|  |  |  |
| --- | --- | --- |
| 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» | | |
| **ОЦЕНИВАНИЕ ЛИНЕЙНЫХ РЕГРЕССИОННЫХ МОДЕЛЕЙ В УСЛОВИЯХ ГЕТЕРОСКЕДАСТИЧНОСТИ ВОЗМУЩЕНИЙ** | | |
|  | Факультет: | ПМИ |
| Группа: | ПМИ-12 |
| Бригада: | 10 |
| Студенты: | Швадченко Артём Cубботин Дмитрий |
|  |
| Преподаватель: | Попов Александр Александрович |
|  | | |
| Новосибирск | | |
|  | | |
| 2024 | | |

1. **Цель:**

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 – возрастающая функция от двух факторов(взвешенная сумма квадратов факторов).

1. **Ход работы:**

*f*(*x*) = (1*, x*1*, x*2*, x*3*, x*21*, x*22*, x*23*, x*1*x*2*, x*1*x*3*, x*2*x*3)*Т*

*θ* = (1*,* 1*,* 1*,* 1*,* 0*.*01*,* 0*.*01*,* 0*.*01*,* 1*,* 1*,* 1)*Т*

*xi ∈* [*−*1*,* 1]

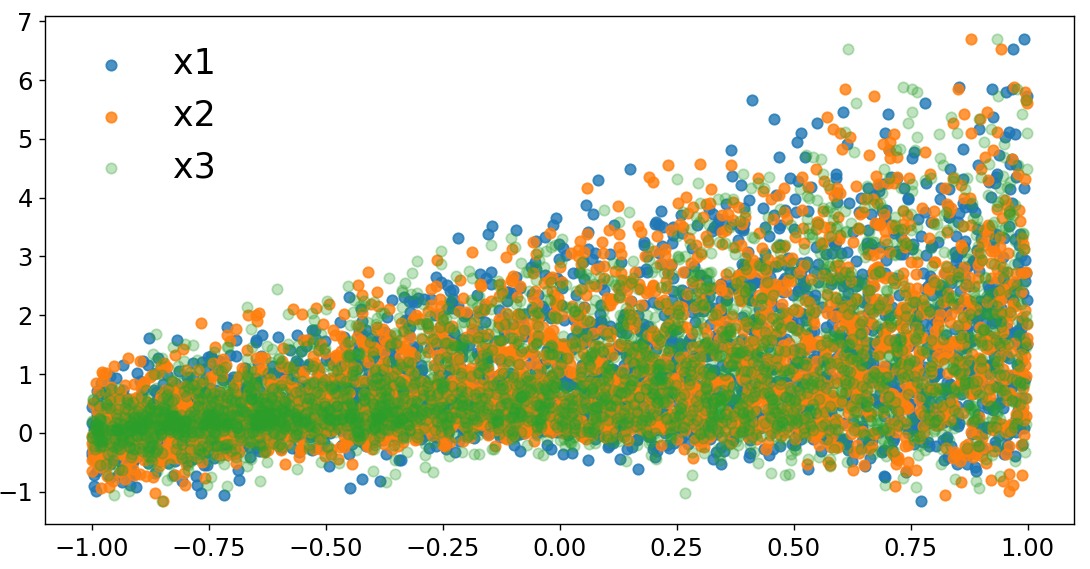
*i* = 1*,* 2*,* 3

*ϵ ∈ N*(0*, σ*2)

*σ*2 = 0.1x1^2+0.3x2^2

*n* = 3000

Визуализация зависимости значений модели от значений её параметров:



Наблюдается гетероскедастичность.

**3 Проверка гетероскедастичности**

**3.1 Тест Бреуша-Пагана**

*zТt* = (1*,* 0.1x1^2+0.3x2^2)

*αT* = (*α*0*, α*1)

Гипотеза о гомоскедастичности:

*α*1 = 0

Оценим исходное уравнение и дисперсию (ОМП):

*et* = *yt − f*(*xt*)*T* ˆ*θ*

Построим регрессию с откликом:

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

Гипотеза о гомоскедантичности принимается, если

Получаем:

= 2403.14

Гипотеза отвергается.

**3.2 Тест Голдфельда-Квандтона**

Предположим, что источник нарушения гомоскедастичности взят в форме:

Упорядочим последовательность наблюдений в соответствии с величиной отклика.

Опустим

*nc* = *n/*3 = 1000

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

наблюдений.

Гипотеза о гомоскедантичности принимается, если

Получаем:

Гипотеза отвергается.

**4. Оценивание параметров модели**

Т

Т

*θ* = (1*,* 1*,* 1*,* 1*,* 0*.*01*,* 0*.*01*,* 0*.*01*,* 1*,* 1*,* 1)Т

Сравним эффективность оценок по квадрату их расстояния до истинных значений:

*RLS* = (*θ − θLS*)*T* (*θ − θLS*) = 0*.*0016

*RGLS* = (*θ − θGLS*)*T* (*θ − θGLS*) = 0*.*0016

Доступный обобщенный МНК примерно такой же по эффективности как и обыкновенный.

**4. Код программы**

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\*x1\*\*2

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)

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]

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)