How to perform a Wald Test in Python?

Authored by

stats writer

November 19, 2025

To perform a Wald Test in Python, we primarily rely on the powerful statsmodels package. This robust library provides necessary statistical tools for complex hypothesis testing. The process generally involves fitting a statistical model, such as a multiple linear regression, to the dataset. Following the model fitting, the dedicated methods within the model object--or the summary output--allow us to extract the specific Wald test statistic and its corresponding p-value. This test is fundamentally utilized to evaluate the collective statistical significance of a group of regression coefficients within the established model, helping practitioners determine which predictors contribute meaningfully to the outcome variable.

Theoretical Foundation: Understanding the Wald Test

A **Wald test** is a fundamental tool in frequentist statistics, employed to determine if a set of constraints imposed on the model parameters are valid. Specifically, it can be used to test if one or more parameters in a statistical model are simultaneously equal to certain hypothesized values, typically zero. The primary advantage of the Wald test lies in its versatility and ease of implementation, often appearing in the summary outputs of various regression models, including ordinary least squares (OLS) and generalized linear models (GLMs). It provides a quantitative measure of how far the estimated parameters deviate from the values specified under the null hypothesis, scaled by the precision of those estimates.

This test is particularly crucial in regression analysis because it directly addresses the question of variable relevance. By testing if the coefficient associated with one or more predictor variables is zero, we are essentially determining whether those variables have any statistically significant linear relationship with the response variable, after accounting for all other predictors in the model. This application is central to the process of model selection and simplification, ensuring that only necessary predictors are retained in the final specification.

Formulating the Null and Alternative Hypotheses

The application of the Wald test requires the establishment of formal statistical hypotheses, defining the condition we are attempting to reject (the null hypothesis) and the condition we assume if the rejection occurs (the alternative hypothesis). When testing the significance of multiple regression coefficients simultaneously, the hypotheses are framed as follows:

H0 (Null Hypothesis): Some specified set of predictor variables are all equal to zero (e.g., $\beta 1 = \beta 2 = 0$). This suggests that none of these variables, collectively, contribute significantly to the prediction of the response variable.
HA (Alternative Hypothesis): Not all predictor variables in the specified set are equal to zero. This implies that at least one of the variables in the group possesses a statistically significant effect on the response.

If we fail to reject the null hypothesis, based on a large p-value, then we can confidently drop the specified set of predictor variables from the model because they do not offer a statistically significant improvement in the fit of the model. Conversely, rejecting the null hypothesis implies that the subset of variables is jointly meaningful and should be retained.

Setting Up the Environment and Dataset in Python

To demonstrate the practical execution of the Wald test, we will utilize the classic **mtcars** dataset. We aim to construct a multiple linear regression model predicting miles per gallon (mpg) based on several car characteristics. The necessary software environment includes the **statsmodels** library for statistical modeling and the **pandas** library for efficient data handling.

Our chosen regression model will take the following mathematical form, where mpg is the dependent variable, and `disp` (displacement), `carb` (carburetor count), `hp` (horsepower), and `cyl` (number of cylinders) are the independent predictor variables:

mpg = β0 + β1disp + β2carb + β3hp + β4cyl

The following code shows how to fit this regression model and view the model summary using the `ols` function provided by statsmodels.

Executing the Regression and Analyzing the Summary Output

The execution of the model fitting process requires careful data loading and the use of the `fit()` method on the OLS formula. The initial model summary provides valuable, though individual, insights into the significance of each coefficient through their respective t-tests.

```
import statsmodels.formula.api as smf
import pandas as pd
import io

#define dataset as string
mtcars_data="""model,mpg,cyl,disp,hp,drat,wt,qsec,vs,am,gear,carb
Mazda RX4,21,6,160,110,3.9,2.62,16.46,0,1,4,4
Mazda RX4 Wag,21,6,160,110,3.9,2.875,17.02,0,1,4,4
Datsun 710,22.8,4,108,93,3.85,2.32,18.61,1,1,4,1
Hornet 4 Drive,21.4,6,258,110,3.08,3.215,19.44,1,0,3,1
Hornet Sportabout,18.7,8,360,175,3.15,3.44,17.02,0,0,3,2
Valiant,18.1,6,225,105,2.76,3.46,20.22,1,0,3,1
Duster 360,14.3,8,360,245,3.21,3.57,15.84,0,0,3,4
Merc 240D,24.4,4,146.7,62,3.69,3.19,20,1,0,4,2
```

