МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Новосибирский государственный технический университет»

НГТУ НЭТИ | Факультет прикладной математики и информатики

Кафедра теоретической и прикладной информатики

Лабораторная работа №4
по дисциплине «Статистические методы анализа данных»

Студенты ИВАНОВ ВЛАДИСЛАВ (92)

ОБЕРШТ ЕЛЕНА (93)

Вариант 5

Преподаватель ПОПОВ АЛЕКСАНДР АЛЕКСАНДРОВИЧ

Новосибирск, 2022

1 Постановка задачи

Провести моделирование регрессионного процесса с гетероскедастичным возмущением. Полученные данные проверить по тестам на наличие гетероскедастичности. Оценить параметры регрессионной модели по доступному обобщенному МНК и по обыкновенному МНК. Сравнить эффективность оценок в этих двух случаях по квадрату их расстояния до известных истинных значений параметров.

Дисперсия возмущений – убывающая функция от модуля взаимодействия первого и второго факторов.

2 Моделирование процесса

$$f(x) = (1, x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3)^T$$

$$\theta = (1, 1, 1, 1, 0.01, 0.01, 0.01, 1, 1, 1)^T$$

$$x_i \in [-1, 1]$$

$$i = 1, 2, 3$$

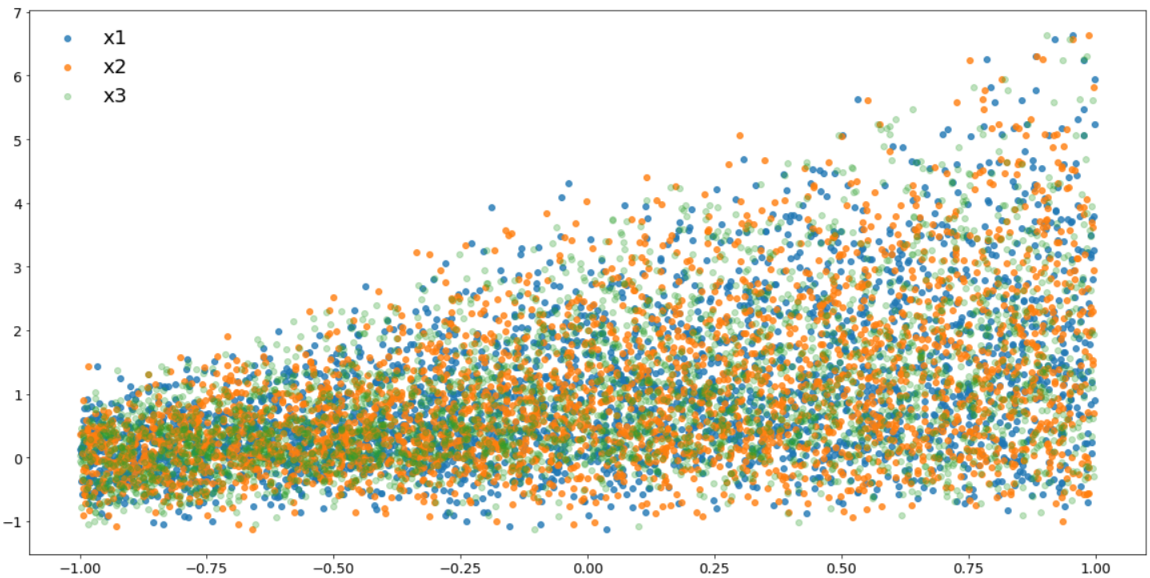
$$\epsilon \in N(0, \sigma^2)$$

$$\sigma^2 = 0.5(1 - |x_1x_2|)$$

$$n = 3000$$

$$m = 9$$

Визуализация зависимости значений модели от значений её параметров:



Наблюдается гетероскедастичность.

