Learn How to Identify Outliers with Grubbs' Test in Python

Authored by

Mohammed looti

November 7, 2025

The effective management of unusual observations, commonly known as **outliers**, is fundamental to rigorous statistical analysis and robust data modeling. If left unchecked, these extreme values can severely skew results, leading to inaccurate conclusions. To address this challenge, statisticians frequently employ the **Grubbs' Test**, formally recognized as the maximum normalized residual test. This powerful statistical procedure is specifically designed to detect whether the highest or lowest value in a univariate dataset is a genuine anomaly.

For the reliable application of the **Grubbs' Test**, two strict preconditions must be met: first, the data must be approximately **normally distributed**; and second, the dataset must contain a minimum of seven observations. This detailed tutorial provides a practical guide to implementing the **Grubbs' Test** efficiently within the **Python** environment, ensuring your data analysis maintains the highest levels of integrity and accuracy.

Prerequisites and Installation for Grubbs' Test in Python

To seamlessly execute the **Grubbs' Test** in **Python**, we utilize the specialized function `smirnov_grubbs.test()`, which is part of the excellent **outlier_utils** package. This package is specifically engineered to handle the complex statistical computations required to determine if the most extreme observation in your sample constitutes a statistical outlier, based on a predefined confidence threshold. Using this robust utility dramatically simplifies outlier detection workflows.

The core syntax for initiating the test is highly intuitive, primarily requiring the input dataset and offering an optional parameter to adjust the statistical stringency of the detection process:

smirnov_grubbs.test(data, alpha=.05)

The functional parameters used here govern the execution and interpretation of the test results:

data: This required parameter expects a numeric vector or a NumPy array containing the set of observations slated for outlier analysis.
alpha: This parameter specifies the critical **significance level** ($alpha$) for the test. It defines the probability threshold below which we reject the **null hypothesis** (H0: that no outliers exist). The default setting is 0.05, which corresponds to the standard 95% confidence level used across many scientific disciplines.

Before proceeding to the practical examples, the necessary dependency must be installed in your environment. The powerful **outlier_utils** library can be readily installed using Python's standard package installer, `pip`, via your command line interface:

pip install outlier_utils

Once the installation is complete, you can import the necessary modules and begin applying the **Grubbs' Test** to your specific datasets. The subsequent examples illustrate various applications of the testing functions tailored for different outlier detection requirements.

Example 1: Executing the Standard Two-Sided Grubbs' Test

The most frequent application of the **Grubbs' Test** involves the two-sided approach. This method is designed to be comprehensive, actively seeking statistical **outliers** that may reside at either the upper or the lower extreme of the data distribution. This is the preferred default application when analysts lack prior knowledge regarding the likely direction of the potential anomaly.

The following **Python** snippet demonstrates how to define a sample univariate dataset using the fundamental **NumPy** library. We then execute the core `grubbs.test()` function, which automatically identifies and returns the dataset with any detected outliers removed. Notice that the function modifies the array in place of returning a simple boolean result, streamlining the data cleaning process.

import numpy as np
from outliers import smirnov_grubbs as grubbs

#define data
data = np.array()

#perform Grubbs' test (two-sided)
grubbs.test(data, alpha=.05)

array()

The output array is the refined dataset from which the statistically significant **outlier** has been automatically removed. In this instance, the highest value, 40, was determined to be an anomaly at the specified $alpha=.05$ **significance level**. Consequently, this extreme observation was excluded from the resulting array, ensuring that the remaining data points are statistically sound for subsequent modeling.

Example 2: Focused Detection with the One-Sided Grubbs' Test

Although the two-sided test offers comprehensive detection, situations often arise where domain expertise dictates that an anomaly, if present, will only occur at one specific end of the distribution. In these focused scenarios, a one-sided **Grubbs' Test** is the appropriate tool. This method allows us to specifically test whether the absolute minimum value is an outlier, or whether the absolute maximum value is an outlier, providing increased statistical power in that direction.

