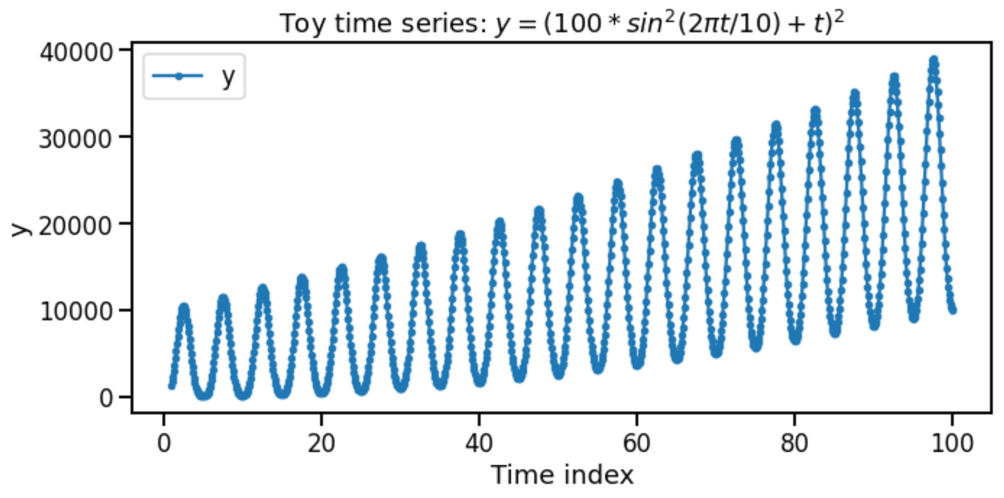
Box Cox transform

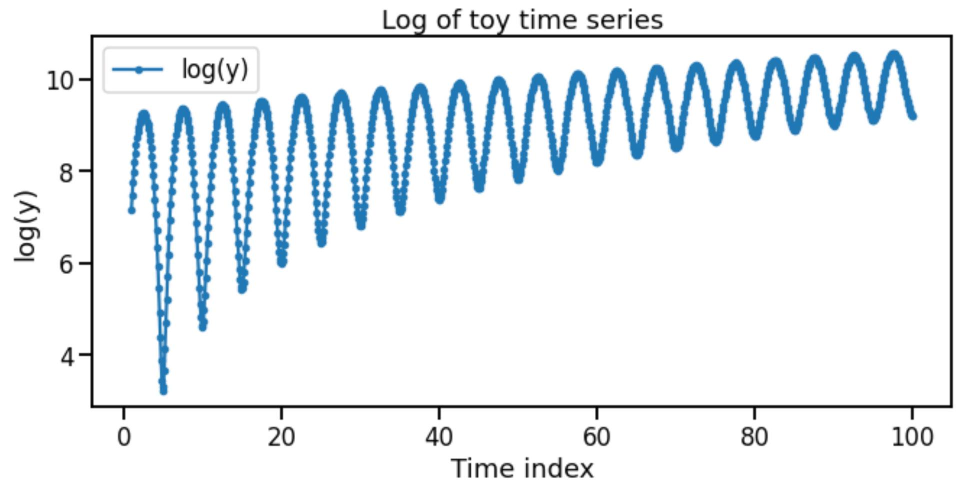
Time series decomposition

Contents

Motivation

- Some forecasting & decomposition methods perform better if the variance of the time series does not change with the level of the time series (e.g., ARIMA).
- The log transform does not always stabilize the variance of a time series. It depends on the time series.
- The Box Cox transform is a more general transform that can be used to stabilize the variance of a time series.





What kind of transforms are useful?

- What kind of transform of our time series, $f(y_t)$, could be useful in stabilising the variance? It depends on the time series.

