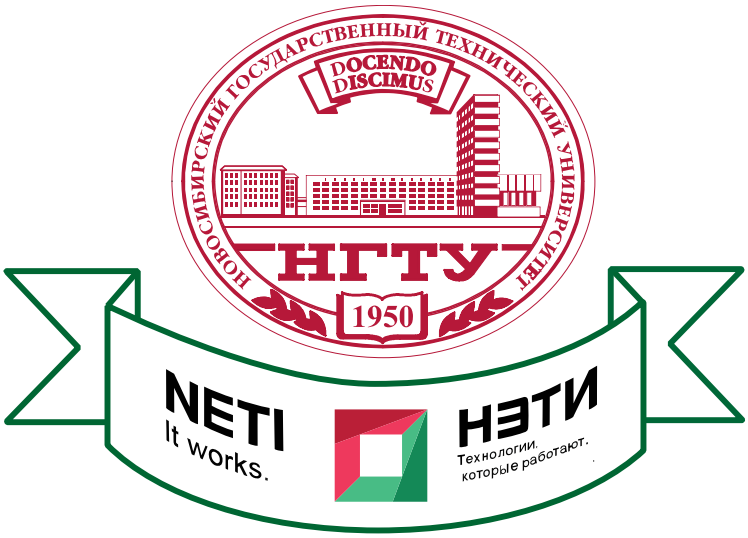
Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования

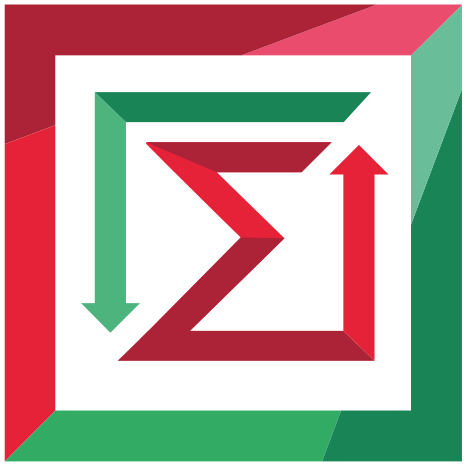
«Новосибирский государственный технический университет»



Кафедра теоретической и прикладной информатики

Лабораторная работа № 4

по дисциплине «Статистические методы анализа данных»



Факультет:	ПМИ
Группа:	ПМИ-02
Вариант:	6
Студент:	Сидоров Даниил,
	Дюков Богдан
Преподаватель:	Попов Александр Александрович.

Новосибирск

2026

1. Постановка задачи

1. Провести моделирование регрессионного процесса с гетероскедастичным возмущением.

2. Полученные данные проверить по тестам на наличие гетероскедастичности.

3. Оценить параметры регрессионной модели по доступному обобщенному МНК и по обыкновенному МНК.

4. Сравнить эффективность оценок в этих двух случаях по квадрату их расстояния до известных истинных значений параметров.

   Дисперсия возмущений для 6 варианта – экспонента от взвешенной суммы квадратов факторов.

2. Ход работы

Моделирование регрессионного процесса

$\eta(\mathrm{x}, \theta) = \theta^{\mathrm{T}} f(x_1, x_2, x_3)$;

$f(x_1, x_2, x_3) = (1, x_1, x_2, x_3, x_1x_2, x_1x_3, x_1^2, x_2^2)$;

$\theta = (1, 2, 5, 4, 1.5, 2.5, 0.02, 0.01)^{\mathrm{T}}$;

$x_i \in [-1, 1], \qquad i = \overline{1 \ldots 3}$;

