Static model in Time Series and Heteroscedasticity Test

Date: Monday July 4 04:22:57 2022

Author: kkitonga and ewayagi

In [1]:
```
#=========================Libraries=========================================#
from statsmodels.compat import lzip
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
import statsmodels.stats.diagnostic as dg
import statsmodels.stats.api as sms
```

In [4]:
```
#=========================Loading Data=======================================#

#Importing inflation csv
#Use the path directory for you computer
inflation = pd.read_csv ('C:\DOCs\Py, R and Stata\Py\data_inflation.csv')

#Converting to dataframe and selecting only column on argentina
infDf = pd.DataFrame(inflation,columns= ["ARG"])

#Renaming column arg to inflation
infDf=infDf.rename(columns={"ARG":"inf"})

#Seeing output
print(infDf)

#Importing unemployment csv
unemp= pd.read_csv('C:\DOCs\Py, R and Stata\Py\data_unemployment.csv')

#Convert unemp to dataframe
unempDf = pd.DataFrame(unemp,columns=["ARG"])

#Rename ARG column
unempDf=unempDf.rename(columns={"ARG":"unemployment"})

#View dataframe
print(unempDf)

#Creating merged data frame of unemployment and infaltion for argentina
tdDF= pd.concat([unempDf,infDf],axis="columns")

#Drop Nan rows
tdDF=tdDF.dropna()
print(tdDF)
```

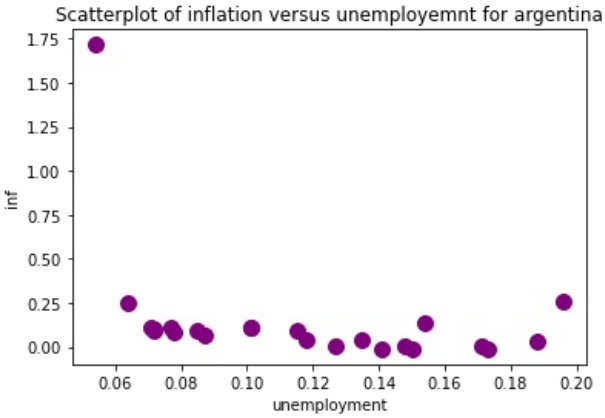
```
         inf
0   1.716717
1   0.248999
2   0.106115
3   0.041773
4   0.033761
5   0.001557
6   0.005273
7   0.009203
8  -0.011669
9  -0.009359
10 -0.010666
11  0.258685
12  0.134428
13  0.044157
14  0.096394
15  0.109011
16  0.088314
17  0.085840
18  0.062828
19  0.107801
20  0.094657
21  0.100303
22  0.106194
23       NaN
24       NaN
25       NaN
    unemployment
0          0.054
1          0.064
2          0.101
3          0.118
4          0.188
5          0.171
6          0.148
7          0.127
8          0.141
9          0.150
10         0.173
11         0.196
12         0.154
13         0.135
14         0.115
15         0.101
16         0.085
17         0.078
18         0.087
19         0.077
20         0.072
21         0.072
22         0.071
23           NaN
24           NaN
25           NaN
    unemployment       inf
0          0.054  1.716717
1          0.064  0.248999
2          0.101  0.106115
3          0.118  0.041773
4          0.188  0.033761
5          0.171  0.001557
6          0.148  0.005273
7          0.127  0.009203
8          0.141 -0.011669
9          0.150 -0.009359
10         0.173 -0.010666
11         0.196  0.258685
12         0.154  0.134428
13         0.135  0.044157
14         0.115  0.096394
15         0.101  0.109011
16         0.085  0.088314
17         0.078  0.085840
18         0.087  0.062828
19         0.077  0.107801
20         0.072  0.094657
21         0.072  0.100303
22         0.071  0.106194
```

In [5]:

```
#======================Data Visualization==================================#

#Scatter plot
tdDF.plot(x='unemployment',y='inf',kind='scatter',s=100,color='purple')
plt.title("Scatterplot of inflation versus unemployemnt for argentina")
```

Out[5]:

```
Text(0.5, 1.0, 'Scatterplot of inflation versus unemployemnt for argentina')
```