The `smirnov_grubbs` module facilitates this focused analysis by providing two distinct functions: `min_test()` for detecting low outliers and `max_test()` for detecting high outliers. We utilize the identical dataset established previously to demonstrate both possibilities: testing for an unusually low value and testing for an unusually high value.

import numpy as np
from outliers import smirnov_grubbs as grubbs

#define data
data = np.array()

#perform Grubbs' test to see if minimum value (5) is an outlier
grubbs.min_test(data, alpha=.05)

array()

#perform Grubbs' test to see if maximum value (40) is an outlier
grubbs.max_test(data, alpha=.05)

array()

The results confirm the efficacy of the one-sided test: the value 5 is retained as non-anomalous when using `min_test()`, but the value 40 is decisively flagged and removed using `max_test()`. This outcome confirms the findings from the two-sided test, demonstrating that 40 is the statistically significant high extreme.

Example 3: Advanced Extraction: Locating and Quantifying Outliers

In practical data cleaning and forensic analysis, simply filtering the data may not be sufficient. Data professionals often need to know the exact location or the numerical magnitude of the anomaly. Knowing the specific **index** of the identified outlier within the original array is critical for targeted remediation, especially when working with large datasets or integrating results back into a database framework.

To retrieve positional information, the `max_test_indices()` function is invaluable. It returns the **index** (or indices) of the data points that were statistically flagged as outliers. This feature bypasses the need for manual searching, directly pointing to the array position that demands attention:

import numpy as np
from outliers import smirnov_grubbs as grubbs

```
#define data
data = np.array()

#perform Grubbs' test and identify index (if any) of the outlier
grubbs.max_test_indices(data, alpha=.05)
```

The resulting output, , clearly indicates that the detected anomaly is situated at **index** position 16 of the input array. Complementing this, the `max_test_outliers()` function allows for the direct extraction of the actual numerical value of the anomaly, providing the specific magnitude required for comprehensive reporting and review.

```
import numpy as np
from outliers import smirnov_grubbs as grubbs

#define data
data = np.array()

#perform Grubbs' test and identify the actual value (if any) of the outlier
grubbs.max_test_outliers(data, alpha=.05)
```

The output `` confirms the precise numerical value of the extreme observation, validating the findings from all previous applications of the **Grubbs' Test** on this dataset.

Responsible Data Governance: Next Steps After Outlier Detection

Successfully employing the **Grubbs' Test** to flag an observation as an outlier is only the first stage of robust data handling. The critical subsequent phase involves determining the appropriate and responsible strategy for managing this anomalous data point. It is essential to recognize that automatic removal is rarely the optimal or only recourse; careful consideration of the outlier's source and context is paramount.

Analysts should adhere to a structured, three-step decision-making framework when addressing data points identified as statistical **outliers**:

Verification of Data Integrity: The initial and most crucial action is to meticulously scrutinize the flagged value to verify that it does not stem from a mechanical error, such as a transcription mistake, a data entry typo, or a sensor malfunction. A significant number of apparent anomalies are simply human or recording errors. Always confirm the data source and ensure the value was captured correctly before proceeding with statistical intervention.

Imputation or Value Replacement: If the outlier is confirmed to be an error, or if maintaining the original sample size is a necessary requirement for downstream analysis, imputation should be considered. This involves replacing the erroneous value using a statistically justifiable method. Standard replacement techniques include assigning a measure of **central tendency**, such as the **mean or the median**, to the specific anomalous data point.

Strategic Removal: Removal should be reserved for cases where the value is confirmed to be a true, statistical outlier that significantly undermines the assumptions of the intended analysis. This step is justified only when the outlier is determined not to represent a fundamental, underlying process that must be accounted for in the final model.

By coupling the precise detection capabilities of the **Grubbs' Test** with these sound data governance practices, you ensure your analytical results are both statistically rigorous and contextually responsible.