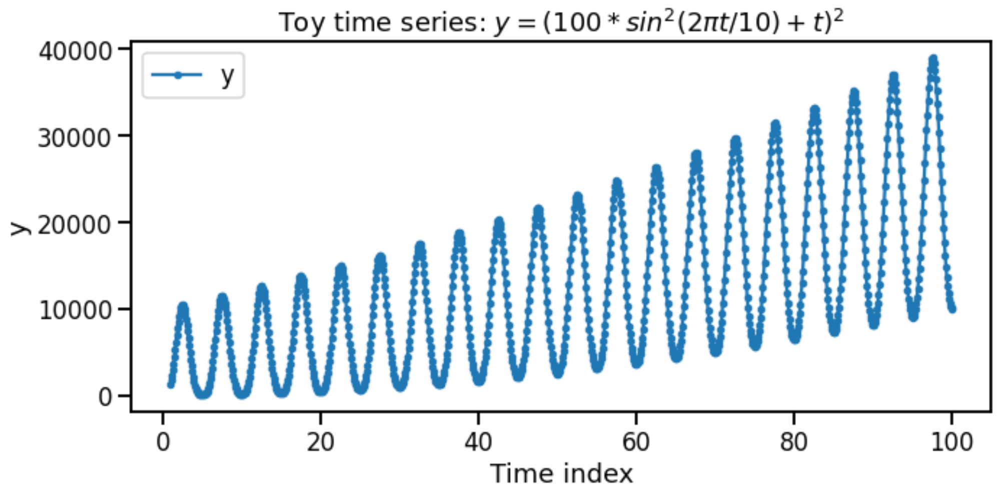
- To stabilise the variance we want the transformation to remove the interaction between the trend and any seasonality or noise term so we can write them additively: $f(y_t) = T_t + S_t + R_t$

- In our toy time series we had: $y_t = (\sin^2\left(\frac{2\pi t}{10}\right) + t)^2$

- We can think of this as: $y_t = (S_t + T_t)^2$

- A better transform would be using the square root!

$$y_t^{\frac{1}{2}} = S_t + T_t$$

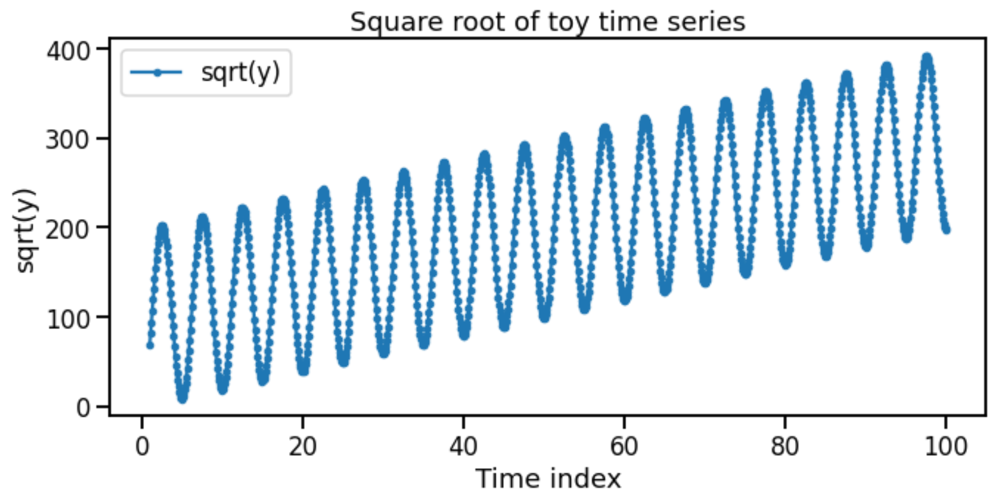


What kind of transforms are useful?

- What kind of transform of our time series, $f(y_t)$, could be useful in stabilising the variance? It depends on the time series.
- To stabilise the variance we want the transformation to remove the interaction between the trend and any seasonality or noise term so we can write them additively: $f(y_t) = T_t + S_t + R_t$
- In our toy time series we had: $y_t = (\sin^2\left(\frac{2\pi t}{10}\right) + t)^2$
- We can think of this as: $y_t = (S_t + T_t)^2$
- A better transform would be using the square root!

$$y_t^{\frac{1}{2}} = S_t + T_t$$

- Now the variance does not change with the trend.



