Learning White’s Test for Heteroscedasticity in Python: A Step-by-Step Guide

Authored by

Mohammed looti

November 1, 2025

RECOMMENDED CITATION

Mohammed looti (2025). *Learning White’s Test for Heteroscedasticity in Python: A Step-by-Step Guide*. PSYCHOLOGICAL STATISTICS. Retrieved from https://statistics.arabpsychology.com/?p=7816

Introduction: The Critical Importance of Homoscedasticity in Regression Modeling

When developing any robust regression model, a set of underlying assumptions must be satisfied for the resulting statistical inferences to be valid and reliable. One of the most critical assumptions pertaining to the error term (or residuals) is that of **homoscedasticity**. This sophisticated term simply means that the variance, or spread, of the residuals remains constant across all observations, regardless of the corresponding values of the independent (predictor) variables. Essentially, the errors are uniformly distributed throughout the range of the model.

The violation of this core assumption is known as heteroscedasticity--a scenario where the variance of the residuals changes systematically. For instance, the errors might be very small for low values of a predictor but become increasingly larger as the predictor variable increases. Such non-constant variance introduces significant complications into the interpretation of the model results, even though the calculated coefficient estimates themselves remain mathematically unbiased under Ordinary Least Squares (OLS) estimation.

The primary danger of heteroscedasticity lies in its impact on the calculation of the standard errors of the coefficients. When variance is unstable, OLS estimates of these standard errors become biased and inconsistent. Unreliable standard errors, in turn, invalidate traditional hypothesis tests, such as t-tests and F-tests, leading to inaccurate confidence intervals and potentially erroneous conclusions about the statistical significance of the predictor variables. Therefore, diagnosing and addressing this issue is paramount for sound econometric and statistical analysis.

Diagnosing Variance Instability: The Utility of White's Test

To accurately diagnose the presence of non-constant error variance, statistical practitioners frequently turn to diagnostic tests. Among the most popular and powerful tools is White's test, formally known as White's general test for heteroscedasticity. Developed by economist Halbert White in 1980, this test is highly valued because it is remarkably general; it makes no specific assumptions about the functional form of the heteroscedasticity. Unlike some other tests, White's test checks for variance instability against all potential forms of relationship between the error variance and the regressors.

The mechanism behind White's test involves conducting an auxiliary regression model. In this secondary regression, the squared residuals from the original OLS model are regressed against the original predictor variables, the squares of those predictors, and all possible cross-products (interaction terms) between the predictors. The core hypothesis is then tested by examining the R-squared value derived from this auxiliary regression. A statistically significant R-squared value indicates that the independent variables are capable of explaining the variance in the errors,

confirming the presence of heteroscedasticity.

Implementing White's test efficiently requires robust computational tools. This comprehensive tutorial provides a detailed, step-by-step guide on how to apply and interpret this crucial diagnostic test within the Python environment, leveraging the powerful statistical capabilities provided by the **statsmodels** library. Understanding this workflow is essential for ensuring the veracity of any regression model built using Python.

Preparation: Setting Up the Python Environment and Data

Before we can execute any statistical modeling or diagnostic testing, we must ensure our Python environment is properly configured with the necessary scientific computing libraries. The implementation of White's test is highly streamlined within the **statsmodels** library, which offers extensive functionality for statistical modeling, estimation, and hypothesis testing. Furthermore, effective data handling is managed using **pandas**, the industry standard for tabular data manipulation in Python.

For this practical demonstration, we will utilize the publicly available **mtcars** dataset. This classic dataset contains 32 observations detailing various performance metrics and design aspects of automobiles. Our objective is to construct a multiple linear regression model aimed at predicting Miles Per Gallon (mpg) based on selected characteristics. Once the model is fitted, we will extract the residuals and subject them to White's test to assess the critical assumption of constant variance.

