```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression


data = pd.read_csv('/content/Семинар 5.csv', sep = ';')
data.head()
```

	n	y	x
0	1	310	390
1	2	350	385
2	3	330	545
3	4	425	680
4	5	502	810

```
reg = LinearRegression().fit(data['x'].values.reshape(-1,1), data['y'].values.reshape(-1,1


epsilons = data['y'].values.reshape(-1,1) - reg.predict(data['x'].values.reshape(-1, 1))


mean_e = epsilons.mean()
std_e = epsilons.std()

vec_w = (-1) * (epsilons - mean_e) / std_e
vec_w[:5]
```

```
array([[-0.37668623],
       [-1.46225638],
       [ 0.7806929 ],
       [-0.21206563],
       [-0.79457912]])
```

```
vec_w = sorted(vec_w)


import scipy.stats as scp

new_w = [round(scp.norm.cdf(vec_w[i])[0], 3) for i in range(len(vec_w))]
new_w
```

```
[0.072,
 0.127,
 0.129,
 0.156,
 0.185,
 0.21,
 0.213,
 0.246,
 0.247,
 0.351,
```

```
        0.353,
        0.38,
        0.389,
        0.416,
        0.561,
        0.588,
        0.693,
        0.783,
        0.816,
        0.865,
        0.878,
        0.897,
        0.937,
        0.995]
```

```
i = np.linspace(0., 1., len(new_w))
k = 0
```

```
for cur in range(len(new_w) - 1):
  k += (((new_w[cur] > i[cur]) & (new_w[cur] < i[cur + 1])) == 0)

k, len(new_w)
```

```
     (17, 24)
```

Задание 5.1

Задача (с использованием ПК):

Проверьте выполнение предпосылки о гомоскедастичности остатков модели, построенной на семинаре 1, используя:

a) графический анализ остатков,
b) тест ранговой корреляции Спирмена,
c) тест Парка,
d) тест Глейзера,
e) тест Уайта,
f) тест Бреуша- Пагана,
g) тест Голдфелда – Квандта

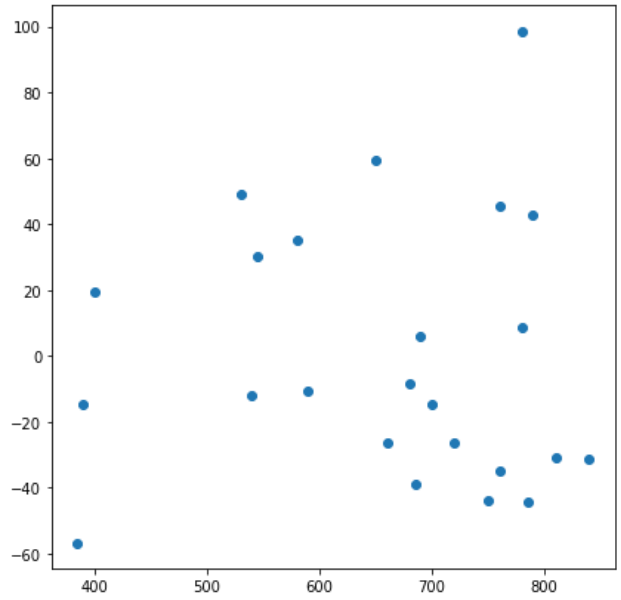
Задача семинар 1.1.

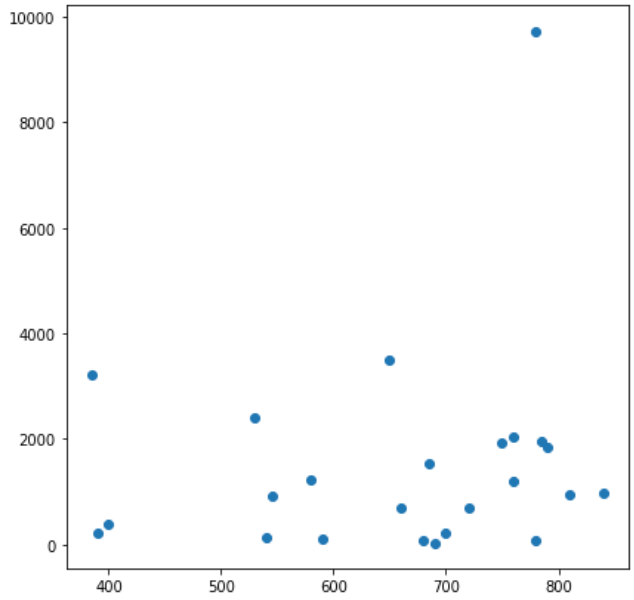
Гетероскедастичность

Графический анализ остатков

```
import matplotlib.pyplot as plt

fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (15, 7))
ax1.scatter(y = -epsilons, x = data['x'].values)
ax2.scatter(y = epsilons ** 2, x = data['x'].values)
plt.show()
```





тест ранговой корреляции Спирмена

```
rho, p = scp.spearmanr(data['x'].values, abs(epsilons))

rho, p
```

```
(0.13701610691564495, 0.5231951496094867)
```

1-й шаг. Построение уравнения регрессии и расчет отклонений $u_i \ \left(i = \overline{1,n}\right)$

$$u_i = y_i - \hat{a}_0 - \hat{a}_1 x_1 - \ldots - \hat{a}_m x_m .$$

2-й шаг. Использование значений отклонений для получения новой зависимой переменной. В качестве последней используются логарифмы квадратов отклонений. Независимая переменная – фактор пропорциональности (наиболее существенно влияющая переменная на основную переменную). Построение вторичного уравнения регрессии

$$\ln\left(u_i^2\right) = \alpha_0 + \alpha_1 \ln Z_i + \varepsilon_i .$$

В качестве фактора пропорциональности часто выбирают ту независимую переменную, которая имеет дисперсию, близкую к дисперсии ошибок.

