Оглавление

[Различные фишки для работы с датасетом 2](#_Toc134921489)

[1. Импорт библиотек 3](#_Toc134921490)

[2. Импорт датасета + подгонка прямой 4](#_Toc134921491)

[3. Спецификация + интерпретация модели 6](#_Toc134921492)

[4. Анализ данных, записанных в строку 7](#_Toc134921493)

[5. Т - тест 9](#_Toc134921494)

[6. F - тест 10](#_Toc134921495)

[7. Структурные сдвиги 13](#_Toc134921496)

[8. Доверительный интервал 14](#_Toc134921497)

[9. Прогнозирование 15](#_Toc134921498)

[10. Мультиколлинеарность 17](#_Toc134921499)

[11. RESET - тест 18](#_Toc134921500)

[12. Гетероскедастичность + состоятельные оценки стандартных ошибок регрессии при условной гетероскедастичности 19](#_Toc134921501)

[13. Автокорреляция DW - тест 20](#_Toc134921502)

# Различные фишки для работы с датасетом

***#Изменение коэффициентов в датасете***

def tran\_sex(x):

if x == 'male':

return 1

if x == 'female':

return 0

df['sex'] = df['sex'].apply(tran\_sex)

df

***#Или так:***

df[‘gender’]=df.apply(lambda x: 1 if x[‘sex’] == ‘male’ else 0, axis=1)

***#Изменение названия колонки***

df = df.rename(columns={'старое название' : 'новое название'})

***# Переименуем столбцы, так как log(output) интерпреитуется как функция логарифма от output***

df = df.rename(columns={"log(output)": "LOGoutput", "log(capital)": "LOGcapital", "log(labour)":'LOGlabour'})

df

***# Замена запятой в датасете на точку:***

df[**'**Temp\_M2'] = df['Temp\_M2'].str.replace(',', '.')

df

***# Превращение типа "object" в "float"***

df['Temp\_M2'] = pd.to\_numeric(df['Temp\_M2'])

df['Temp\_Infl'] = df['Temp\_Infl'].astype(float)

df

***df.dtypes #проверка***

***# удаление колонки в таблице***

df.drop(columns='totwrk', inplace = True )

***#конвертирование полученных значений в табличку через конструктор***

pd.DataFrame ({ 'pred' : pred.values})

# 1. Импорт библиотек

import scipy.stats as stats   
import statsmodels.stats.api as sms  
import numpy *a*s np#linear algebra   
import pandas as pd #data processing, CSV file I/O (e.g. pd.read\_csv)   
import matplotlib.pyplot as plt #интерактивная визуализация   
import statsmodels.api as sm # regression fitting via matrices of regression design   
import statsmodels.formula.api as smf # data visualization, regression fitting   
from statsmodels.iolib.summary2 import summary\_params # вывод результатов тестирования   
from statsmodels.iolib.summary2 import summary\_col # вывод результатов тестирования   
from statsmodels.stats.outliers\_influence import variance\_inflation\_factor # VIF   
import io #преобразование строк в данные  
import seaborn as sns   
import re #библиотека для работы с регулярными выражениями

import statsmodels.stats.diagnostic as dg

# 2. Импорт датасета + подгонка прямой

**df** = pd.read\_csv('ссылка')

df

***Визуализация***

**sn**s.scatterplot( data = df , y = 'sleep' , x = 'totwrk' )

***Изображение "подогнанной" прямой***

sns**.**regplot(data = df , y = 'sleep' , x = 'totwrk' , ci = None , line\_kws = { "color":"r"})

print('Подогнанная прямая')

***# подгонка прямой***

fitted\_line = smf.ols( formula = 'sleep~totwrk', data = df ).fit()

# коэффициенты с округлением

fitted\_line.params.round(2)

# подгонка прямой, (-1) - чтобы убрать коэффициент b0

fitted\_line = smf.ols( formula = 'sleep~-1+(totwrk)' , data = df ).fit()

#подгонка плоскости

fitted\_plane = smf.ols(formula = 'sleep~totwrk+age' , data = df ).fit()

fitted\_plane.params.round(2)

***Изображение "подогнанной" пораболы***

sns**.**regplot(data = df , y = 'sleep' , x = 'totwrk' , order = 2 , ci = None , line\_kws = {"color" : "r"})

# подгонка параболы

fitted\_pol = smf.ols(formula = 'sleep~totwrk+I(totwrk\*\*2)' , data = df).fit()

# коэффициенты с округлением

fitted\_pol.params.round(2)

***Изображение "подогнанной" прямой под логарифмы***

**sns.**regplot(x = np.log(df\_lab [ 'capital' ]), y = np.log(df\_lab ['output']), order = 1, ci = None, line\_kws = {"color" : "r" })

***#подгонка прямой, I - в какую степень возвести логарифм***

fitted\_polynom = smf.ols(formula = 'np.log(output)~np.log(capital)', data = df\_lab).fit()

***# коэффициенты с округлением***

fitted\_polynom.params.round(2)

***Изображение "подогнанной" пораболы под логарифмы***

sns.regplot(x = np.log(df\_lab [ 'capital' ]), y = np.log(df\_lab ['output']), order = 2, ci = None, line\_kws = {"color" : "r" })

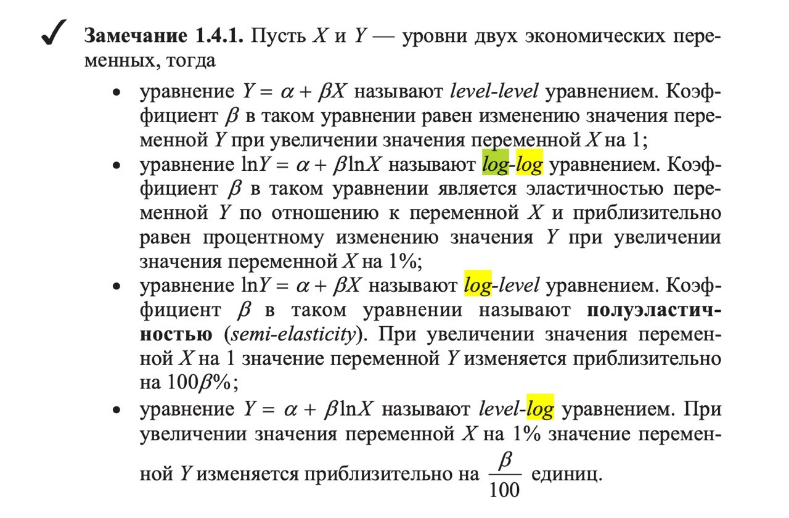
# *подгонка пораболы, I - в какую степень возвести логарифм*

fitted\_polynom = smf.ols(formula = 'np.log(output)~np.log(capital)+I(np.log(capital)\*\*2)' , data = df\_lab).fit()

# коэффициенты с округлением

fitted\_polynom.params.round(2)