The initial coding phase involves importing the required modules and loading the dataset. We use the `pd.read_csv()` function to pull the data directly from a remote URL into a Pandas DataFrame. The DataFrame is the foundational data structure used for virtually all data analysis in Python, providing essential tools for indexing, inspection, and preparation prior to statistical analysis. Following the data load, using the `data.info()` method allows for a quick structural inspection, confirming that all 32 records were loaded correctly and verifying the data types of the variables, which is vital for numerical modeling.

```
from sklearn.linear_model import LinearRegression
from statsmodels.stats.diagnostic import het_white
import statsmodels.api as sm
import pandas as pd

#define URL where dataset is located
url = "https://raw.githubusercontent.com/Statology/Python-Guides/main/mtcars.csv"

#read in data
```

```
data = pd.read_csv(url)

#view summary of data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 12 columns):
# Column Non-Null Count Dtype
--- ------ -------------- -----
0 model 32 non-null object
1 mpg 32 non-null float64
2 cyl 32 non-null int64
3 disp 32 non-null float64
4 hp 32 non-null int64
5 drat 32 non-null float64
6 wt 32 non-null float64
7 qsec 32 non-null float64
8 vs 32 non-null int64
9 am 32 non-null int64
10 gear 32 non-null int64
11 carb 32 non-null int64
dtypes: float64(5), int64(6), object(1)
```

Step 2: Defining and Estimating the Linear Regression Model

With the data loaded and inspected within the Pandas DataFrame, the immediate next step involves specifying and fitting the primary linear regression model. This model will serve as the basis for our diagnostic test, as we must first generate the model residuals (the differences between the observed and predicted values) before we can analyze their variance. We select `mpg` (Miles Per Gallon) as our dependent or **response variable**, which we aim to predict. For our independent or **predictor variables**, we choose `disp` (engine displacement) and `hp` (horsepower), creating a simple multivariate model.

We employ the Ordinary Least Squares (OLS) estimation technique, accessed via `statsmodels.OLS`. A crucial methodological detail when using `statsmodels` is the explicit inclusion of an intercept term (or constant). By default, many statistical packages include this automatically, but `statsmodels` requires the user to prepend a column of ones to the predictor matrix. This is achieved efficiently using the `sm.add_constant(x)` function, ensuring that the resulting model incorporates a y-intercept, which is statistically essential unless a specific

theoretical justification dictates otherwise.

After correctly structuring the response variable vector (y) and the predictor matrix (x), the model estimation is completed using the `.fit()` method. This process minimizes the sum of squared residuals, yielding the optimal coefficient estimates. Critically, the fitted `model` object retains all necessary components for post-estimation analysis, including the raw residuals (`model.resid`) and the design matrix (`model.model.exog`), both of which are direct inputs for the subsequent White's test.

#define response variable
y = data

#define predictor variables
x = data]

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

#fit regression model
model = sm.OLS(y, x).fit()

Step 3: Executing White's Test and Interpreting the Results

The final analytical phase involves performing the diagnostic test using the `het_white` function, which is located within the `statsmodels.stats.diagnostic` module. This function automates the creation and testing of the auxiliary regression model mentioned earlier. It requires two mandatory inputs: the vector of residuals from the fitted model (`model.resid`) and the full design matrix of exogenous variables used in the original model (`model.model.exog`), which includes the constant term.

The output of `het_white` provides four key statistical measures. The most critical result for hypothesis testing is the Chi-squared Test Statistic and its corresponding p-value. This statistic is derived from the R-squared of the auxiliary regression, scaled by the sample size, and follows a Chi-squared distribution under the null hypothesis. A large test statistic and a small p-value suggest that the squared residuals are indeed related to the predictors, indicating variance instability.

To properly contextualize the output, we must establish the formal hypotheses governing White's test. This structure provides the framework for our decision-making process based on the calculated p-value:

Null Hypothesis (H0): Homoscedasticity is present (The error variance is constant).
Alternative Hypothesis (HA): Heteroscedasticity is present (The error variance is non-constant).