Merc 230,22.8,4,140.8,95,3.92,3.15,22.9,1,0,4,2
Merc 280,19.2,6,167.6,123,3.92,3.44,18.3,1,0,4,4
Merc 280C,17.8,6,167.6,123,3.92,3.44,18.9,1,0,4,4
Merc 450SE,16.4,8,275.8,180,3.07,4.07,17.4,0,0,3,3
Merc 450SL,17.3,8,275.8,180,3.07,3.73,17.6,0,0,3,3
Merc 450SLC,15.2,8,275.8,180,3.07,3.78,18,0,0,3,3
Cadillac Fleetwood,10.4,8,472,205,2.93,5.25,17.98,0,0,3,4
Lincoln Continental,10.4,8,460,215,3,5.424,17.82,0,0,3,4
Chrysler Imperial,14.7,8,440,230,3.23,5.345,17.42,0,0,3,4
Fiat 128,32.4,4,78.7,66,4.08,2.2,19.47,1,1,4,1
Honda Civic,30.4,4,75.7,52,4.93,1.615,18.52,1,1,4,2
Toyota Corolla,33.9,4,71.1,65,4.22,1.835,19.9,1,1,4,1
Toyota Corona,21.5,4,120.1,97,3.7,2.465,20.01,1,0,3,1
Dodge Challenger,15.5,8,318,150,2.76,3.52,16.87,0,0,3,2
AMC Javelin,15.2,8,304,150,3.15,3.435,17.3,0,0,3,2
Camaro Z28,13.3,8,350,245,3.73,3.84,15.41,0,0,3,4
Pontiac Firebird,19.2,8,400,175,3.08,3.845,17.05,0,0,3,2
Fiat X1-9,27.3,4,79,66,4.08,1.935,18.9,1,1,4,1
Porsche 914-2,26,4,120.3,91,4.43,2.14,16.7,0,1,5,2
Lotus Europa,30.4,4,95.1,113,3.77,1.513,16.9,1,1,5,2
Ford Pantera L,15.8,8,351,264,4.22,3.17,14.5,0,1,5,4
Ferrari Dino,19.7,6,145,175,3.62,2.77,15.5,0,1,5,6
Maserati Bora,15,8,301,335,3.54,3.57,14.6,0,1,5,8
Volvo 142E,21.4,4,121,109,4.11,2.78,18.6,1,1,4,2"""

#convert string to DataFrame
df = pd.read_csv(io.StringIO(mtcars_data), sep=",")

#fit multiple linear regression model
results = smf.ols('mpg ~ disp + carb + hp + cyl', df).fit()

#view regression model summary
results.summary()

coef std err t P>|t|
Intercept34.0216 2.523 13.482 0.000 28.844 39.199
disp -0.0269 0.011 -2.379 0.025 -0.050 -0.004
carb -0.9269 0.579 -1.601 0.121 -2.115 0.261
hp 0.0093 0.021 0.452 0.655 -0.033 0.052
cyl -1.0485 0.784 -1.338 0.192 -2.657 0.560

From the output above, we note that while `disp` appears statistically significant (P>|t| = 0.025), the other predictors, including `hp` and `cyl`, have p-values far above the conventional 0.05 threshold. However, individual insignificance does not guarantee joint insignificance. This necessitates the use of a joint test, such as the Wald test, to rigorously test the compound hypothesis that these coefficients are simultaneously zero.

Applying the wald_test() Function for Joint Significance

Next, we can use the **wald_test()** function from the fitted model object in **statsmodels** to test the joint null hypothesis: βhp = 0 and βcyl = 0. This procedure determines if we can safely remove both `hp` and `cyl` from the model without any significant decrease in its explanatory power. This is a crucial step in model simplification.

The `wald_test()` function requires a string argument that specifies the linear constraints under the null hypothesis. For testing if coefficients are equal to zero, we simply list the terms, separated by commas, and set them equal to zero, as shown in the example below. This capability allows for complex hypothesis testing far beyond what is provided by the standard summary output.

The following code shows how to apply this function in practice to evaluate the joint significance of the `hp` and `cyl` variables:

#perform Wald Test to determine if 'hp' and 'cyl' coefficients are both zero
print(results.wald_test('(hp = 0, cyl = 0)'))

F test: F=array(]), p=0.41403001184235005, df_denom=27, df_num=2

Interpreting the P-Value and Drawing Conclusions

The output of the Wald test yields an F test statistic (F ≈ 0.911) and, most importantly, the associated p-value, which is **0.414**. This p-value quantifies the probability of observing the current data, or data more extreme, assuming the null hypothesis (H0: βhp = 0, βcyl = 0) is true.

Since this p-value (0.414) is significantly larger than the standard significance level ($alpha = 0.05$), we **fail to reject the null hypothesis** of the Wald test. Statistically, this means there is insufficient evidence to conclude that the coefficients for `hp` and `cyl` are collectively non-zero.

This result suggests we can assume the regression coefficients for the predictor variables "hp" and "cyl" are both equal to zero, making them viable candidates for removal during a forward or backward selection process aimed at achieving a more efficient statistical model.

Further Applications of the Wald Test

Beyond testing simultaneous zero restrictions, the Wald test is powerful enough to test any linear constraints on the model parameters. For example, a researcher might hypothesize that two variables have an equal effect (e.g., βdisp = βcarb), or that the sum of two coefficients equals a specific non-zero value. The flexibility afforded by the `wald_test()` syntax allows for testing these complex structural assumptions, ensuring that the final model specification adheres to both statistical principles and theoretical expectations of the modeled phenomena.

The following tutorials explain how to perform other common statistical operations in Python: