

How can I perform a Box-Cox transformation in Python?

Authored by
stats writer

April 23, 2024

RECOMMENDED CITATION

stats writer (2024). *How can I perform a Box-Cox transformation in Python?*.

PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=138283>

The Box-Cox transformation is a statistical technique used to normalize data by transforming it into a Gaussian distribution. This transformation is commonly used in data analysis and machine learning. In order to perform a Box-Cox transformation in Python, you will need to first import the necessary libraries, such as `scipy.stats` and `numpy`. Then, you can use the `scipy.stats.boxcox()` function to apply the transformation to your data. This function takes in the data as an input and returns the transformed data as well as the lambda parameter used in the transformation. By adjusting this lambda parameter, you can control the degree of transformation applied to the data. This allows for greater flexibility in finding the most suitable transformation for your data. Overall, performing a Box-Cox transformation in Python is a simple and effective way to normalize your data for further analysis.

Perform a Box-Cox Transformation in Python

A box-cox transformation is a commonly used method for transforming a non-normally distributed dataset into a more normally distributed one.

The basic idea behind this method is to find some value for λ such that the transformed data is as close to normally distributed as possible, using the following formula:

$$y(\lambda) = (y^\lambda - 1) / \lambda \text{ if } y > 0 \quad y(\lambda) = \log(y) \text{ if } y = 0$$

We can perform a box-cox transformation in Python by using the `scipy.stats.boxcox()` function.

The following example shows how to use this function in practice.

Example: Box-Cox Transformation in Python

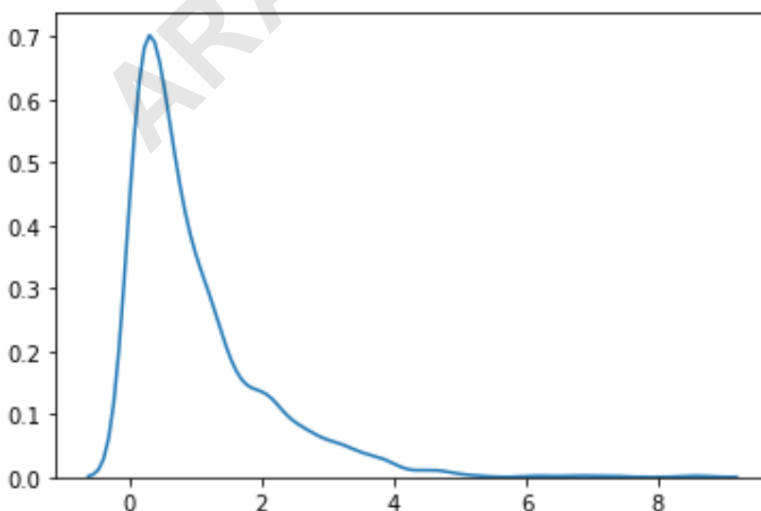
Suppose we generate a random set of 1,000 values that come from an :

```
#load necessary packagesimport numpy as np
from scipy.statsimport boxcox
import seaborn as sns
```

```
#make this example reproducible
np.random.seed(0)
```

```
#generate dataset
data = np.random.exponential(size=1000)
```

```
#plot the distribution of data values
sns.distplot(data, hist=False, kde=True)
```

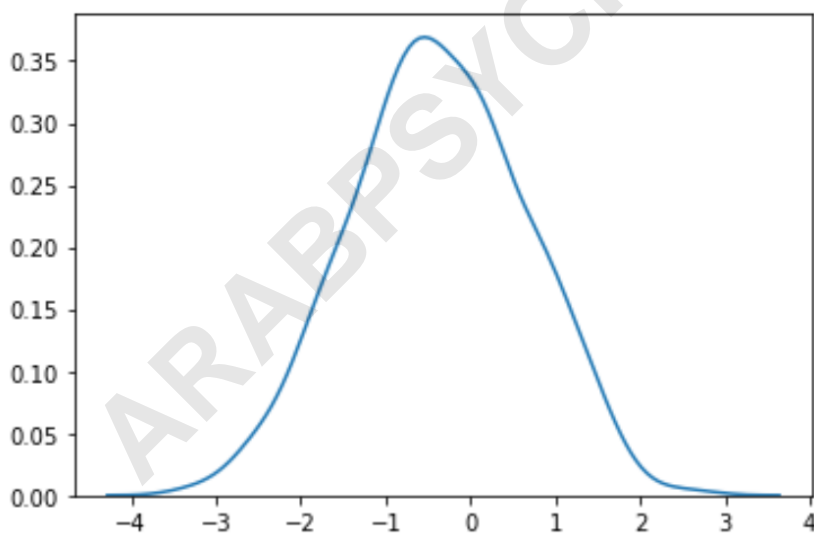


We can see that the distribution does not appear to be normal.

We can use the `boxcox()` function to find an optimal value of `lambda` that produces a more normal distribution:

```
#perform Box-Cox transformation on original data  
transformed_data, best_lambda = boxcox(data)
```

```
#plot the distribution of the transformed data values  
sns.distplot(transformed_data, hist=False, kde=True)
```



We can see that the transformed data follows much more of a normal distribution.

We can also find the exact lambda value used to perform the Box-Cox transformation:

```
#display optimal lambda value  
print(best_lambda)
```

```
0.2420131978174143
```

The optimal lambda was found to be roughly 0.242.

```
New = (old0.242 - 1) / 0.242
```

We can confirm this by looking at the values from the original data compared to the transformed data:

```
#view first five values of original dataset  
data
```

```
array()
```

```
#view first five values of transformed dataset  
transformed_data
```

```
array()
```

The first value in the original dataset was 0.79587. Thus,

we applied the following formula to transform this value:

$$\text{New} = (.795870.242 - 1) / 0.242 = -0.222$$

We can confirm that the first value in the transformed dataset is indeed -0.222.

How to Create & Interpret a Q-Q Plot in Python

How to Perform a Shapiro-Wilk Test for Normality in Python