Learning the Breusch-Godfrey Test for Autocorrelation in Python

Authored by

Mohammed looti

November 5, 2025

RECOMMENDED CITATION

Mohammed looti (2025). *Learning the Breusch-Godfrey Test for Autocorrelation in Python*. PSYCHOLOGICAL STATISTICS. Retrieved from https://statistics.arabpsychology.com/?p=10235

The Critical Role of Autocorrelation Testing in Regression Analysis

One of the most foundational principles underlying classical statistical modeling, particularly in time series analysis and **linear regression**, is the assumption of independent errors. This means that the **residuals**--the calculated differences between the observed data points and the values predicted by the model--must be uncorrelated with each other across observations. When this assumption is violated, typically meaning that the error term in one period is related to the error term in previous periods, we encounter the significant econometric problem known as **autocorrelation** (or serial correlation).

Ignoring the presence of serial correlation can have severe consequences for the reliability of model inferences. While the coefficient estimates themselves often remain unbiased under autocorrelation, the calculated standard errors become biased and inconsistent. This inconsistency leads to inefficient estimation, meaning our statistical tests (like t-tests) may yield invalid results, potentially causing us to make incorrect conclusions regarding the significance of our model parameters. Therefore, accurately detecting and quantifying the presence and extent of autocorrelation is a prerequisite for building robust and trustworthy predictive models.

While the traditional Durbin-Watson statistic is a reliable and frequently used tool for identifying first-order autocorrelation (lag 1), its utility diminishes when the correlation pattern spans multiple time periods or higher orders. To overcome this limitation and effectively test for serial correlation extending beyond the immediate previous observation, researchers rely on the powerful and flexible **Breusch-Godfrey test**. This generalized diagnostic approach allows analysts to explicitly specify the maximum lag order (denoted as *p*) to be included in the investigation of error correlation.

Theoretical Framework of the Breusch-Godfrey Test

The **Breusch-Godfrey test** operates by constructing an auxiliary regression model based on the **residuals** obtained from the original fitted model. Essentially, this test determines whether these residuals can be explained by their own lagged values up to the specified order *p*. If the residuals are truly random and independent, their lagged values should not hold any explanatory power.

The statistical hypotheses guiding the test are clearly defined and center on the absence of serial correlation up to the maximum lag order *p* specified by the user:

H0 (Null Hypothesis): There is no **autocorrelation** among the residuals for all lag orders less than or equal to *p*.
HA (Alternative Hypothesis): Significant autocorrelation exists among the residuals at at least one lag order less than or equal to *p*.

The test statistic derived from this procedure is typically based on the R-squared value of the auxiliary regression, adjusted for the sample size. This statistic asymptotically follows a **Chi-Square distribution** with *p* degrees of freedom, where *p* is the maximum lag order tested. The decision criteria hinges entirely upon the calculated **p-value** associated with this test statistic. If the calculated **p-value** falls below the chosen significance level (commonly set at 0.05 or 5%), we possess sufficient statistical evidence to reject the **Null Hypothesis**. A rejection of H0 strongly implies that significant serial correlation is indeed present in the **residuals**, signaling a potential problem with the underlying model specification that must be addressed.

Prerequisites and Data Preparation in Python

To perform the **Breusch-Godfrey test** effectively, we must first establish and fit a base statistical model from which we can extract the residuals for examination. Python's rich ecosystem, particularly the **statsmodels** library, simplifies this process significantly. The initial steps involve preparing the data structure using the powerful **pandas** library and defining the variables that will form the basis of our regression analysis.

In this practical implementation, we utilize **pandas** to construct a simple time-series-like dataset suitable for a standard multiple regression context. This dataset includes a response variable (`y`) and two independent predictor variables (`x1` and `x2`). Defining the data clearly is crucial as the subsequent regression model will rely on the relationship between these variables to generate the error terms that the Breusch-Godfrey test will diagnose.

```
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
```

Fitting the Ordinary Least Squares (OLS) Model

Once the data frame is prepared, the next essential stage involves fitting the primary regression model. Since the **Breusch-Godfrey test** is a post-estimation diagnostic, it requires the error terms (residuals) generated by a previously established model. We utilize the Ordinary Least Squares (OLS) method, which is the standard approach for estimating parameters in **linear regression**, accessible via the **statsmodels** API.