# 3. Спецификация + интерпретация модели



## Спецификация модели:

$$

sleep = \beta\_0 + \beta\_1\*totwrk + \beta\_2\*male

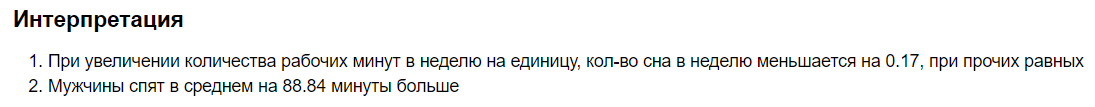
$$

***#обобщающенная информация о полученном решении***

sleep\_eq1.summary()

***#обобщающая инфа в колонке***

summary\_col(sleep\_eq1, stars=None)



#спецификация с логарифмом

$$

sleep=\beta\_0+\beta\_1\*totwrk+\beta\_2\*male+\beta\_3\*smsa+\beta\_4\*age+\beta\_5\*south+\beta\_6\*yngkid+\beta\_7\*marr + \beta\_8\*log(hrwage)$$

# 4. Анализ данных, записанных в строку

***#определяем данные***

data = '''sleep totwrk age male hrwage 3113 3438 32 1 7.07 2920 5020 31 1 1.43 2670 2815 44 1 20.53 3083 3786 30 0 9.62 3448 2580 64 1 2.75 4063 1205 41 1 19.25'''

***# создаём DataFrame***

df = pd.read\_csv(io.StringIO(data), sep='\s+') df

# специфицируем модель через формулу

output\_eq1 = smf.ols(formula='sleep~totwrk+age+male+np.log(hrwage)', data=df).fit()

# Коэфициенты модели с округление

output\_eq1.params.round(2)

***#добавляем доп. столбец с предсказанными (fitted) значениями*** df['pred'] = 3525.14 + (-0.16)\*df['totwrk'] + 1.62\*df['age'] + 51.84\*df['male'] + (-9.88)\*np.log(df['hrwage'])

df

**#добавляем столбец с остатками (ei)**

df['residual'] = df['sleep'] - df['pred']

***#Total Sum Square формула через среднее***

TSS = ((df['sleep/60']-np.mean(df['sleep/60']))\*\*2).sum()

TSS.round(3)

***#Или так:***

fitted.centered\_tss

***#Residual Sum Square формула***

RSS = ((df['Факт']-df['Fitted'])\*\*2).sum()

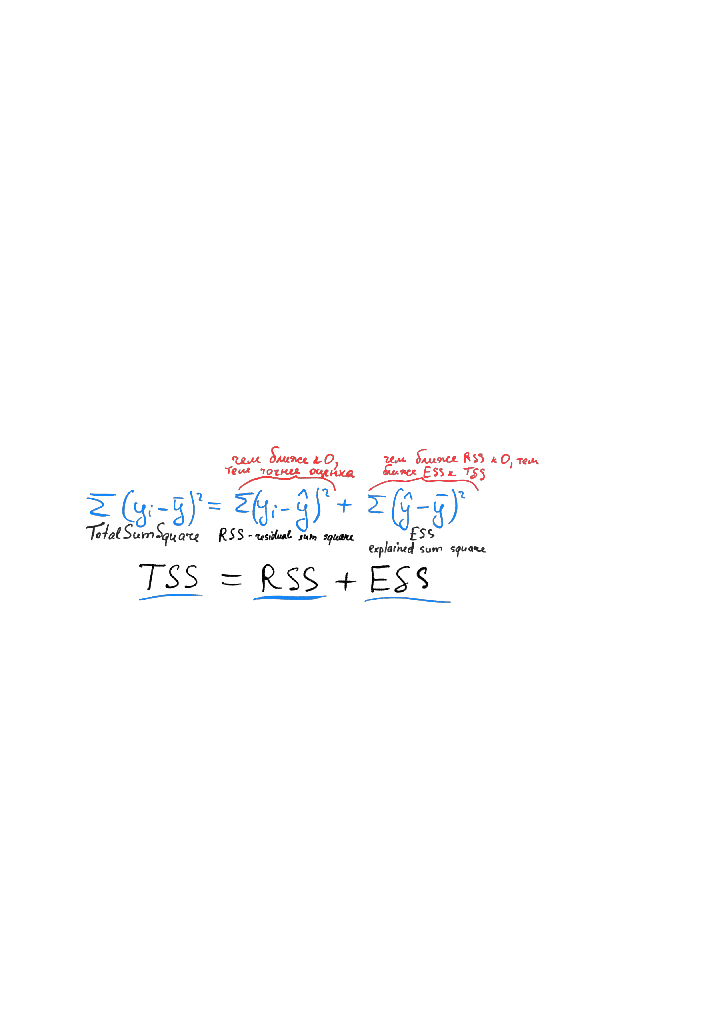
RSS.round(3)

***#Или так:***

fitted.ssr.round(3)

***#Estimated Sum Square формула***

ESS = ((df['pred']-np.mean(df['Infl']))\*\*2).sum()

ESS.round(3)

***#стандартное отклонение***

n = 10 #кол-во наблюдений

k = 2 #кол-во регрессоров

s = (RSS/(n-k-1))\*\*0.5

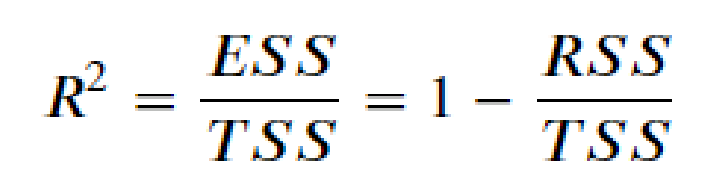
round(s,2)

***#Запись R2***

$$

R^2 = \frac{ESS}{TSS} = 1-\frac{RSS}{TSS}

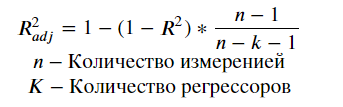
$$



R2 = fitted\_line.rsquared # коэффициент детерминации

***#Запись R2adj***

$$ R^2\_{adj} = 1 - (1-R^2)\*\frac{n-1}{n-k-1} $$ $$ n - Количество \ измеренией $$ $$ K - Количество \ регрессоров $$



n = 526

k = 5

R2 = 0.288

R2adj = 1 - (1-R2)\*(n-1)/(n-k-1)

***#стандартное отклонение***

n = 10 #кол-во наблюдений

k = 2 #кол-во регрессоров

s = (RSS/(n-k-1))\*\*0.5

round(s,2)

***#Подгонка прямой + саммари по ней***  
model = smf.ols(data=df\_labour, formula='np.log(output)~np.log(capital)+np.log(labour)').fit()  
model.summary(alpha=0.01)  
model.summary(alpha=0.01).tables[1] #оставляем только одну табличку от общего саммари

# 5. Т - тест

***#t наблюдаемое при гипотезе beta=0***

df['znachimost'] = np.abs(df[ 'Estimate' ]/df [ 'Std.Error' ])

df

***#t набл для любой гипотезы***

fitted\_line.t\_test('M2\_Log=0, M2=0')