3-й шаг. Проверка значимости коэффициента регрессии при Z с помощью критерия Стьюдента. Если коэффициент существенно отличается от нуля, то это является очевидным признаком наличия гетероскедастичности в отклонениях по отношению к независимой переменной Z; в противном случае, гетероскедастичность, относящаяся к данному конкретному Z, не подкрепляется с очевидностью в данных отклонениях. Тем не менее, невозможно полностью доказать, что ошибка уравнения регрессии обладает свойством гомоскедастичности.

тест Парка

```
z = np.log(data['x'].values)
```

```
import statsmodels.api as sm

results = sm.OLS.from_formula("np.log(epsilons ** 2) ~ np.log(x)", data = data).fit()
```

```
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     np.log(epsilons ** 2)   R-squared:                       0.013
Model:                               OLS   Adj. R-squared:                 -0.032
Method:                    Least Squares   F-statistic:                    0.2973
Date:                   Sat, 26 Nov 2022   Prob (F-statistic):              0.591
Time:                           11:07:41   Log-Likelihood:                -42.020
No. Observations:                     24   AIC:                             88.04
Df Residuals:                         22   BIC:                             90.40
Df Model:                              1
Covariance Type:               nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
```

```
Intercept      1.9529      8.438      0.231      0.819     -15.546      19.452
np.log(x)      0.7111      1.304      0.545      0.591      -1.994       3.416
==============================================================================
Omnibus:                        1.412   Durbin-Watson:                   1.877
Prob(Omnibus):                  0.494   Jarque-Bera (JB):                1.240
Skew:                          -0.505   Prob(JB):                        0.538
Kurtosis:                       2.530   Cond. No.                         188.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
```

```
t = scp.t.isf(0.05 / 2, 22)
t, 0.545
```

```
(2.073873067904015, 0.545)
```

t_tabl > t_fact, гипотеза H0 принимается(т-теста) -> параметр не значим -> гипотеза H0 принимается -> гетероскедастичность отсутствует

тест Глейзера

```
result = [0, 1, 2, 3, 4, 5]
model = ['x', 'np.sqrt(x)', 'np.abs(1/x)', 'np.abs(1 / np.sqrt(x))', 'np.sqrt(x ** 3)']

result[1] = sm.OLS.from_formula("np.abs(epsilons) ~ x", data = data).fit().tvalues['x']
result[2] = sm.OLS.from_formula("np.abs(epsilons) ~ np.sqrt(x)", data = data).fit().tvalue
result[3] = sm.OLS.from_formula("np.abs(epsilons) ~ np.abs(1/x)", data = data).fit().tvalu
result[4] = sm.OLS.from_formula("np.abs(epsilons) ~ np.abs(1 / np.sqrt(x))", data = data).
result[5] = sm.OLS.from_formula("np.abs(epsilons) ~ np.sqrt(x ** 3)", data = data).fit().t
```

```
for i in range(5):
  print(f'Model {model[i]} t-stats : {result[i + 1]}')
```

```
Model x t-stats : 0.7935388284900962
Model np.sqrt(x) t-stats : 0.7432130602685529
Model np.abs(1/x) t-stats : -0.5824717762405766
Model np.abs(1 / np.sqrt(x)) t-stats : -0.6365793767746863
Model np.sqrt(x ** 3) t-stats : 0.840605983252266
```

все t-значения меньше табличного => гипотеза H0 принимается и присутствует гомоскедастичность

```
new_x = np.concatenate((np.ones((len(data['y']), 1)), data['x'].values.reshape(-1, 1)), ax
new_x
```

```
array([[  1., 390.],
       [  1., 385.],
       [  1., 545.],
       [  1., 680.],
       [  1., 810.],
       [  1., 780.],
       [  1., 790.],
       [  1., 785.],
       [  1., 400.],
       [  1., 530.],
       [  1., 580.],
       [  1., 720.],
       [  1., 700.],
       [  1., 690.],
       [  1., 650.],
       [  1., 760.],
       [  1., 780.],
       [  1., 840.],
       [  1., 590.],
       [  1., 540.],
       [  1., 660.],
       [  1., 685.],
       [  1., 750.],
       [  1., 760.]])
```

Тест Голфреда-Кванта

```
from statsmodels.stats import api
api.het_goldfeldquandt(epsilons, new_x)
```

```
(0.5174470920501953, 0.8431082250887834, 'increasing')
```

Тест Уайта

тест работает и используется в дальнейших работах

```
import statsmodels.api as sm
fin_model = sm.OLS.from_formula("y ~ x", data = data)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-6474e115f2ba> in <module>
      1 import statsmodels.api as sm
----> 2 fin_model = sm.OLS.from_formula("y ~ x", data = data)

NameError: name 'data' is not defined
```

SEARCH STACK OVERFLOW

```
from statsmodels.stats.diagnostic import het_white
```

```
#perform White's test
white_test = het_white(fin_model.fit().resid, fin_model.fit().model.exog)

#define labels to use for output of White's test
labels = ['Test Statistic', 'Test Statistic p-value', 'F-Statistic', 'F-Test p-value']

#print results of White's test
print(dict(zip(labels, white_test))['Test Statistic p-value'])
```


2 сек. выполнено в 13:38