In [6]:
```
#==============================Descritives=====================================#

print(tdDF.head())                        #First five observations
print(tdDF.tail())                        #last five observations
print (tdDF.describe())                   #descriptive statistics
```

```
   unemployment       inf
0         0.054  1.716717
1         0.064  0.248999
2         0.101  0.106115
3         0.118  0.041773
4         0.188  0.033761
    unemployment       inf
18         0.087  0.062828
19         0.077  0.107801
20         0.072  0.094657
21         0.072  0.100303
22         0.071  0.106194
       unemployment        inf
count     23.000000  23.000000
mean       0.116435   0.148709
std        0.042774   0.349300
min        0.054000  -0.011669
25%        0.077500   0.021482
50%        0.115000   0.088314
75%        0.149000   0.106998
max        0.196000   1.716717
```

In [7]:
```
#============================Running Regression================================#

#Define independent and dependent variables
X=tdDF['unemployment']
Y=tdDF['inf']

#Add a constant
X=sm.add_constant(X)

#Define model
model = sm.OLS(Y, X)

#Fit model
model_result = model.fit()

                         #Regression Result
print(model_result.summary())

                      #Generating residuals and fitted values
#Fitted values
tdDF['Fitted'] = model_result.predict()

#Residuals (manually) :observed infaltion -fitted values
tdDF['Residual'] =  tdDF['inf']-tdDF['Fitted']

#Residuals :Automatically
tdDF['resids2'] = model_result.resid

#Squared residuals
tdDF['Residualsq'] = tdDF['Residual'] * tdDF['Residual']
```

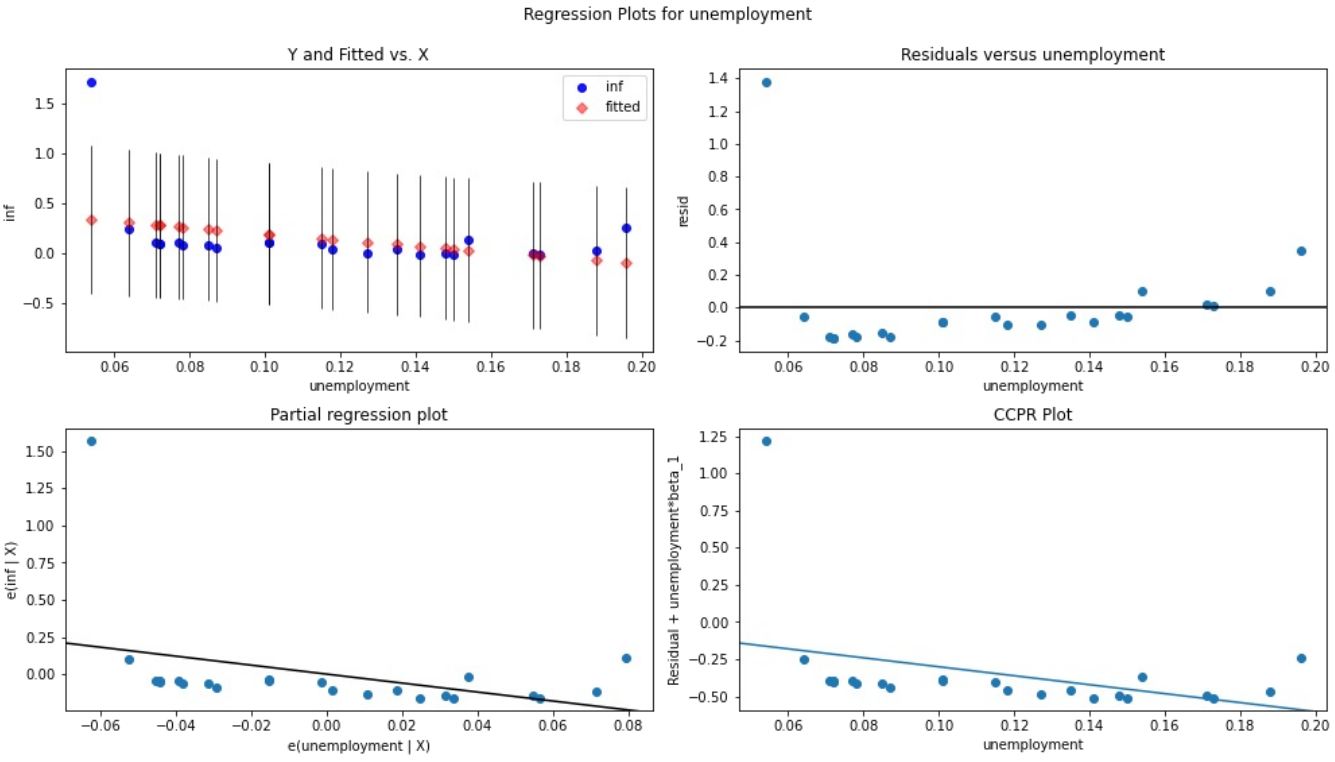
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    inf   R-squared:                       0.136
Model:                            OLS   Adj. R-squared:                  0.095
Method:                 Least Squares   F-statistic:                     3.310
Date:                Mon, 05 Sep 2022   Prob (F-statistic):             0.0832
Time:                        06:12:04   Log-Likelihood:                -6.2494
No. Observations:                  23   AIC:                             16.50
Df Residuals:                      21   BIC:                             18.77
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.4995      0.205      2.438      0.024       0.073       0.926
unemployment  -3.0131      1.656     -1.819      0.083      -6.458       0.431
==============================================================================
Omnibus:                       46.966   Durbin-Watson:                   1.008
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              201.137
Skew:                           3.572   Prob(JB):                     2.11e-44
Kurtosis:                      15.604   Cond. No.                         24.2
==============================================================================