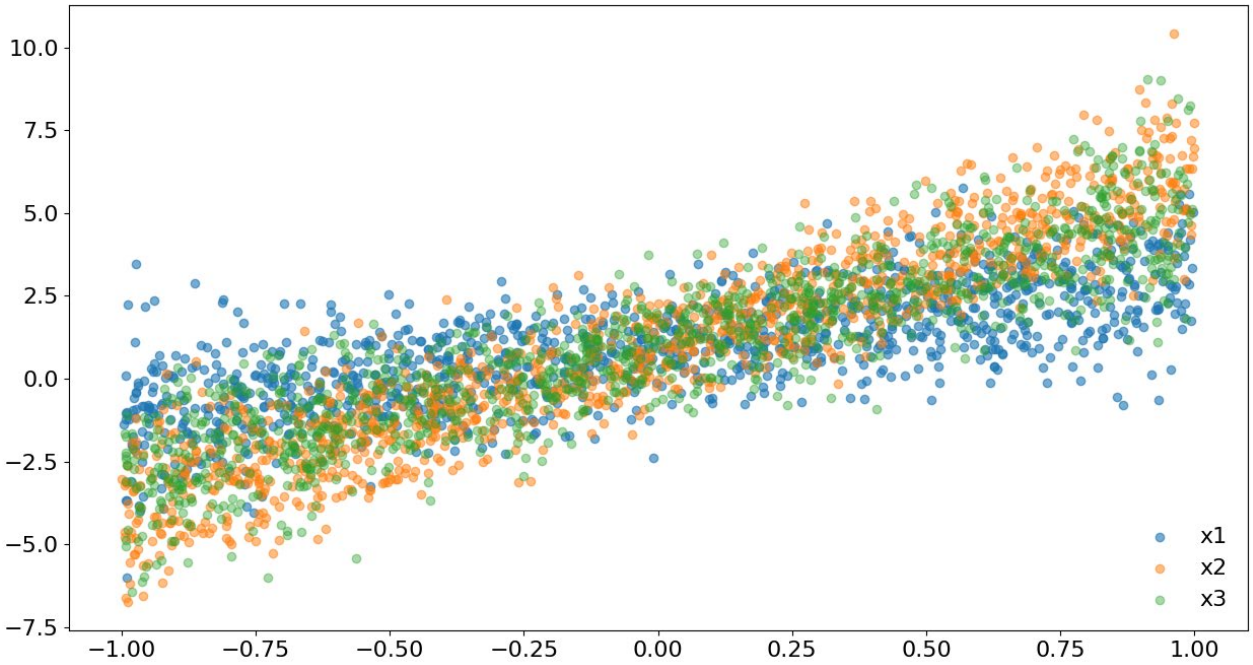
$\varepsilon_{\mathrm{i}} \sim N(0, {\sigma_i}^2)$;

${\sigma_i}^2 = \exp(\mathrm{w1} * \mathrm{x1}^2 + \mathrm{w2} * \mathrm{x2}^2 + \mathrm{w3} * \mathrm{x3}^2)$;

Количество экспериментов $n = 1200$;

Число параметров регрессии $m = 8$.

Точечная диаграмма зависимости отклика от значений факторов:



Можно видеть гетероскедастичность.

Проверка гетероскедастичности с помощью теста Бреуша-Пагана

Вектор известных переменных: $z_t^T = (1, \exp(0.8 * \text{x}1^2 + 0.8 * \text{x}2^2 + 0.9 * \text{x}3^2))$.

Вектор неизвестных параметров: $\alpha^T = (\alpha_0, \alpha_1)$.

Гипотеза о гомоскедастичности в нашем случае имеет вид: $\alpha_1 = 0$.

- оценим исходное уравнения по МНК, а также дисперсию:

$$e_t = y_t - f(x_t)^T \hat{\theta};$$

$$\tilde{\sigma}^2 = \sum \frac{{e_t}^2}{n} = 8.97.$$

- построим регрессию с откликом:

$$c_t = \frac{e_t^2}{\tilde{\sigma}^2};$$

и найдем предсказанные значения нормированных квадратов остатков:

$$\hat{c}_t = \hat{\alpha}^{\mathrm{T}} z_t;$$

Гипотеза о гомоскедастичности будет принята, если

$$\frac{ESS}{2} = \frac{\sum (c_t - \bar{c})^2}{2} \sim \chi^2_{0.05,1} = 3.84922;$$

Получаем:

$$\frac{\sum(c_t - \bar{c})^2}{2} = 696.59091 > 3.84922;$$

Гипотеза о гомоскедастичности не принимается.

Проверка гетероскедастичности с помощью теста Голдфельда-Квандтона

Предположение: источник нарушения гомоскедастичности взят в форме:

$$E(\varepsilon_i^2) = \exp(0.8 * \mathrm{x}1^2 + 0.8 * \mathrm{x}2^2 + 0.9 * \mathrm{x}3^2);$$

- Упорядочим последовательность наблюдений в соответствии с величиной:

$$\exp(0.8 * \mathrm{x}1^2 + 0.8 * \mathrm{x}2^2 + 0.9 * \mathrm{x}3^2).$$

- опустим n_c наблюдений, оказавшихся в середине упорядоченной выборки:

$$n_c = \frac{n}{3} = 400.$$

- Гипотеза о гомоскедастичности будет принята, если:

$$\frac{RSS_2}{RSS_1} \sim F_{\alpha, \frac{n-n_c-2k}{2}, \frac{n-n_c-2k}{2}} = F_{0.05,992,992} = 1.18099;$$

где k = m – число параметров в регрессии.