***#t критическое***

alpha = 0.05 #ppf - percent point function t = stats.t.ppf(1-alpha/2, n-k-1) t

***#Подгонка прямой + саммари по ней***

model = smf.ols(data=df\_labour, formula='np.log(output)~np.log(capital)+np.log(labour)').fit()

model.summary(alpha=0.01)

model.summary(alpha=0.01).tables[1] #оставляем только одну табличку от общего саммари

***#Создаю столбец со значимостью регрессоров (при гипотезе b=0)***

df['result'] = df.apply(lambda x: 'значимо' if abs(x['znachimost'])>t\_cr else 'не значимо',axis=1)

df

***#t наблюдаемое при гипотезе beta=a***

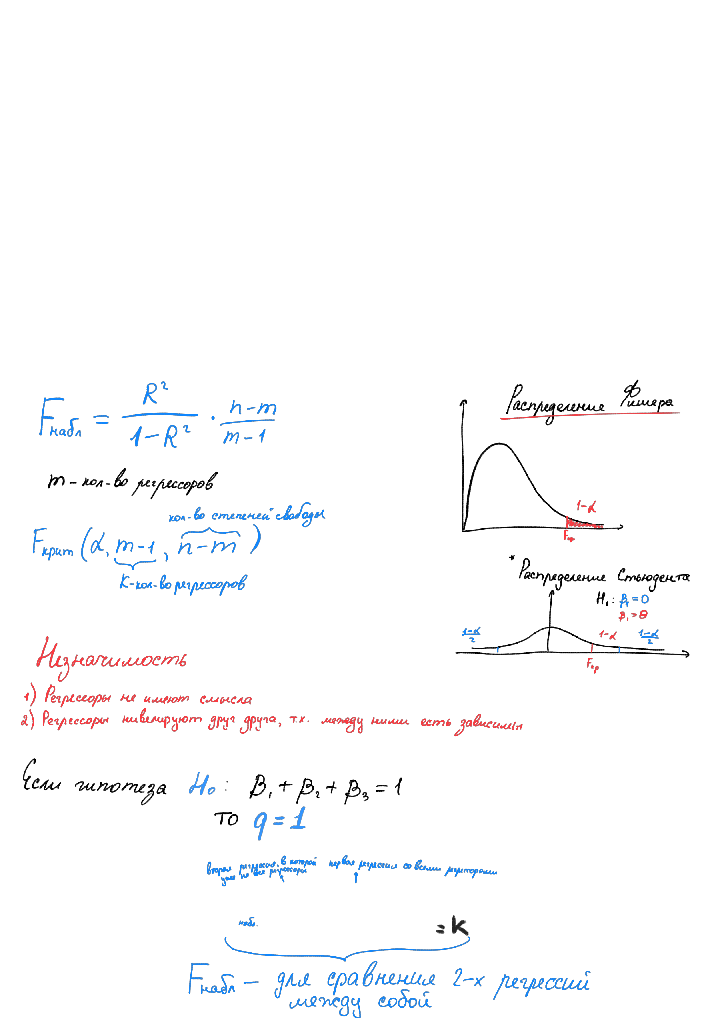
fitted\_line.t\_test('M2\_Log=a')

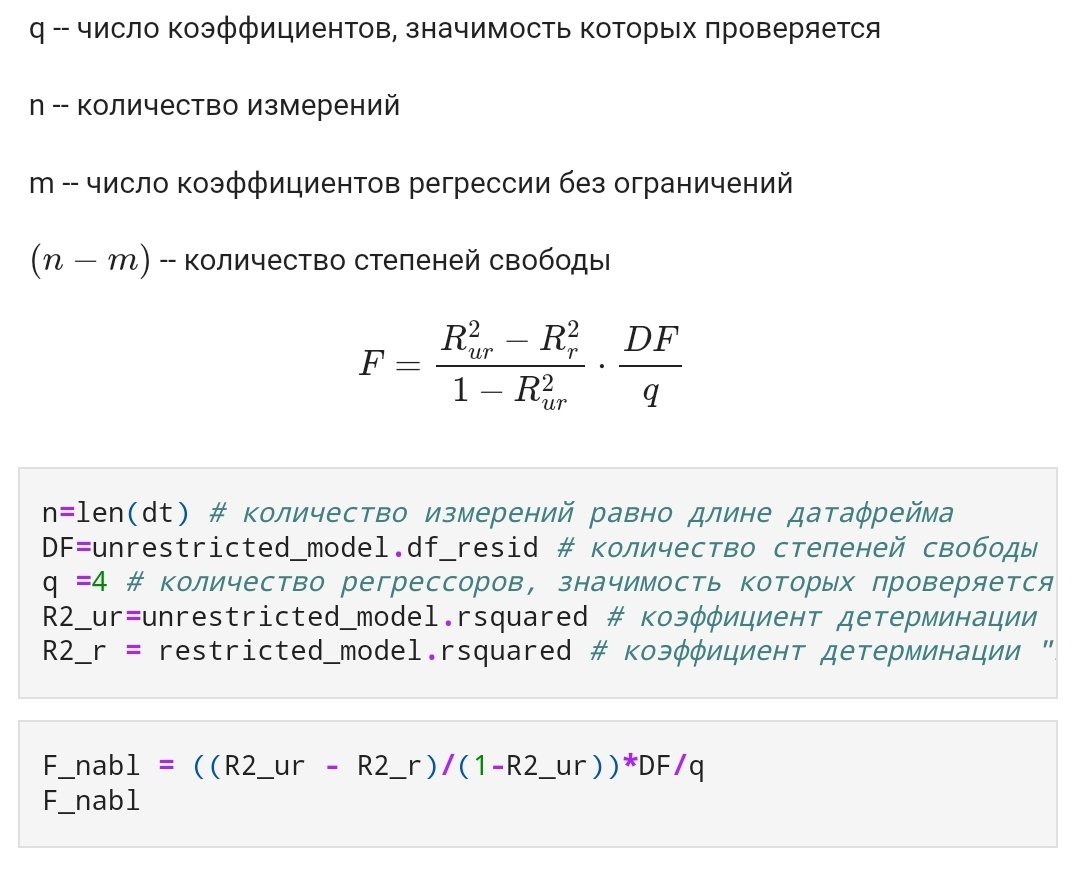
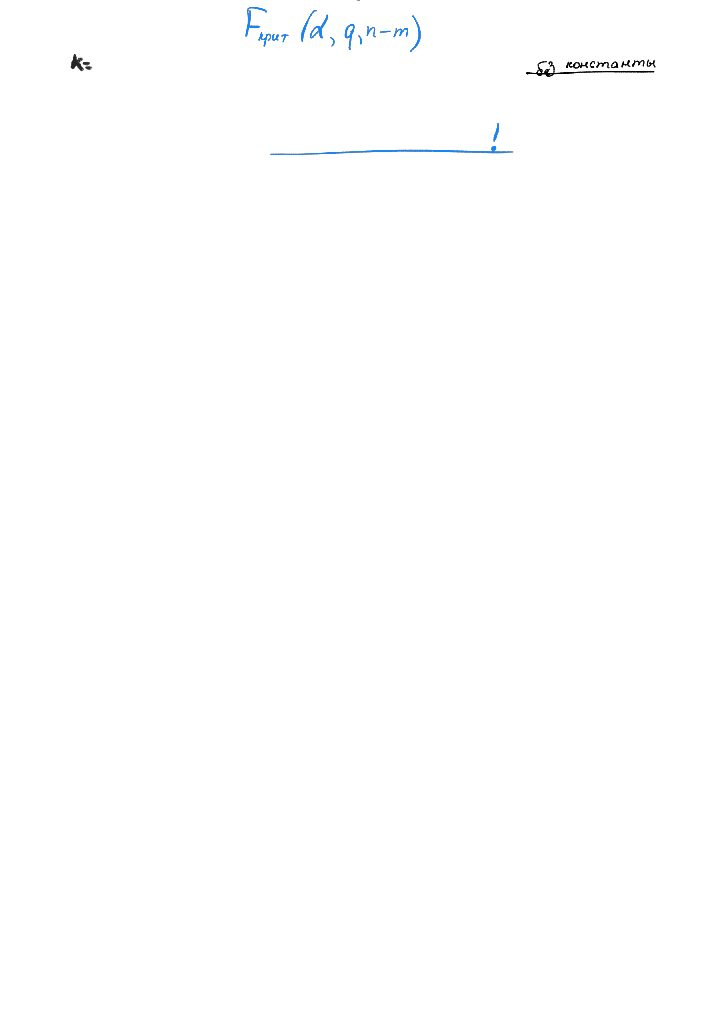
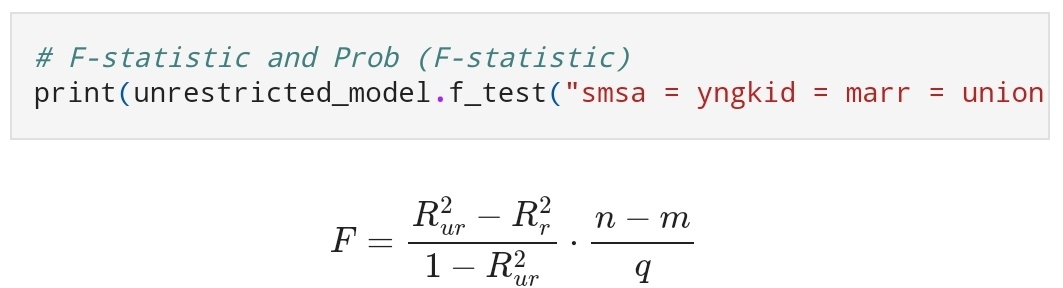
t\_nabl = (beta-a)/st\_err