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

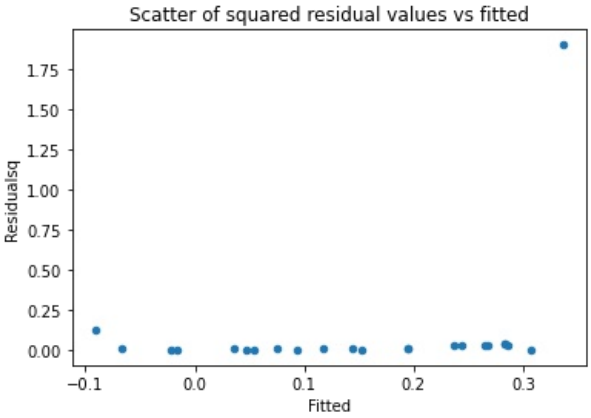
In [8]:

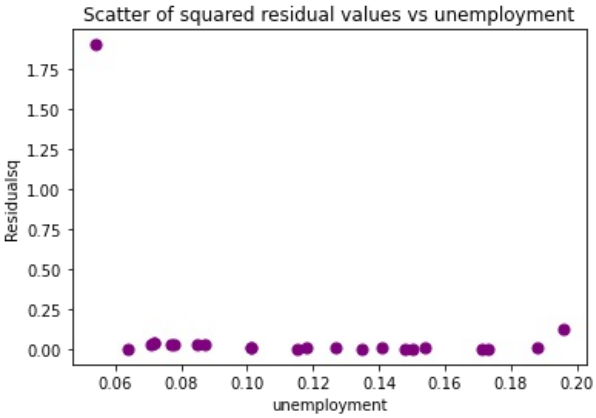
```
#====================Heteroscedasticity Tests================================#

        #=======Visual Inspection:Residual plots======#

               #Plot 1. Residual plots
                   #Modify figure size
fig = plt.figure(figsize=(14, 8))
               #Creating regression plots
fig = sm.graphics.plot_regress_exog(model_result, 'unemployment', fig=fig)

            #Plot2. Residuals squared versus fitted values
tdDF.plot(x='Fitted',y='Residualsq',kind='scatter')
plt.title( "Scatter of squared residual values vs fitted")

             #Plot3. Residuals squared versus explanatory
tdDF.plot(x='unemployment',y='Residualsq',kind='scatter',color='purple',s=50)
plt.title( "Scatter of squared residual values vs unemployment")
```

Out[8]:

```
eval_env: 1
Text(0.5, 1.0, 'Scatter of squared residual values vs unemployment')
```







In [9]:
```
#=======================Statistical Tests=====================================#

                    #Breusch pagan
Bp_output = ['Breusch-pagan Lagrange multiplier test statistic', 'p-value',
         'f-value', 'f p-value']
test_result = np.round(sms.het_breuschpagan(model_result.resid, model_result.model.exog),2)
print(dict(lzip(Bp_output, test_result)))

                    #White test
WhiteTestLabs=['White test Lagrange multiplier test statistic', 'p-value',
         'f-value', 'f p-value']
white_test = np.round(sms.het_white(model_result.resid,  model_result.model.exog),2)
print(dict(lzip(WhiteTestLabs, white_test)))

#=================================END=======================================#
```

```
{'Breusch-pagan Lagrange multiplier test statistic': 2.26, 'p-value': 0.13, 'f-value': 2.29, 'f p-value': 0.14}
{'White test Lagrange multiplier test statistic': 6.84, 'p-value': 0.03, 'f-value': 4.23, 'f p-value': 0.03}
```

In [ ]: