**How to Detect Heteroskedasticity in Time Series**

**Detecting and dealing with non-constant variance in time series**



Photo by [Jannes Glas](https://unsplash.com/@jannesglas?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com?utm_source=medium&utm_medium=referral)

A time series is heteroskedastic if its variance changes over time. Otherwise, the data set is homoskedastic.

Heteroskedasticity affects the modeling of time series. So, it is important to detect and deal with this condition.

**Introduction**

Let’s start with a visual example.

Figure 1 below shows the popular airline passengers’ time series. You can see that the variation is different across the series. The variance is higher in the latter part of the series. This is also where the level of the data is also higher.



Figure 1: Monthly passengers in an airline. [The data set is publicly available in the *pmdarima* Python library.](https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.datasets.load_airpassengers.html) Image by author.

Changes in variance are problematic for forecasting. It impacts the fitting of an adequate model, thereby affecting forecasting performance.

But, visual inspection only is not practical. How can you detect and deal with heteroskedasticity in a more systematic way?

**Detecting Heteroskedasticity**

You can check whether a time series is heteroskedastic using statistical tests. These include the following:

* [White test](https://www.statsmodels.org/stable/generated/statsmodels.stats.diagnostic.het_white.html#statsmodels.stats.diagnostic.het_white);
* [Breusch-Pagan test](https://www.statsmodels.org/stable/generated/statsmodels.stats.diagnostic.het_breuschpagan.html#statsmodels.stats.diagnostic.het_breuschpagan);
* [Goldfeld–Quandt test](https://www.statsmodels.org/stable/generated/statsmodels.stats.diagnostic.het_goldfeldquandt.html#statsmodels.stats.diagnostic.het_goldfeldquandt)

The main input to these tests is the residuals of a regression model (e.g. ordinary least squares). The null hypothesis is that the residuals are distributed with equal variance. If the p-value is smaller than the significance level we reject this hypothesis. This means that the time series is heteroskedastic. The significance level is often set to a value up to *0.05*.

The *statsmodels* Python library has an implementation of the three tests above. Here’s a snippet that wraps these in a single class:

import pandas as pd  
import statsmodels.stats.api as sms  
from statsmodels.formula.api import ols  
  
TEST\_NAMES = ['White', 'Breusch-Pagan', 'Goldfeld-Quandt']  
FORMULA = 'value ~ time'  
  
  
class Heteroskedasticity:  
  
 @staticmethod  
 def het\_tests(series: pd.Series, test: str) -> float:  
 """  
 Testing for heteroskedasticity  
  
 :param series: Univariate time series as pd.Series  
 :param test: String denoting the test. One of 'white','goldfeldquandt', or 'breuschpagan'  
  
 :return: p-value as a float.  
  
 If the p-value is high, we accept the null hypothesis that the data is homoskedastic  
 """  
 assert test in TEST\_NAMES, 'Unknown test'  
  
 series = series.reset\_index(drop=True).reset\_index()  
 series.columns = ['time', 'value']  
 series['time'] += 1  
  
 olsr = ols(FORMULA, series).fit()  
  
 if test == 'White':  
 \_, p\_value, \_, \_ = sms.het\_white(olsr.resid, olsr.model.exog)  
 elif test == 'Goldfeld-Quandt':  
 \_, p\_value, \_ = sms.het\_goldfeldquandt(olsr.resid, olsr.model.exog, alternative='two-sided')  
 else:  
 \_, p\_value, \_, \_ = sms.het\_breuschpagan(olsr.resid, olsr.model.exog)  
  
 return p\_value  
  
 @classmethod  
 def run\_all\_tests(cls, series: pd.Series):  
  
 test\_results = {k: cls.het\_tests(series, k) for k in TEST\_NAMES}  
  
 return test\_results

The class *Heteroskedasticity* contains two functions. The function *het\_tests* applies a specific test (White, Breusch-Pagan, or Goldfeld-Quandt). The function *run\_all\_tests* applies all three tests in one go. The output of these functions is the p-value of the corresponding test.

Here’s how you can apply this code to the time series in Figure 1.

from pmdarima.datasets import load\_airpassengers  
  
# https://github.com/vcerqueira/blog/blob/main/src/heteroskedasticity.py  
from src.heteroskedasticity import Heteroskedasticity  
  
series = load\_airpassengers(True)  
  
test\_results = Heteroskedasticity.run\_all\_tests(series)  
  
# {'Breusch-Pagan': 4.55e-07,  
# 'Goldfeld-Quandt': 8.81e-13,  
# 'White': 4.34e-07}

The p-value of all tests is close to zero. So, you can reject the null hypothesis. These tests give compelling evidence for the presence of heteroskedasticity.