t\_cr = stats.t.ppf(1-alpha/2,569-2-1)

print('Коэффициент beta незначимо отличается от значения а') if abs(t\_nabl)<t\_cr else print('Коэффициент beta значительно отличается от значения a')

# 6. F - тест



**Предположение, что все регрессоры незначимы:**

***# подгоним модель***

fitted = smf.ols (formula = 'sleep~smsa+yngkid+marr+union', data = df\_sleep).fit()

***# Результаты F-теста на значимость регрессии (тестовая статистика и P-значение)***

fitted.fvalue

***#F критическое:***

F\_cr = stats.f.ppf(1-alpha, q, n-m)

F\_cr

data\_string = '''Observations 706 706 706 706

R2 0.132 0.131 0.128 0.007

AdjustedR2 0.121 0.123 0.121 0.002 '''

***#header нужен для того, чтобы первая строка не обозначалась как наименования***

df = pd.read\_csv(io.StringIO(data\_string),

delimiter='\s+', header=None, index\_col=0)

df.reset\_index() #сдвиг названий столбцов вниз

***#транспонируем матрицу***

df = df.transpose()

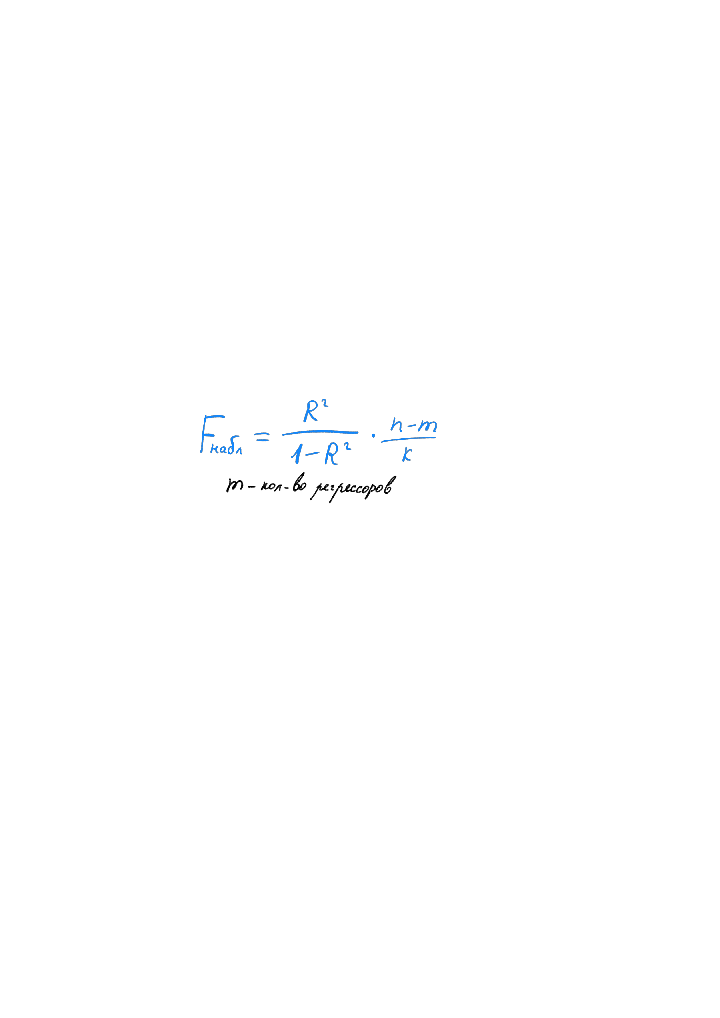
df

***# Заполним столбец значением m - кол-во регрессоров для каждой модели***

df['m'] = [10,8,6,5]

df

***#создаем столбец со значениями Fнабл***

df['F\_nab'] = (df['R2']/(1-df['R2']))\*((df['Observations']-df['m'])/(df['m']-1))

df

***#F критическое: (df['m']-1) - кол-во регрессоров без константы, (df['Observations']-df['m']) - кол-во степеней свободы***