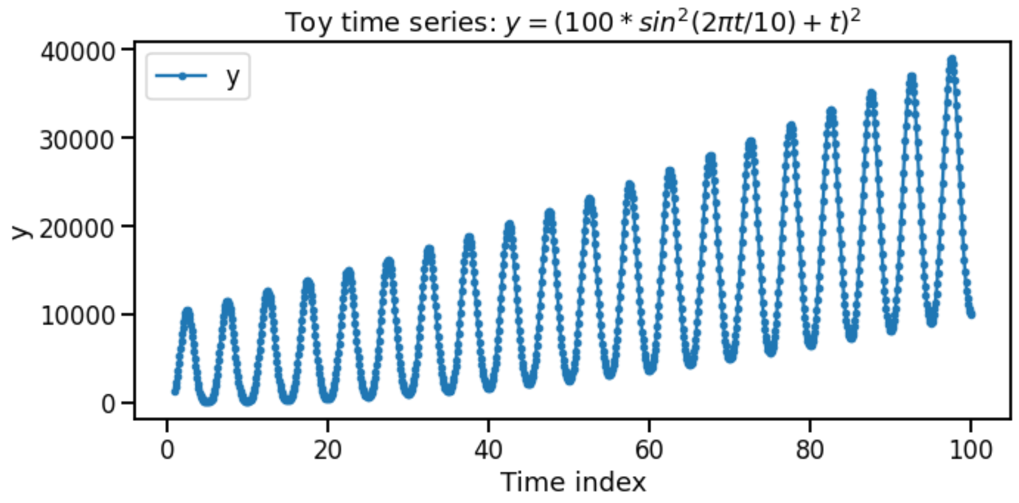
What kind of transforms are useful?

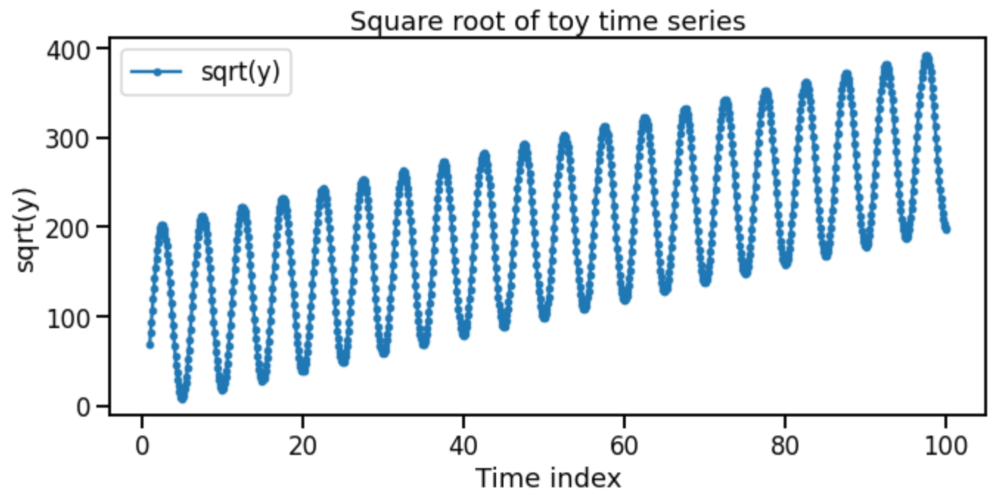
- $y^{\frac{1}{2}}$ is an example of what is called a power transform. We raised the original variable to some power.

- A more general way of writing power transforms are: y^{λ}

- Sometimes a power transform can be better at stabilizing the variance than a log transform. It depends on the time series.

- The Box Cox transform combines both a log transform and a power transform.





Box Cox transform

- The Box Cox transform is defined as:

$$
\begin{aligned}
y^{(\lambda)} &= \frac{y^{\lambda}-1}{\lambda}; && \text{if } \lambda \neq 0 \\
&= \log(y)\,; && \text{if } \lambda = 0
\end{aligned}
$$

- Different values of λ correspond to different kinds of transforms.