Here’s the distribution of the residuals in the first and second half of the time series:

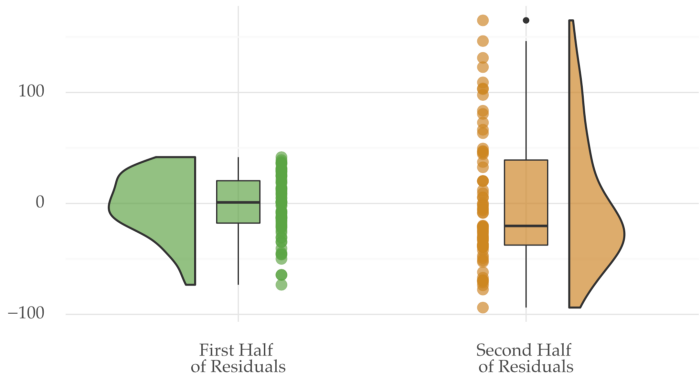


Figure 2: Distribution of the residuals in the first and second half of the time series. Image by author.

The distribution of residuals is different in these two parts. The Goldfeld-Quandt test uses this type of split to test for heteroskedasticity. It checks if the variance of the residuals is different in two data subsamples.

**Transforming the Data**

A common remedy to heteroskedasticity in time series is to transform the data. Taking the logarithm of the time series is helpful to stabilize its variability.

Here’s the same time series as before but log-scaled:

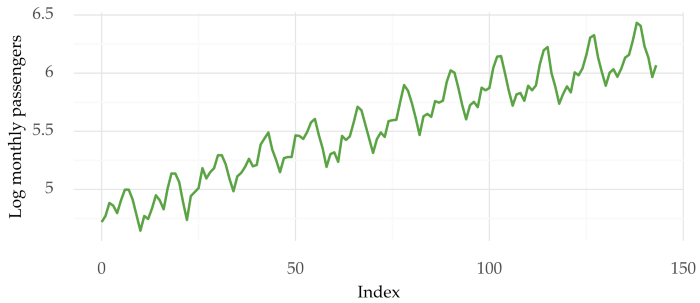


Figure 3: Like figure 1, but log-scaled. Here, the time series shows a stable variance over time.

This time, the variability looks steady along the series. Let’s re-run the tests using the log-scaled time series:

import numpy as np  
  
test\_results = Heteroskedasticity.run\_all\_tests(np.log(series))  
  
# {'Breusch-Pagan': 0.033,  
# 'Goldfeld-Quandt': 0.18,  
# 'White': 0.10}

The p-values are greater this time. Only one of the tests (Breusch-Pagan) rejects the hypothesis of constant variance. This is assuming a significance level of *0.05*.

**Reverting the log transformation**

Suppose you’re making predictions using log-transformed data. In that case, you need to revert the predictions to the original scale. This is done with the inverse of the transformation — in the case of log, you should use the exponential.

So, the steps of the forecasting process are the following:

1. Transform the data to stabilize the variance;
2. Fit a forecasting model;
3. Get the forecasts, and revert them to the original scale.

Here’s an example.

import numpy as np  
from pmdarima.datasets import load\_airpassengers  
from pmdarima.arima import auto\_arima  
from sklearn.model\_selection import train\_test\_split  
  
series = load\_airpassengers(True)  
  
# leaving the last 12 points for testing  
train, test = train\_test\_split(series, test\_size=12, shuffle=False)  
# stabilizing the variance in the train  
log\_train = np.log(train)  
  
# building an arima model, m is the seasonal period (monthly)  
mod = auto\_arima(log\_train, seasonal=True, m=12)  
  
# getting the log forecasts  
log\_forecasts = mod.predict(12)  
  
# reverting the forecasts  
forecasts = np.exp(log\_forecasts)

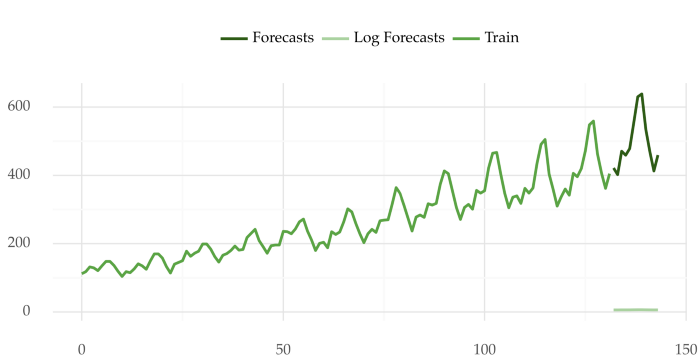


Figure 4: Reverting the forecasts to the original scale after modeling the transformed series. Image by author.

**Take Aways**

* Time series are heteroskedastic if the variance is not constant;
* You can test if a time series is heteroskedastic using statistical tests. These include the White Breusch-Pagan, or Goldfeld–Quandt tests;
* Use the log transformation to stabilize the variance;
* Don’t forget to revert the forecasts to the original scale.

Thanks for reading and see you in the next story!