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

#define labels to use for output of White's test
labels =

#print results of White's test
print(dict(zip(labels, white_test)))

{'Test Statistic': 7.076620330416624, 'Test Statistic p-value': 0.21500404394263936,
'F-Statistic': 1.4764621093131864, 'F-Test p-value': 0.23147065943879694}

Analyzing the numerical output, we observe that the Chi-squared Test Statistic is **7.0766**, yielding a corresponding p-value of approximately **0.215**. If we adopt the conventional significance level (alpha, α) of 0.05, we compare the p-value against this threshold. Since 0.215 > 0.05, the evidence is insufficient to warrant the rejection of the null hypothesis. Consequently, we conclude that there is no statistically significant evidence of heteroscedasticity in this particular model, validating the assumption of constant error variance, or **homoscedasticity**.

Addressing Heteroscedasticity: Remedial Strategies

The outcome of White's test dictates the subsequent steps in the model validation process. If, as in our example, the test suggests that **homoscedasticity** holds, we can proceed confidently with the standard interpretation of the OLS coefficient estimates and their associated standard errors, knowing that our statistical inference is robust. However, if the null hypothesis is rejected (p-value < 0.05), corrective measures must be implemented to ensure the reliability of the statistical conclusions.

A confirmed case of heteroscedasticity means that while the coefficient estimates themselves remain unbiased, the calculated standard errors are incorrect, jeopardizing inference. Addressing this issue is critical for accurate hypothesis testing regarding predictor significance. Fortunately, statisticians have developed several robust strategies to handle this challenge, ranging from modifying the calculation of standard errors to fundamentally altering the model estimation process or the data itself.

The following are the three leading methods used to correct for variance instability in linear regression models:

Utilizing Heteroscedasticity-Consistent (Robust) Standard Errors: This is generally the most straightforward and preferred modern solution. Instead of relying on the assumption of constant variance, robust standard errors (such as Huber-White standard errors) recalculate the variance-covariance matrix of the estimators in a way that is consistent even under the presence of heteroscedasticity. This method preserves the original OLS coefficient estimates while producing statistically sound p-values and confidence intervals, thus correcting the inference without changing the model fit.

Transformation of Variables: Another common approach involves applying a mathematical transformation to the variables, most often the **response variable**. The log transformation (using the natural logarithm) is frequently effective in compressing the scale of the response, thereby stabilizing the variance of the residuals. While successful in resolving the variance issue, this approach necessitates careful reinterpretation of the regression coefficients, as they now reflect proportional or percentage changes rather than absolute unit changes, which may complicate communication of results.

Employing Weighted Regression (WLS): Weighted regression, or Weighted Least Squares, is a more sophisticated estimation technique. WLS requires knowledge of, or estimates of, the precise functional form of the heteroscedasticity. Each observation is assigned a weight that is inversely proportional to the estimated variance of its error term. By giving less influence to data points associated with higher variance (the 'noisy' observations), WLS restores the efficient OLS property and satisfies the assumption of **homoscedasticity**, leading to more efficient coefficient estimates than OLS provides under heteroscedasticity.

Conclusion and Next Steps in Python Statistical Modeling

The ability to perform, interpret, and act upon diagnostic tests like White's test is fundamental to conducting responsible and reliable quantitative research using regression models. Acknowledging and correcting assumption violations ensures that the conclusions drawn from the data are statistically sound and trustworthy. The Python ecosystem, particularly through the **statsmodels** library, offers exceptional ease of use and computational speed for executing these complex statistical diagnostics.

Mastery of these concepts goes beyond simple model fitting. It involves a continuous process of assumption validation. To further enhance your expertise in statistical modeling within Python, we highly recommend exploring advanced topics related to model diagnostics and alternative estimation techniques. Continued learning in this domain will significantly strengthen the reliability and impact of your data analysis projects.

We encourage you to explore further documentation and tutorials related to the following subjects to enhance your mastery of statistical modeling and Python implementation:

Detailed exploration of the different types of Heteroscedasticity-Consistent (standard error) estimators (HC0, HC1, HC3) available in `statsmodels`.
Advanced applications of linear regression, including the use of generalized linear models (GLMs) when OLS assumptions fail.
Techniques for data cleaning, transformation, and manipulation using the Pandas DataFrame for complex statistical projects.