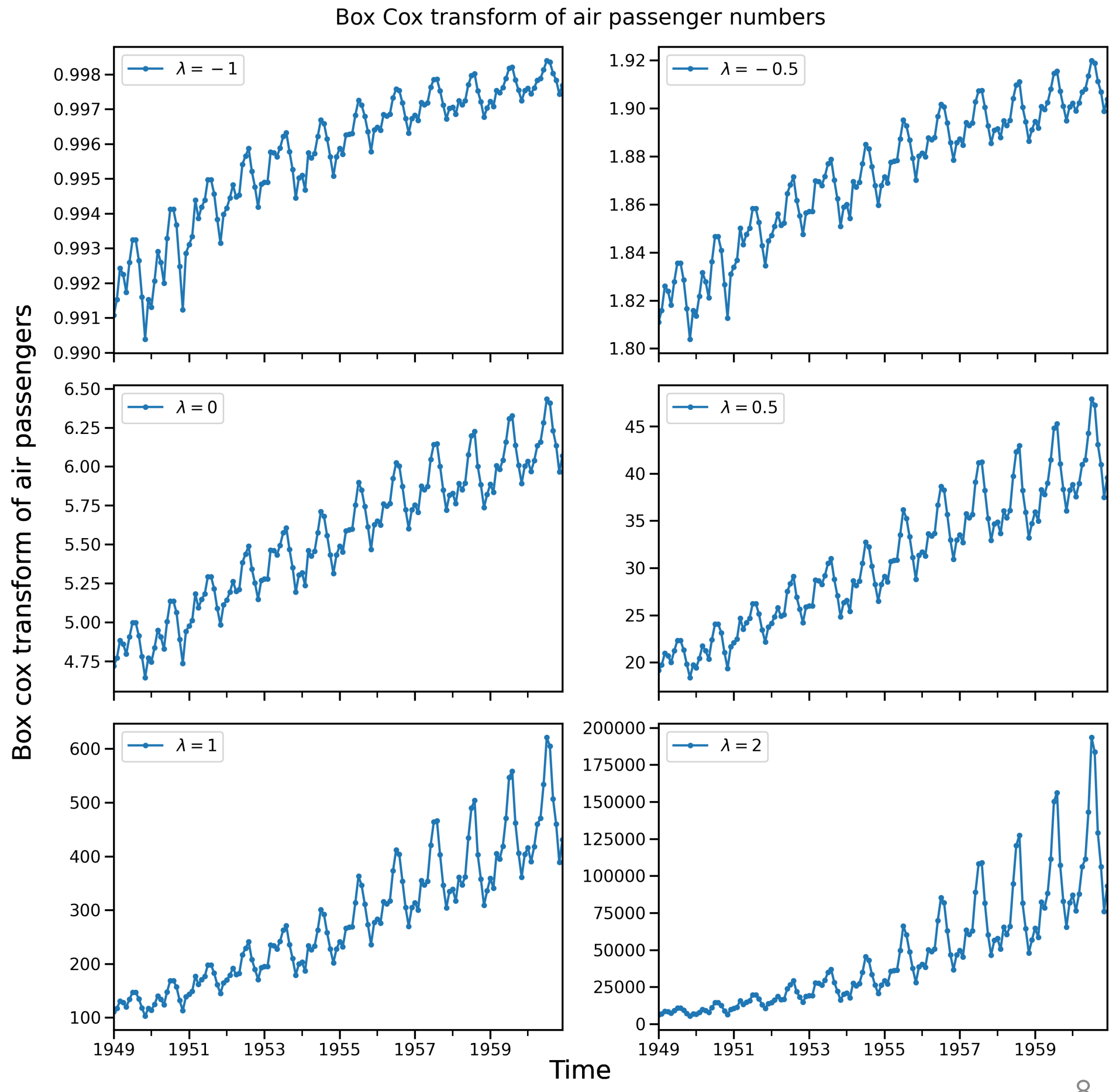
- In practice, λ is typically set between -5 and 5.

λ	Box Cox formula	Transform
0	$\log(y)$	Log
1	$y-1$	Shift by 1
2	½ (y^2-1)	Square
0.5	$2(\sqrt{y}-1)$	Square root
-1	$-(y^{-1}-1)$	Inverse

- y must be positive. If the data has any negative values then it can be transformed to be positive by adding a constant beforehand to the whole time series: $y \rightarrow y + c$.