3 Проверка гетероскедастичности

3.1 Тест Бреуша-Пагана

$$z_t^T = (1, 0.5(1 - |x_1 x_2|))$$
$$\alpha^T = (\alpha_0, \alpha_1)$$

Гипотеза о гомоскедастичности:

$$\alpha_1 = 0$$

Оценим исходное уравнение и дисперсию (ОМП):

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

$$\widetilde{\sigma}^2 = \Sigma(\frac{e_t^2}{n}) = 0.15$$

Построим регрессию с откликом:

$$c_t = \frac{e_t^2}{\widetilde{\sigma}^2}$$

Найдем предсказанные значения нормированных квадратов остатков (не учитывая свободный член):

$$\hat{c}_t = \hat{\alpha}^T Z_t$$

Гипотеза о гомоскедантичности принимается, если

$$\frac{ESS}{2} = \Sigma(\hat{c}_t - \bar{c})^2 \sim \chi^2_{0.05,1} = 3.84$$

Получаем:

$$\Sigma(\hat{c}_t - \bar{c})^2 = 640.8$$

Гипотеза отвергается.

3.2 Тест Голдфельда-Квандтона

Предположим, что источник нарушения гомоскедастичности взят в форме:

$$E(e_i^2) = 0.5(1 - |x_1 x_2|)$$

Упорядочим последовательность наблюдений в соответствии с величиной отклика. Опустим

$$n_c = n/3 = 1000$$

наблюдений в середине выборки, оценим RSS для первых 1000 и последних 1000 наблюдений.

Гипотеза о гомоскедантичности принимается, если

$$\frac{RSS_2}{RSS_1} \sim F_{\alpha, \frac{n-n_c-2m}{2}, \frac{n-n_c-2m}{2}} = F_{0.05,991,991} \approx 1.11$$

Получаем:

$$\frac{RSS_2}{RSS_1} = 1.99$$

Гипотеза отвергается.

4 Оценивание параметров модели

$$\theta_{LS} = (0.9702, 0.9899, 0.9891, 1.0136, 0.0692, 0.0244, -0.0051, 1.0054, 0.9949, 1.0011)^T$$

$$\theta_{GLS} = (0.9744, 0.9929, 0.9877, 1.0165, 0.0631, 0.0191, -0.0066, 1.0045, 0.9972, 1.0033)^T$$

$$\theta = (1, 1, 1, 1, 0.01, 0.01, 0.01, 1, 1, 1)^T$$

Сравним эффективность оценок по квадрату их расстояния до истинных значений:

$$R_{LS} = (\theta - \theta_{LS})^T(\theta - \theta_{LS}) = 0.005$$

$$R_{GLS} = (\theta - \theta_{GLS})^T(\theta - \theta_{GLS}) = 0.003$$

Доступный обобщенный МНК немного эффективнее обыкновенного.