A critical procedural step when using the `sm.OLS` class in **statsmodels** is the explicit inclusion of an intercept term. Unlike some other libraries which automatically include an intercept, **statsmodels** requires the user to manually add a constant to the set of predictor variables using the `sm.add_constant()` function. Failing to include this constant would result in a model that forces the regression line through the origin, potentially leading to severely biased estimates and misleading residuals.

The following Python code defines the dependent (`y`) and independent (`x1` and `x2`) variables, adds the necessary constant, and then fits the linear regression model. The resulting `model` object contains all the necessary information, including the calculated **residuals**, which are crucial inputs for the subsequent diagnostic test.

```
import statsmodels.api as 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()
```

Executing and Interpreting the Breusch-Godfrey Test Results

With the linear model successfully fitted, we can now execute the diagnostic test using the `acorr_breusch_godfrey()` function available within the `statsmodels.stats.diagnostic` submodule. This function requires two primary arguments: the fitted model object itself and the parameter `nlags`, which specifies the maximum lag order *p* for which we are testing serial correlation. In our practical example, we set `nlags=3`, indicating an investigation of first, second,

and third-order autocorrelation.

Executing the function yields a tuple containing four specific output values, though the first two are generally the most important for drawing a conclusion based on standard hypothesis testing procedures: the test statistic and the corresponding **p-value**.

```
import statsmodels.stats.diagnostic as 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 output tuple provides the following crucial information for our test (where *p*=3):

Element 1: The LM (Lagrange Multiplier) Test Statistic (X2 = **8.7031**).
Element 2: The P-value corresponding to the Test Statistic (**0.0335**).
Element 3: The F-statistic (5.2796).
Element 4: The P-value corresponding to the F-statistic (0.0404).

Our primary focus is on the comparison between the P-value (0.0335) and the conventional significance level, $\alpha = 0.05$. Since the calculated P-value (0.0335) is decisively less than the significance threshold (0.05), we conclude that there is sufficient statistical evidence to reject the **Null Hypothesis** (H0). This rejection confirms the presence of statistically significant **autocorrelation** among the **residuals** up to the third lag (p=3), indicating that our current OLS model is likely inefficient and requires adjustment.

Remedial Strategies for Addressing Serial Correlation

The detection of **autocorrelation** via the **Breusch-Godfrey test** is a signal that the model has failed to capture all the systematic temporal information present in the data. Addressing this issue is vital to ensure that the estimated standard errors are reliable, leading to valid confidence intervals and hypothesis tests. The appropriate corrective action often depends on the nature and source of the serial correlation identified.

Several key strategies are employed to mitigate the effects of autocorrelation:

Model Augmentation for Positive Correlation: If the test reveals positive serial correlation (where a positive residual tends to be followed by another positive residual), the model may be dynamically underspecified. A common remedy is to augment the original **linear regression** model by incorporating lagged terms of the dependent variable (an Autoregressive model component) or including lagged values of the independent variables. These additions help the model explicitly

account for the temporal dependence that was previously absorbed into the error term.

Checking for Overdifferencing in Negative Correlation: Negative serial correlation, characterized by alternating positive and negative **residuals**, can sometimes be an artifact of overdifferencing the data. If the data transformation involved taking differences more times than necessary to achieve stationarity, this artificial correlation pattern can be introduced. Analysts must review their data preparation steps to confirm that the degree of differencing is appropriate.

Using HAC Standard Errors: When the exact form of the autocorrelation is unknown or difficult to model parametrically, a robust approach is to adjust the standard error calculations directly. Heteroskedasticity and Autocorrelation Consistent (HAC) standard errors, such as the Newey-West estimator, provide reliable estimates of the standard errors even in the presence of serial correlation and heteroskedasticity, thereby safeguarding the validity of statistical inferences without altering the model coefficients themselves.

Introducing Seasonal Adjustments: In cases where the correlation occurs systematically at fixed, repeating intervals (e.g., quarterly, yearly), the presence of seasonal correlation is likely. This is best addressed by introducing seasonal dummy variables into the model, allowing the regression equation to absorb these predictable periodic shifts in the data.

By carefully selecting and applying these remedial techniques, researchers can transform an inefficient model into a robust framework, ensuring that the estimated standard errors are consistent and that the hypothesis tests conducted on the model parameters are statistically valid.