alpha = 0.05

df['F\_cr'] = stats.f.ppf(1-alpha,df['m']-1,df['Observations'] - df['m'])

df

***#Создаю столбец со значимостью коэффициентов***

df['result'] = df.apply(lambda x: 'регрессоры значимы' if abs(x['F\_nab'])<abs(x['F\_cr']) else 'регрессоры не значимы', axis=1)

df

**Определение значимости группы регрессоров**

$$

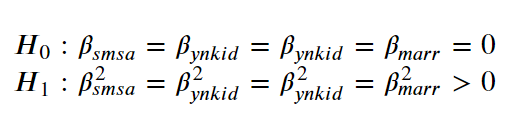
H\_0 : \beta\_{smsa} = \beta\_{ynkid} = \beta\_{ynkid} = \beta\_{marr} = 0

$$

$$

H\_1 : \beta\_{smsa}^2 = \beta\_{ynkid}^2 = \beta\_{ynkid}^2 = \beta\_{marr}^2 > 0

$$



***# подгоним модель***

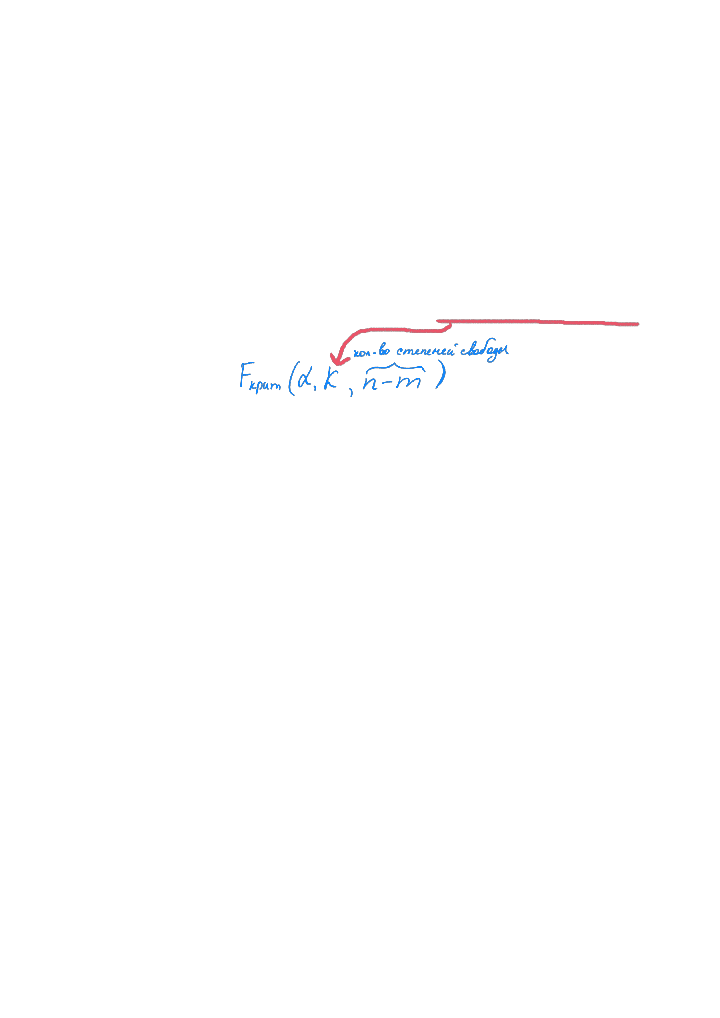
fitted\_unrest = smf.ols(formula = 'sleep~smsa+yngkid+marr+union', data = df\_sleep).fit()

***#явно специфицируем тестируемую гипотезу***

f\_test = fitted\_unrest.f\_test( 'smsa=0, yngkid=0, marr=0, union=0' )

f\_test.fvalue ***#будет показано только Fнабл***.

***#F критическое:***

F\_cr = stats.f.ppf(1-alpha, k - кол-во регрессоров, которые проверяются, n-m)

F\_cr

#регулярное выражение, позволяющее разбивать и находить из совокупности значений нужные цифры

unrestricted, restricted = np.float\_(re.findall(r'([-+]?\d+.\d+)','R2 0.131 0.127'))

print(unrest,rest)

***#подсчет R2 restricted + unrestricted***

R2\_ur = fitted\_unrestricted.rsquared # коэффициент детерминации "длинной регрессии"

R2\_r = fitted\_restricted.rsquared # коэффициент детерминации "короткой регресси"

***#F observated для гипотезы о регресорах***

F\_obs = (R2\_ur - R2\_r)/(1-R2\_ur)\*(n-m)/q

F\_obs

***#интерпретация результатов***

$F\_{obs} < F\_{cr}$ данные согласуются с гипотезой $H\_0$

# 7. Структурные сдвиги

***# Отфильтруем датафрейм по гендерному признаку***

df\_male = df[df [ 'male' ] == 1 ] # Датафрейм для мужчин

df\_female = df [ df [ 'male' ] == 0 ] # Датафрейм дла женщин

***#Через имя переменной***

d1 =df[df['sex']=='male']

d0 = df[df['sex']=='female']

d1

***# ci-доверительный интервал***

***# hue - сортировка по определенной переменной***

sns.lmplot(data=df, x='totwrk', y='sleep', ci=None, hue='male')

slp = smf.ols(data= sleep\_df, formula='sleep60~totwrk+age+I(age\*\*2)+south +smsa +marr').fit()

slp\_ml = smf.ols(data= sleep\_male, formula='sleep60~totwrk+age+I(age\*\*2) +south +smsa +marr').fit() slp\_fml = smf.ols(data = sleep\_female, formula='sleep60~totwrk+age+I(age\*\*2) +south +smsa +marr').fit()

***# F-наблюдаемое***

F\_obs = (slp.ssr - (slp\_ml.ssr+slp\_fml.ssr))/(slp\_ml.ssr+slp\_fml.ssr) \* (len(sleep\_male)+len(sleep\_female)-2\*7)/(7)

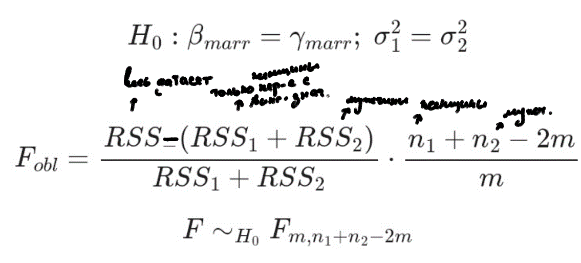
F\_obs

***#Гипотеза:***

$$

H\_0 : \beta\_{married} = \beta\_{notmarried}; \ \sigma\_1^2 = \sigma\_2^2

$$



**F\_cr** = stats.f.ppf(1-alpha, m, n1+n2-2\*m)

***#подгонка модели с учетом структурных сдвигов у остальных регрессоров***

slp = smf.ols(data= sleep\_df, formula='np.log(wage) ~ age+IQ+urban+married+south+age:south+IQ:south+**married:south**+**urban:south')**.fit()

df=pd.read\_csv('https://raw.githubusercontent.com/artamonoff/Econometrica/master/python-notebooks/data-csv/Diamond.csv')