Получаем:

$$\frac{RSS_2}{RSS_1} = 8.84023 > 1.18099.$$

Гипотеза о гомоскедастичности не принимается.

Оценка параметров регрессионной модели

Истинные значения параметров:

$\theta = (1, 2, 5, 4, 1.5, 2.5, 0.02, 0.01)^{\mathrm{T}};$

Оценки параметров регрессионной модели по обыкновенному МНК:

$\theta_{ls} = (0.819, 2.1373, 4.8642, 3.9502, 1.6999, 2.764, -0.0659, 0.431)^{\mathrm{T}}$;

Оценки параметров регрессионной модели по доступному обобщенному МНК:

$\theta_{fgls} = (0.8105, 2.2528, 4.9771, 3.9225, 1.7805, 2.7216, 0.2131, 0.1337)^{\mathrm{T}}$;

Сравним эффективность оценок в этих двух случаях по квадрату их расстояния до истинных значений:

$$R_{ls} = (\theta - \theta_{ls})^T(\theta - \theta_{ls}) = 0.36683;$$

$$R_{gls} = \left(\theta - \theta_{gls}\right)^T\left(\theta - \theta_{gls}\right) = 0.28669.$$

Доступный обобщенный МНК эффективнее обыкновенного МНК.

3. Код программы

```
import numpy as np
from scipy.stats import f
import matplotlib.pyplot as plt
np.random.seed(10)


# Генерация комбинаций факторов
def generate_random(n):
    x1_list = [np.random.uniform(-1, 1) for _ in range(n)]
    x2_list = [np.random.uniform(-1, 1) for _ in range(n)]
    x3_list = [np.random.uniform(-1, 1) for _ in range(n)]

    return map(np.array, [x1_list, x2_list, x3_list])


def get_dataframe_for_graphs(x1, x2, x3):
    # Вычисление отклика без шума для каждого фактора при нулевых значениях остальных факторов
    u1 = theta[0] + theta[1] * x1 + theta[6] * x1 ** 2
    u2 = theta[0] + theta[2] * x2 + theta[7] * x2 ** 2
    u3 = theta[0] + theta[3] * x3

    # Вычисление весов для каждого фактора
    weights = np.array([0.8, 0.8, 0.9])
```

```
    # Вычисление взвешенной суммы квадратов факторов
    weighted_sum_squares_1 = weights[0] * x1 ** 2
    weighted_sum_squares_2 = weights[1] * x2 ** 2
    weighted_sum_squares_3 = weights[2] * x3 ** 2

    # Вычисление sigma2 как экспоненты от взвешенной суммы квадратов факторов
    sigma2_1 = np.exp(weighted_sum_squares_1)
    sigma2_2 = np.exp(weighted_sum_squares_2)
    sigma2_3 = np.exp(weighted_sum_squares_3)

    e1 = np.random.normal(0, np.sqrt(sigma2_1), size=n)
    e2 = np.random.normal(0, np.sqrt(sigma2_2), size=n)
    e3 = np.random.normal(0, np.sqrt(sigma2_3), size=n)

    y1 = u1 + e1
    y2 = u2 + e2
    y3 = u3 + e3

    # Построение точек для каждого фактора с использованием разных цветов и прозрачности
    fig = plt.figure(figsize=(15, 8))
    plt.scatter(x1, y1, alpha=0.6, label='x1')
    plt.scatter(x2, y2, alpha=0.5, label='x2')
    plt.scatter(x3, y3, alpha=0.4, label='x3')
    plt.legend(loc='lower right', frameon=False, prop={'size': 16})
    plt.xticks(fontsize=16)
    plt.yticks(fontsize=16)
    plt.show()

n=1200
m=8

# Определение параметров
theta = np.array([1, 2, 5, 4, 1.5, 2.5, 0.02, 0.01])

#---------- Задание №1 ----------#
# Генерация комбинаций факторов
x1, x2, x3 = generate_random(n)

# Вычисление истинного отклика без шума
u = theta[0] + theta[1]*x1 + theta[2]*x2 + theta[3]*x3 \
    + theta[4]*x1*x2 + theta[5]*x1*x3 \
    + theta[6]*x1**2 + theta[7]*x2**2
```

```
# Вычисление весов для каждого фактора
weights = np.array([0.8, 0.8, 0.9])