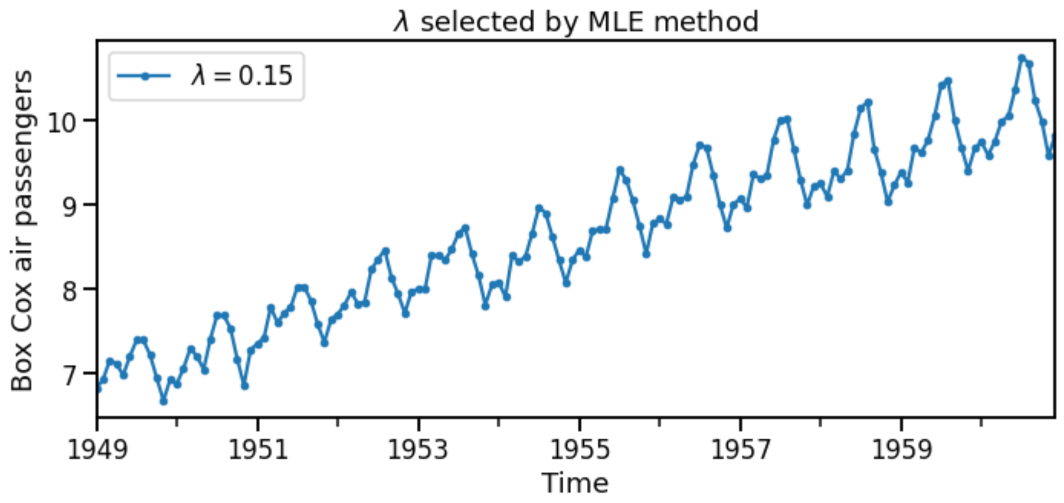
Box Cox transform

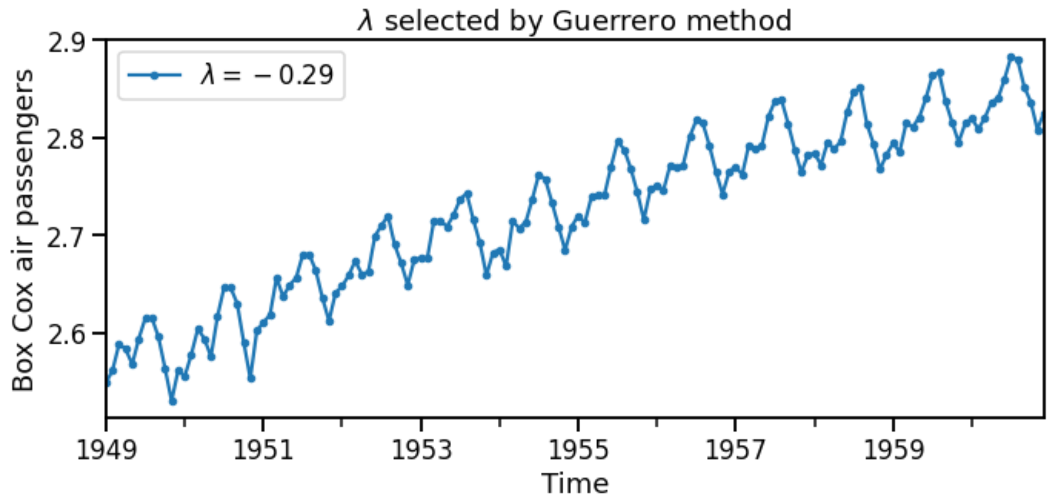
- A good value for λ makes the variance the same size across the time series.
- How do we pick a good value for λ?
- Try different λ, plot the data and check visually that the variance is nearly constant.
- Use a method that automatically selects λ that optimizes on some criteria.