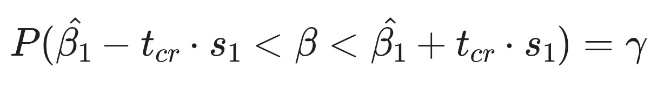
***# Метод .loc[] фильтрует данные таблицы по строкам и столбцам***

***# Метод "df.loc[:,['carat']]" делает срез данных и означает, что из датафрейма выбираются все строки (при помощи знака ":") столбца "carat"***

df\_all = pd.get\_dummies(df.loc[:,['carat','colour','certification','price']])

df\_all

# 8. Доверительный интервал



$$ P(\hat{\beta\_1} - t\_{cr}\cdot s\_1 < \beta < \hat{\beta\_1} + t\_{cr} \cdot s\_1) = \gamma $$

#добавляем данные

data\_string = '''totwrk -0.169\*\*\*

(0.018)

Constant 3450.913\*\*\*

(80.726) '''

*# Строку с данными можно разбить на 8 подстрок в 3 мя элементами в каждой, разделённых различными количеством пробелом.*

*# С этим справится метод reshape() из библиотеки numpy*

*# strip - удаляет из строки все ненужные элементы*

*# split - размещает отдельно по отдельным строкам*

np.reshape(data\_string.strip().split(), (8,3))

*#Делаем из этого всего табличку*

df = pd.DataFrame(np.reshape(data\_string.strip().split(), (8,3)), columns=['Coeff','Est str','std.err. str'])

df

***# оценки. str(- строка).extract( - извлекает), astype - преобразование в число, r' ' - регулярные выражения***

df['Estimate\_numb'] = df['Est str'].str.extract(r'([-+]?\d+.\d+)').astype(float)

***# ошибки***

df['Std.error'] = df['std.err. str'].str.extract(r'([-+]?\d+.\d+)').astype(float)

df

***#t critical***

alpha = 0.1

t\_cr=stats.t.ppf(1-alpha/2, 706-8)

t\_cr

***#формируем доверительный интервал***

df['low']=(df['Estimate\_numb']-t\_cr\*df['Std.error']).round(2)

df['up']=(df['Estimate\_numb']+t\_cr\*df['Std.error']).round(2)

df

***#Подгонка прямой + саммари по ней***  
model = smf.ols(data=df\_labour, formula='np.log(output)~np.log(capital)+np.log(labour)').fit()  
model.summary(alpha=0.01)  
model.summary(alpha=0.01).tables[1] #оставляем только одну табличку от общего саммари

# 9. Прогнозирование

data = '''totwrk -0.169\*\*\* (0.018)

age 2.689\*

(1.469)

south 101.568\*\*

(41.837)"'

***# извлекаем цифры***

data\_num = re.findall(r'([-+]?\d+.\d+)',data)

data\_num

***# создаем матрицу***

# 8 --- количество строк

# 2 --- кол-во столбцов

beta = np.reshape(data\_num,(8,2))

beta

***# задаем другой формат числам***

beta = beta.astype(float)

beta

***#срез ":" - выводим на экран все элементы, которые стоят в нулевом "0" столбце***

# если хотим строку, то [0,:]

# первое число - номер строки, второе число - номер столбца

beta[:,0]

ppl\_string = '''totwrk age south male smsa yngkid marr

1 2150 37 0 1 1 0 1

2 1950 28 1 1 0 1 0

3 2240 26 0 0 1 0 0'''

ppl\_string = re.findall(r'([-+]?\d+)',ppl\_string)

ppl\_string

ppl\_matrix = np.reshape(ppl\_string,(3,8))

ppl\_matrix

***# удаляем некоторые данные из таблицы***

# 1 - работаем в столбце, 0 - номер строки/столбца

matrix = np.delete(matrix, 0,1)

matrix

***# теперь необходимо добавить столбец из единиц для того, чтобы учеть константу*** ppl\_matrix = np.c\_[ppl\_matrix,np.ones(3)] ppl\_matrix

***# умножаем матрицы***

ppl\_matrix@beta[:,0]

***# Либо можно так:***

**sl**eep\_df = pd . read\_csv ( 'https://raw.githubusercontent.com/artamonoff/Econometrica/master/python-notebooks/data-csv/sleep75.csv' )

**mod1**  = smf . ols ( formula = 'sleep~totwrk+age+south+male+smsa+yngkid+marr' , data = sleep\_df ).fit()

mod1.params

***# рассмотрение нескольких людей***

**new\_df** = pd.DataFrame ( { 'totwrk' : [ 2150 , 1950 , 2240 ], 'age' : [ 37 , 28 , 26 ], 'south' : [ 0 , 1 , 0 ], 'male' : [ 1 , 1 , 0 ], 'smsa' : [ 1 , 0 , 1 ], 'yngkid' : [ 0 , 1 , 0 ], 'marr' : [ 1 , 0 , 0 ] } )

new\_df

***# удаление колонки***

df.drop(columns='№', inplace = True )

***# Построение прогноза при помощи .predict()***

pred = mod1.predict(exog = new\_df , transform = True)

***# предсказанные значения, если использовался логарифм***

np.exp(pred).round(2)

***#конвертирование полученных значений в табличку***

# через конструктор

pd.DataFrame ({ 'pred' : pred.values})

# 10. Мультиколлинеарность

df = pd.read\_csv('https://raw.githubusercontent.com/artamonoff/Econometrica/master/python-notebooks/data-csv/sleep75.csv')

mod = smf.ols( formula = 'sleep~totwrk+age+I(age\*\*2)+smsa+south+male', data = df).fit()

mod1 = smf.ols( formula = 'age~totwrk+I(age\*\*2)+smsa+south+male', data = df).fit()

mod2 = smf.ols( formula = 'I(age\*\*2)~totwrk+age++smsa+south+male', data = df).fit()

mod3 = smf.ols( formula = 'male~totwrk+age+I(age\*\*2)+smsa+south', data = df).fit()

VIF1 = 1/(1-mod1.rsquared)

VIF2 = 1/(1-mod2.rsquared)

VIF3 = 1/(1-mod3.rsquared)

VIF1, VIF2, VIF3

***# Матрица корреляций***

df = df[['male', 'age', 'smsa']].copy()

corr\_matrix = df.corr().round(3)