# Вычисление взвешенной суммы квадратов факторов
weighted_sum_squares = weights[0]*x1**2 + weights[1]*x2**2 + weights[2]*x3**2

# Вычисление sigma2 как экспоненты от взвешенной суммы квадратов факторов
sigma2 = np.exp(weighted_sum_squares)

e = np.random.normal(0, sigma2, size=n)

y = u + e

get_dataframe_for_graphs(x1,x2,x3)

#---------- Задание №2.1. Тест Бреуша-Пагана ----------#
X = np.column_stack((np.ones(len(x1)), x1, x2, x3, x1*x2, x1*x3, x1**2, x2**2))

# МНК-оценки
theta_hat = np.linalg.inv(X.T @ X) @ X.T @ y

e_t = y - X @ theta_hat

sigma_hat_squared = np.sum(pow(e_t, 2) / n)

print(sigma_hat_squared);

# Построение регрессии
c_t = e_t**2 / sigma_hat_squared
z_t = np.column_stack((np.ones(len(x1)), np.exp(weights[0]*x1**2 + weights[1]*x2**2 +
weights[2]*x3**2)))
alpha_hat = np.linalg.inv(z_t.T @ z_t) @ z_t.T @ c_t

# Вычисление предсказанных значений
c_hat = z_t @ alpha_hat

# Вычисление ESS
ESS = np.sum((c_hat - np.mean(c_t))**2)

# Вычисление статистики теста Бреуша-Пагана
BP = ESS / 2
FT = f.ppf(1-0.05, 1, n)
```

```
if BP > FT:
    print("{0} > {1} => гипотеза о гомоскедастичности не принимается".format(BP.round(5), FT.round(5)))
else:
    print("{0} < {1} => Гипотеза о гомоскедастичности принимается".format(BP.round(5), FT.round(5)))

#---------- Задание №2.2. Тест Голдфельда-Квандтона ----------#
# Упорядочивание наблюдений по взвешенной сумме
order = np.argsort(weighted_sum_squares)
x1_ordered = x1[order]
X_ordered = X[order]
y_ordered = y[order]

# Определение n_c и разделение данных на две части
n_c = n // 3
X1 = X_ordered[:(n-n_c)//2]
y1 = y_ordered[:(n-n_c)//2]
X2 = X_ordered[(n+n_c)//2:]
y2 = y_ordered[(n+n_c)//2:]

# Оценка параметров модели для каждой части данных
theta_hat1 = np.linalg.inv(X1.T @ X1) @ X1.T @ y1
theta_hat2 = np.linalg.inv(X2.T @ X2) @ X2.T @ y2

# Вычисление RSS для каждой части данных
RSS1 = np.sum((y1 - X1 @ theta_hat1)**2)
RSS2 = np.sum((y2 - X2 @ theta_hat2)**2)

# Вычисление статистики теста Голдфельда-Квандтона
GQ = RSS2 / RSS1

# Вычисление критического значения F-распределения
F_crit = f.ppf(1-0.05, (n-n_c-2*m)/2, (n-n_c-2*m)/2)

if GQ > F_crit:
    print("{0} > {1} => гипотеза о гомоскедастичности не принимается".format(GQ.round(5),
F_crit.round(5)))
else:
    print("{0} < {1} => Гипотеза о гомоскедастичности принимается".format(GQ.round(5), F_crit.round(5)))

#---------- Задание №3 ----------#
# Вычисление весов
sigma_hat_squared = np.exp(z_t @ alpha_hat)
```

```
W = np.diag(sigma_hat_squared)
theta_hat_fgls = np.linalg.inv(X.T @ np.linalg.inv(W) @ X) @ X.T @ np.linalg.inv(W) @ y



print("Оценки параметров модели:", theta)
print("Оценки параметров модели по обыкновенному МНК:", theta_hat.round(4))
print("Оценки параметров модели по обобщенному МНК:", theta_hat_fgls.round(4))

#---------- Задание №4 ----------#
# Вычисление квадрата расстояния между оценками параметров и истинными значениями
distance_squared_ls = np.sum((theta_hat - theta) ** 2)
distance_squared_fgls = np.sum((theta_hat_fgls - theta) ** 2)

print("Квадрат расстояния для оценок МНК:", distance_squared_ls.round(5))
print("Квадрат расстояния для оценок обобщенного МНК:", distance_squared_fgls.round(5))
```