([norm\_x[i][1] \* y\_av[i] for i in range(len(norm\_x))]) / n

beta\_2 = sum([norm\_x[i][2] \* y\_av[i] for i in range(len(norm\_x))]) / n

beta\_3 = sum([norm\_x[i][3] \* y\_av[i] for i in range(len(norm\_x))]) / n

t = [abs(beta\_0) / s\_beta\_S, abs(beta\_1) / s\_beta\_S, abs(beta\_2) / s\_beta\_S, abs(beta\_3) / s\_beta\_S]

f3 = f\_1 \* f\_2

t\_table = {8: 2.306, 9: 2.262, 10: 2.228, 11: 2.201, 12: 2.179, 13: 2.160, 14: 2.145, 15: 2.131, 16: 2.120,

17: 2.110, 18: 2.101, 19: 2.093, 20: 2.086, 21: 2.08, 22: 2.074, 23: 2.069, 24: 2.064, 25: 2.06}

d = 4

for i in range(len(t)):

if (t\_table.get(f3) > t[i]):

b[i] = 0

d -= 1

print(f"Regression equation: y = {b[0]:.3f} + {b[1]:.3f} \* x1 + {b[2]:.3f} \* x2 + {b[3]:.3f} \* x3")

check\_0 = b[0] + b[1] \* x[0][0] + b[2] \* x[0][1] + b[3] \* x[0][2]

check\_1 = b[0] + b[1] \* x[1][0] + b[2] \* x[1][1] + b[3] \* x[1][2]

check\_2 = b[0] + b[1] \* x[2][0] + b[2] \* x[2][1] + b[3] \* x[2][2]

check\_3 = b[0] + b[1] \* x[3][0] + b[2] \* x[3][1] + b[3] \* x[3][2]

ckeck\_list = [check\_0, check\_1, check\_2, check\_3]

print("Values are normalized: ", ckeck\_list)

print("\n[ Fisher's test ]")

f\_4 = n - d

s2\_ad = m / f\_4 \* sum([(ckeck\_list[i] - y\_av[i]) \*\* 2 for i in range(len(y\_av))])

fP = s2\_ad / s2\_B

fT = [

[164.4, 199.5, 215.7, 224.6, 230.2, 234],

[18.5, 19.2, 19.2, 19.3, 19.3, 19.3],

[10.1, 9.6, 9.3, 9.1, 9, 8.9],

[7.7, 6.9, 6.6, 6.4, 6.3, 6.2],

[6.6, 5.8, 5.4, 5.2, 5.1, 5],

[6, 5.1, 4.8, 4.5, 4.4, 4.3],

[5.5, 4.7, 4.4, 4.1, 4, 3.9],

[5.3, 4.5, 4.1, 3.8, 3.7, 3.6],

[5.1, 4.3, 3.9, 3.6, 3.5, 3.4],

[5, 4.1, 3.7, 3.5, 3.3, 3.2],

[4.8, 4, 3.6, 3.4, 3.2, 3.1],

[4.8, 3.9, 3.5, 3.3, 3.1, 3],

[4.7, 3.8, 3.4, 3.2, 3, 2.9],

[4.6, 3.7, 3.3, 3.1, 3, 2.9],

[4.5, 3.7, 3.3, 3.1, 2.9, 2.8],

[4.5, 3.6, 3.2, 3, 2.9, 2.7],

[4.5, 3.6, 3.2, 3, 2.8, 2.7],

[4.4, 3.6, 3.2, 2.9, 2.8, 2.7],

[4.4, 3.5, 3.1, 2.9, 2.7, 2.6],

[4.4, 3.5, 3.1, 2.9, 2.7, 2.6]

]

if (fP > fT[f3][f\_4]):

print(f"fp = {fP} > ft = {fT[f3][f\_4]}.\nThe mathematical model is not adequate to the experimental data\n")

else:

print(f"fP = {fP} < fT = {fT[f3][f\_4]}.\nThe mathematical model is adequate to the experimental data\n")

print("\nRegression equation --- y = b\_0 + b\_1 \* x1 + b\_1 \* x2 +b\_3 \* x3")

main(m, n)

Результати виконання роботи

Regression equation --- y = b\_0 + b\_1 \* x1 + b\_1 \* x2 +b\_3 \* x3

Matrix of code values

1. 1 -1 -1 -1

2. 1 -1 1 1

3. 1 1 -1 1

4. 1 1 1 -1

X-matrix:

1. -25 10 -5

2. -25 60 60

3. -5 10 60

4. -5 60 -5

Y-matrix:

Average of the response features:

y\_1 = 218.33

y\_2 = 206.33

y\_3 = 199.33

y\_4 = 205.67

The normalized regression equation: y = 205.90321 + -0.49167 \* x1 + -0.05667 \* x2 + -0.14103 \* x3

Audit:

y\_1 = 205.903 + -0.492 \* -25 + -0.057 \* 10 + -0.141 \* -5 = 218.333

y\_2 = 205.903 + -0.492 \* -25 + -0.057 \* 60 + -0.141 \* 60 = 206.333

y\_3 = 205.903 + -0.492 \* -5 + -0.057 \* 10 + -0.141 \* 60 = 199.333

y\_4 = 205.903 + -0.492 \* -5 + -0.057 \* 60 + -0.141 \* -5 = 205.667

[ Kohren's test ]

The variance is homogeneous: Gp = 0.44859 < Gt = 0.6841

[ Student's test ]

Regression equation: y = 205.903 + -0.492 \* x1 + 0.000 \* x2 + -0.141 \* x3

Values are normalized: [218.90000000000057, 209.73333333333397, 199.90000000000055, 209.06666666666715]

[ Fisher's test ]

fP = 1.6493059125970162 < fT = 4.3.

The mathematical model is adequate to the experimental data