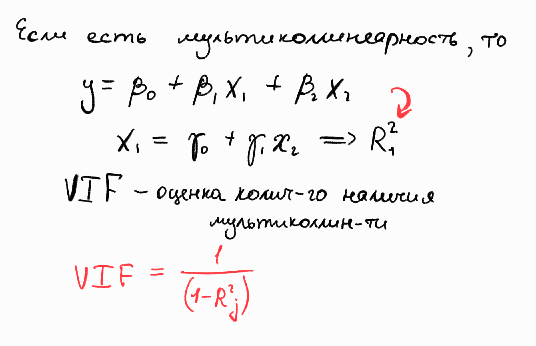
corr\_matrix

# на пересечении находятся коэффициенты корреляции. В регрессии коэффициенты корреляции должны стремиться к 0

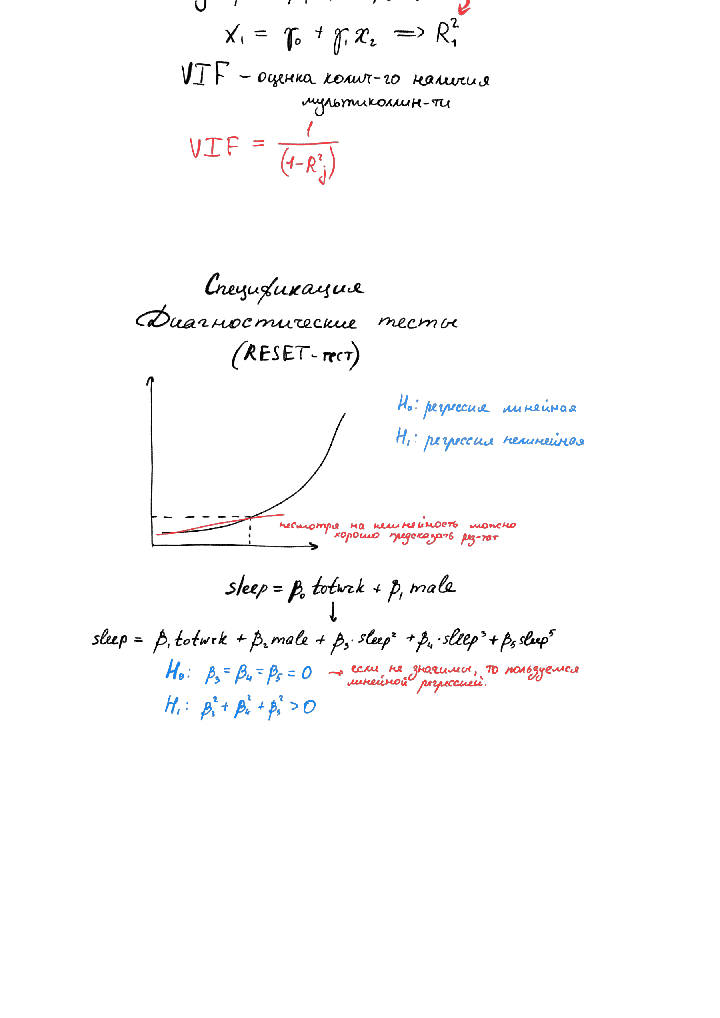
**# Визуализация корреляций**

sm.graphics.plot\_corr(corr\_matrix, xnames=df.columns, normcolor=True, cmap='coolwarm')

plt.show()



# 11. RESET - тест



***#Проверка гипотезы через F-статистику***

df['pred'] = 3470.460185-0.170220\*df['totwrk'] + 2.831414\*df['age'] + 91.257164\*df['male'] -56.759172\*df['smsa'] + 99.508559\*df['south']

df['resid'] = df['sleep']- df['pred']

df['pred2'] = df['pred']\*\*2

df['pred3'] = df['pred']\*\*3

df['pred4'] = df['pred']\*\*4

df

***#F observated для гипотезы о регресорах***

F\_obs = (slp\_unrest.rsquared - slp\_rest.rsquared)/(1-slp\_unrest.rsquared)\*(len(df)-10)/3

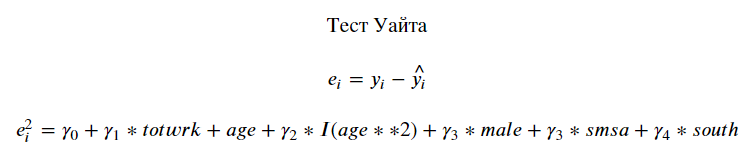
F\_obs.round(2)

\hat{\beta}} ***#добавляем шляпку к спецификации***

# 12. Гетероскедастичность + состоятельные оценки стандартных ошибок регрессии при условной гетероскедастичности

**# квадраты остатков добавим в датафрейм, воспользовавшись методом '.resid', возвращающий значения остатков**  
df['res2']=model.resid\*\*2  
df['res']=model.resid

*Breusch Pagan test*

******  
*Проверка на зависимость остатков от регрессоров*

$$e\_i = y\_i - \hat{y}\_i$$

$$

e\_i^2 = \gamma\_0 + \gamma\_1\*age+\gamma\_2\*I(age\*\*2)+\gamma\_3\*bmi+\gamma\_4\*bmi^2+\gamma\_5\*children+\gamma\_6\*smoker+\gamma\_7\*sex

$$

## Сформулируем проверямемую гипотезу, о характере зависимости дисперсии ошибок от факторов регресси

$$

H\_0: \sigma\_i^2 \equiv \sigma^2=f(\gamma\_0)

$$

$$

H\_1:\sigma\_i^2 = f(\gamma\_0 + z\_{i1}\gamma\_1 + z\_{i2}\*\gamma\_2 + \dots + \gamma\_{ip}\gamma\_p)