5 Код программы

```python
import pandas as pd
import numpy as np
import random
import scipy.stats
from matplotlib import pyplot as plt
from scipy.linalg import toeplitz
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.decomposition import PCA
np.set_printoptions(suppress=True)
random.seed(42)

n = 3000
x1j, x2j, x3j, yj = [], [], [], []

theta = [1, 1, 1, 1, 0.01, 0.01, 0.01, 1, 1, 1]

def u(x1,x2,x3):
    return theta[0] + theta[1]*x1 + theta[2]*x2 + theta[3]*x3 + theta[4]*x1**2 + theta[5]*x2**2 + theta[6]*x3**2 + theta[7]*x1*x2 + theta[8]*x1*x3 + theta[9]*x2*x3

for i in range(n):
    x1, x2, x3 = random.uniform(-1, 1), random.uniform(-1, 1), random.uniform(-1, 1)
    sigma2 = 0.5*(1-abs(x1*x2))
    e = np.random.normal(0, sigma2)
    y = u(x1,x2,x3) + e
    x1j.append(x1), x2j.append(x2), x3j.append(x3), yj.append(y)

fig = plt.figure(figsize=(20,10))
```

```
plt.scatter(x1j, yj, alpha=0.8, label='x1')
plt.scatter(x2j, yj, alpha=0.8, label='x2')
plt.scatter(x3j, yj, alpha=0.3, label='x3')
plt.legend(loc='upper left', frameon=False, prop={'size': 20})
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()

X = np.array([np.ones(n),
              np.reshape(x1j, (n, )),
              np.reshape(x2j, (n, )),
              np.reshape(x3j, (n, )),
              pow(np.reshape(x1j, (n, )), 2),
              pow(np.reshape(x2j, (n, )), 2),
              pow(np.reshape(x3j, (n, )), 2),
              np.array(x1j)*np.array(x2j),
              np.array(x1j)*np.array(x3j),
              np.array(x2j)*np.array(x3j)]).T
y = np.reshape(yj, (n, ))

et = y - np.dot(X, theta_ls)
sigma2_new = sum(pow(et, 2)/n)
print('sigma2_new =', sigma2_new)

V = sigma2_new ** toeplitz(np.arange(n))
theta_ls = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
theta_gls = np.dot(np.linalg.inv(np.dot(np.dot(X.T, np.linalg.inv(V)),
    X)), np.dot(np.dot(X.T, np.linalg.inv(V)), y))
print('theta_ls =', np.around(theta_ls, 4))
print('theta_gls =', np.around(theta_gls, 4))

R_ls = np.dot((theta - theta_ls).T, (theta - theta_ls))
R_gls = np.dot((theta - theta_gls).T, (theta - theta_gls))
print('R_ls =', R_ls)
print('R_gls =', R_gls)

Zt = np.array([np.ones(n), 0.5*(1-abs(np.array(x1j)*np.array(x2j)))]).T
ct = pow(et, 2)/sigma2_new
alpha = np.dot(np.linalg.inv(np.dot(Zt.T, Zt)), np.dot(Zt.T, ct))
c_new = np.dot(Zt, alpha)
ess_div_2 = sum((c_new - np.mean(ct))**2)
FT = scipy.stats.f.ppf(q=1-0.05, dfn=1, dfd=n)
print('F =', ess_div_2)
print('FT =', FT)

zipped_lists = zip(yj, x1j, x2j, x3j)
sorted_pairs = sorted(zipped_lists)
tuples = zip(*sorted_pairs)
yj, x1j, x2j, x3j = [ list(tuple) for tuple in  tuples]
```

```python
X = np.array([np.ones(n)[:1000],
              np.reshape(x1j, (n, ))[:1000],
              np.reshape(x2j, (n, ))[:1000],
              np.reshape(x3j, (n, ))[:1000],
              pow(np.reshape(x1j, (n, )), 2)[:1000],
              pow(np.reshape(x2j, (n, )), 2)[:1000],
              pow(np.reshape(x3j, (n, )), 2)[:1000],
              np.array(x1j)[:1000]*np.array(x2j)[:1000],
              np.array(x1j)[:1000]*np.array(x3j)[:1000],
              np.array(x2j)[:1000]*np.array(x3j)[:1000]]).T
y = np.reshape(yj, (n, ))[:1000]
theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
y_hat = np.dot(X, theta)
RSS1 = sum((y - y_hat)**2)

zipped_lists = zip(yj, x1j, x2j, x3j)
sorted_pairs = sorted(zipped_lists)
tuples = zip(*sorted_pairs)
yj, x1j, x2j, x3j = [ list(tuple) for tuple in  tuples]
X = np.array([np.ones(n)[2000:],
              np.reshape(x1j, (n, ))[2000:],
              np.reshape(x2j, (n, ))[2000:],
              np.reshape(x3j, (n, ))[2000:],
              pow(np.reshape(x1j, (n, )), 2)[2000:],
              pow(np.reshape(x2j, (n, )), 2)[2000:],
              pow(np.reshape(x3j, (n, )), 2)[2000:],
              np.array(x1j)[2000:]*np.array(x2j)[2000:],
              np.array(x1j)[2000:]*np.array(x3j)[2000:],
              np.array(x2j)[2000:]*np.array(x3j)[2000:]]).T
y = np.reshape(yj, (n, ))[2000:]
theta = np.dot(np.linalg.inv(np.dot(X.T, X)), np.dot(X.T, y))
y_hat = np.dot(X, theta)
RSS2 = sum((y - y_hat)**2)

FT = scipy.stats.f.ppf(q=1-0.05, dfn=991, dfd=991)
print('F =', RSS2 / RSS1)
print('FT =', FT)
```