How can I perform a Breusch-Godfrey test using Python?

Authored by

stats writer

April 28, 2024

The Breusch-Godfrey test is a statistical test used to determine the presence of autocorrelation in a time series data. This test can be performed using Python by first importing the necessary libraries such as statsmodels and numpy. Then, the data should be preprocessed and converted into a time series format. Next, the Breusch-Godfrey test function from the statsmodels library can be used to calculate the test statistic and p-value. Finally, the results can be interpreted to determine the presence of autocorrelation in the data. Overall, performing a Breusch-Godfrey test using Python involves importing libraries, preprocessing the data, and using a specific function to obtain the test results.

Perform a Breusch-Godfrey Test in Python

One of the is that there is no correlation between the residuals, e.g. the residuals are independent.

To test for first-order autocorrelation, we can perform a . However, if we'd like to test for autocorrelation at higher orders then we need to perform a Breusch-Godfrey test.

This test uses the following :

H0 (null hypothesis): There is no autocorrelation at any order less than or equal to *p*.

HA (alternative hypothesis): There exists autocorrelation at some order less than or equal to *p*.

The test statistic follows a Chi-Square distribution with *p* degrees of freedom.

If the that corresponds to this test statistic is less than a certain significance level (e.g. 0.05) then we can reject the null hypothesis and conclude that autocorrelation exists among the residuals at some order less than or equal to *p*.

To perform a Breusch-Godfrey test in Python, we can use the acorr_breusch_godfrey() function from the statsmodels library.

The following step-by-step example explains how to perform the Breusch-Godfrey test in Python.

Step 1: Create the Data

First, let's create a dataset that contains two predictor variables (x1 and x2) and one response variable (y).

```
import pandas as pd

#create dataset
df = pd.DataFrame({'x1': ,
'x2': ,
'y': })

#view first five rows of dataset
```

```
df.head()

x1 x2 y
0 3 7 24
1 4 7 25
2 4 8 25
3 5 8 27
4 8 12 29
```

Step 2: Fit a Regression Model

Next, we can fit a using x1 and x2 as predictor variables and y as the .

```
import statsmodels.apias sm

#define response variable
y = df

#define predictor variables
x = df]

#add constant to predictor variables
x = sm.add_constant(x)

#fit linear regression model
```

```
model = sm.OLS(y, x).fit()
```

Step 3: Perform the Breusch-Godfrey Test

Next, we'll perform the Breusch-Godfrey test to test for autocorrelation among the residuals at order *p*. For this example, we'll choose *p* = 3.

```
import statsmodels.stats.diagnosticas dg

#perform Breusch-Godfrey test at order p = 3
print(dg.acorr_breusch_godfrey(model, nlags=3))

(8.70314827, 0.0335094873, 5.27967224, 0.0403980576)
```

The first value in the output represents the test statistic and the second value represents the corresponding p-value.

From the output we can see the following:

Test statistic X2 = 8.7031P-value = 0.0335

Since this p-value is less than 0.05, we can reject the null hypothesis and conclude that autocorrelation exists among the residuals at some order less than or

equal to 3.

How to Handle Autocorrelation

If you reject the null hypothesis and conclude that autocorrelation is present in the residuals, then you have a few different options to correct this problem if you deem it to be serious enough:

For positive serial correlation, consider adding lags of the dependent and/or independent variable to the model.For negative serial correlation, check to make sure that none of your variables are *overdifferenced*.For seasonal correlation, consider adding seasonal dummy variables to the model.