$$

### Другими словами, значимость вспомогательной регресси в целом, т.е.

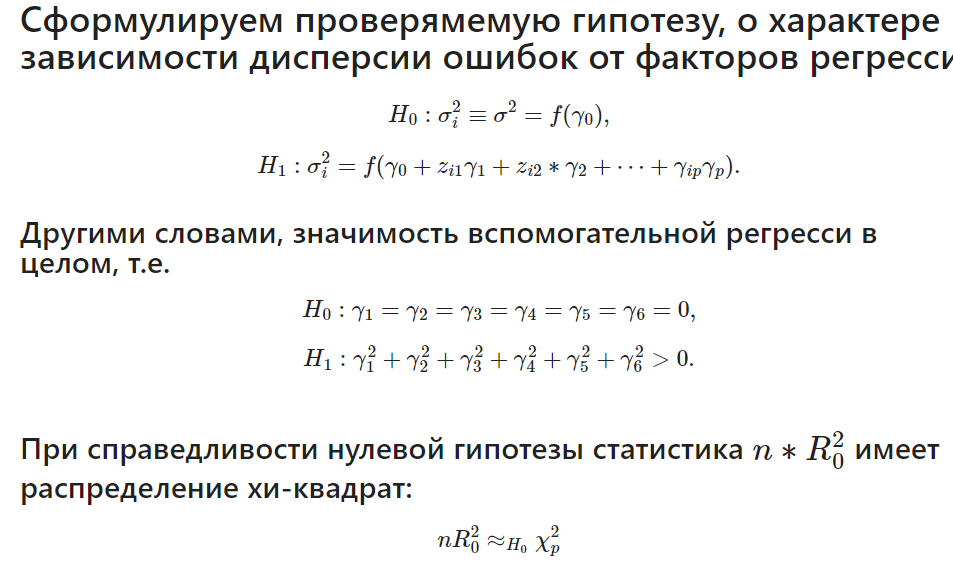
$$

H\_0: \gamma\_1=\gamma\_2=\gamma\_3=\gamma\_4=\gamma\_5=\gamma\_6=\gamma\_7=0

$$

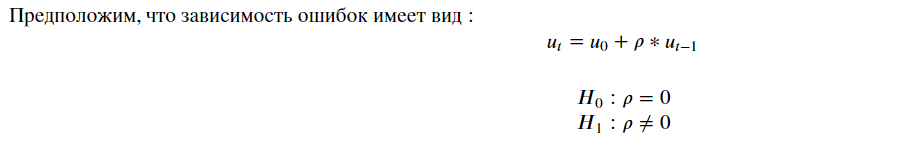
$$

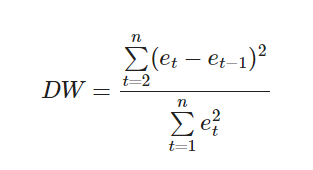
H\_1: \gamma\_1^2+\gamma\_2^2+\gamma\_3^2+\gamma\_4^2+\gamma\_5^2+\gamma\_6^2>0

$$  
  
**

BP\_model = smf.ols(data=df, formula='res2~..........').fit()  
BP\_model.params  
  
***# Zнабл=n\*R^2 и для проверки гипотезы на справедливость имеет распределение хи-квадрат***  
Z\_nabl = len(df)\*BP\_model.rsquared  
Z\_nabl  
  
***# Z критическое - Хи-квадрат***  
Hi2 = stats.chi2.ppf(1-0.05,5)  
Hi2  
  
***# состоятельные std err в условиях гетероскедастичности***  
BP\_model = smf.ols(data=df, formula='np.log(cost)~....').fit**(cov\_type='HC3')**  
BP\_model.summary(alpha=0.01).tables[1]  
  
***# F-тест модели***  
f\_test = BP\_model.f\_test( 'np.log(pl)=log\_pl2=0' )   
f\_test.fvalue #будет показано только Fнабл.

# 13. Автокорреляция DW - тест





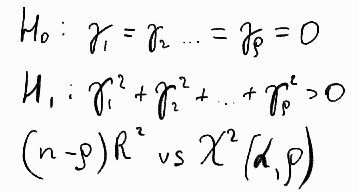
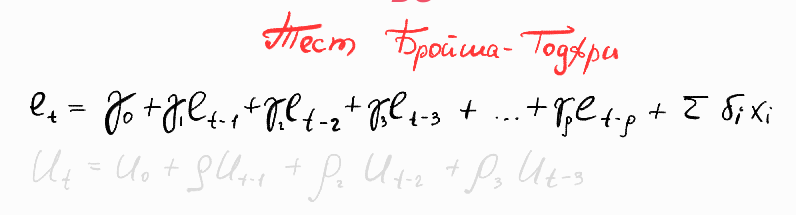
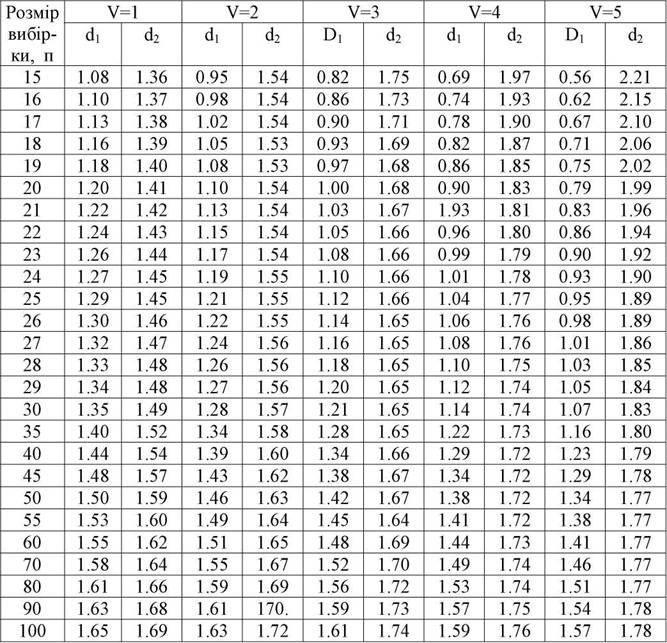
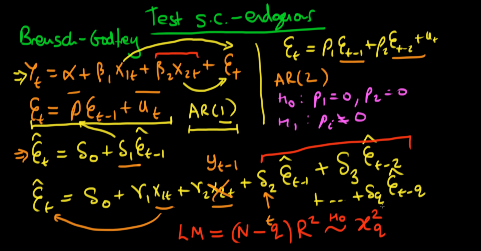
***#Знаменатель DW***  
res\_sum = sum(model.resid\*\*2)  
  
***# Вычислим числитель***  
sum\_resid = 0  
***#запускаем цикл***  
for i in range(len(model.resid)-1):  
 sum\_resid+=(model.resid[i+1]-model.resid[i])\*\*2  
  
***# DW***  
DW = sum\_resid/res\_sum  
DW

from statsmodels.stats.stattools import durbin\_watson

***#perform Durbin-Watson test***

durbin\_watson(model.resid)

******

 ******  
  
***BG***  
  
***LM тест***  
  
  
  
******

df [ 'log\_yd' ] = np.log( df [ 'yd' ])   
df [ 'log\_ce' ] = np.log(df [ 'ce' ])   
df.diff() ***#разница между настоящим и предыдущим значением***

***#perform Breusch-Godfrey test at order p = 3***  
print(dg.acorr\_breusch\_godfrey(fitted, nlags=3))

***#добавляем остатки***  
res0=mdl.resid   
res0   
  
***#удаляем первое значение***   
res1 = res0.drop(labels = [1], axis = 0)   
res1  
  
***#удаляем второе значение***   
res2 = res1.drop(labels = [2], axis = 0)   
res2   
  
***#reset\_index() - чтобы начать отсчет с 0; drop=True - чтобы сдвинуть вниз значения***  
df['res0'] = res0 . reset\_index(drop = True)   
df['res1'] = res1 . reset\_index(drop = True)   
df['res2'] = res2 . reset\_index(drop = True)   
df['res3'] = res3 . reset\_index(drop = True)   
df  
  
***#удаляем значения NaN***   
df.dropna(inplace = True)   
df  
  
***#строим модель + сразу LM тест***  
sub\_mdl = smf.ols(data = df , formula = 'res0~res1+res2+res3+log\_ce.diff()').fit ()   
sub\_mdl.rsquared\*(len(df)-rho)  
  
***#Хи квадрат***  
Hi2 = stats.chi2.ppf(1 - alpha, rho)   
Hi2