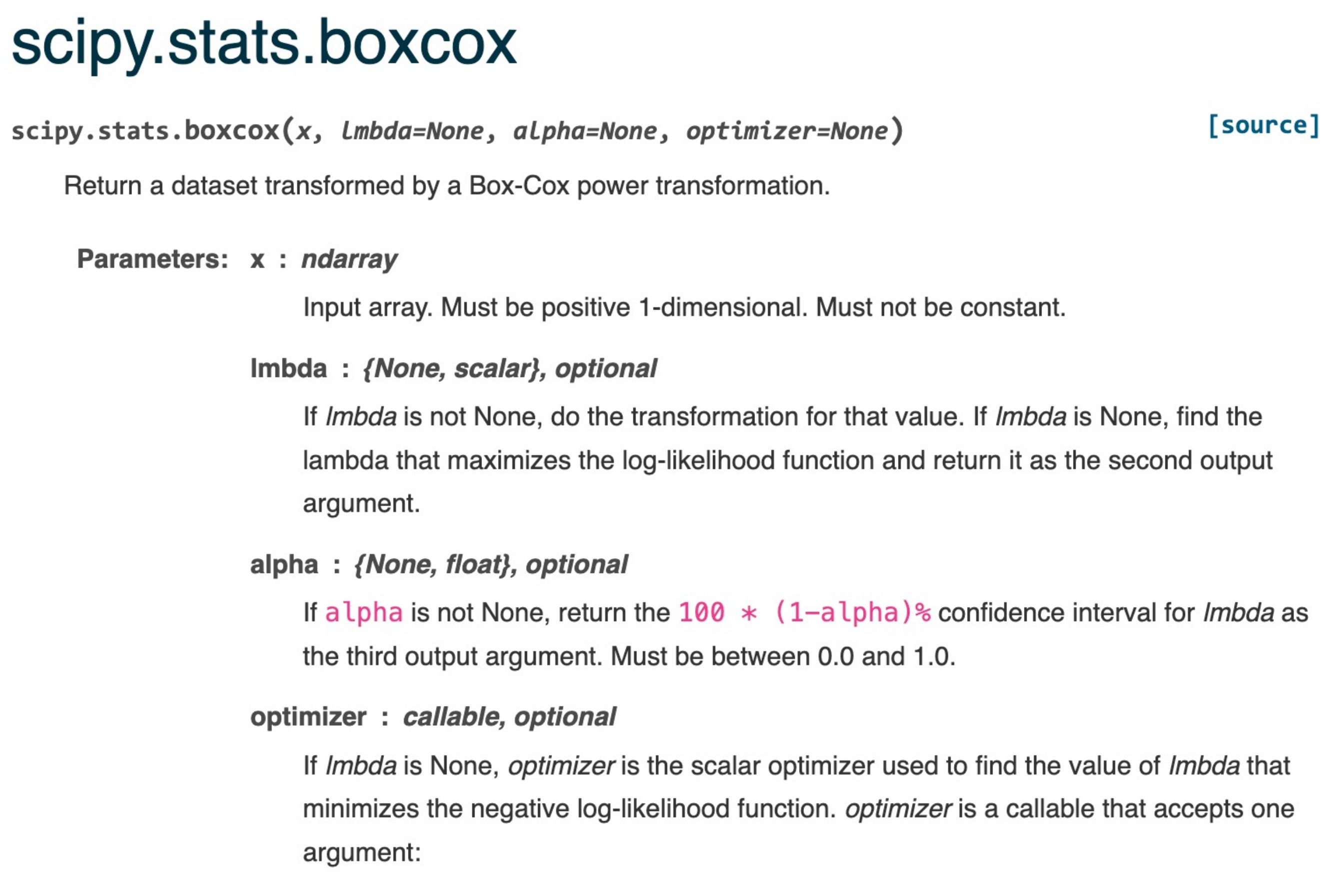
Box Cox transform

- A good value for λ makes the variance the same size across the time series.
- How do we pick a good value for λ?
- Try different λ, plot the data and check visually that the variance is nearly constant.
- Use a method that automatically selects λ that optimizes on some criteria.
  - Maximum likelihood (MLE): Picks λ that makes transformed data look the most normally distributed.
  - Guerrero: Picks λ that tries to make the variance constant across the time series.





Box Cox implementation in Scipy



scipy.stats.boxcox

`scipy.stats.boxcox(x, lmbda=None, alpha=None, optimizer=None)` [source]

Return a dataset transformed by a Box-Cox power transformation.

Parameters: **x** : ***ndarray***

Input array. Must be positive 1-dimensional. Must not be constant.

lmbda : ***{None, scalar}, optional***

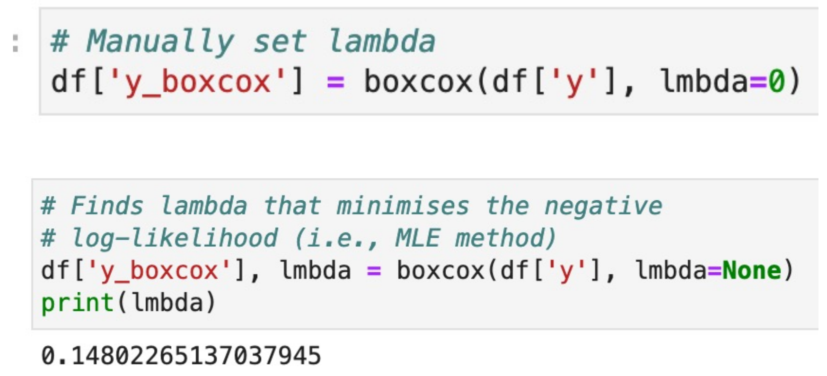
If *lmbda* is not None, do the transformation for that value. If *lmbda* is None, find the lambda that maximizes the log-likelihood function and return it as the second output argument.

alpha : ***{None, float}, optional***

If `alpha` is not None, return the `100 * (1-alpha)%` confidence interval for *lmbda* as the third output argument. Must be between 0.0 and 1.0.

optimizer : ***callable, optional***

If *lmbda* is None, *optimizer* is the scalar optimizer used to find the value of *lmbda* that minimizes the negative log-likelihood function. *optimizer* is a callable that accepts one argument:

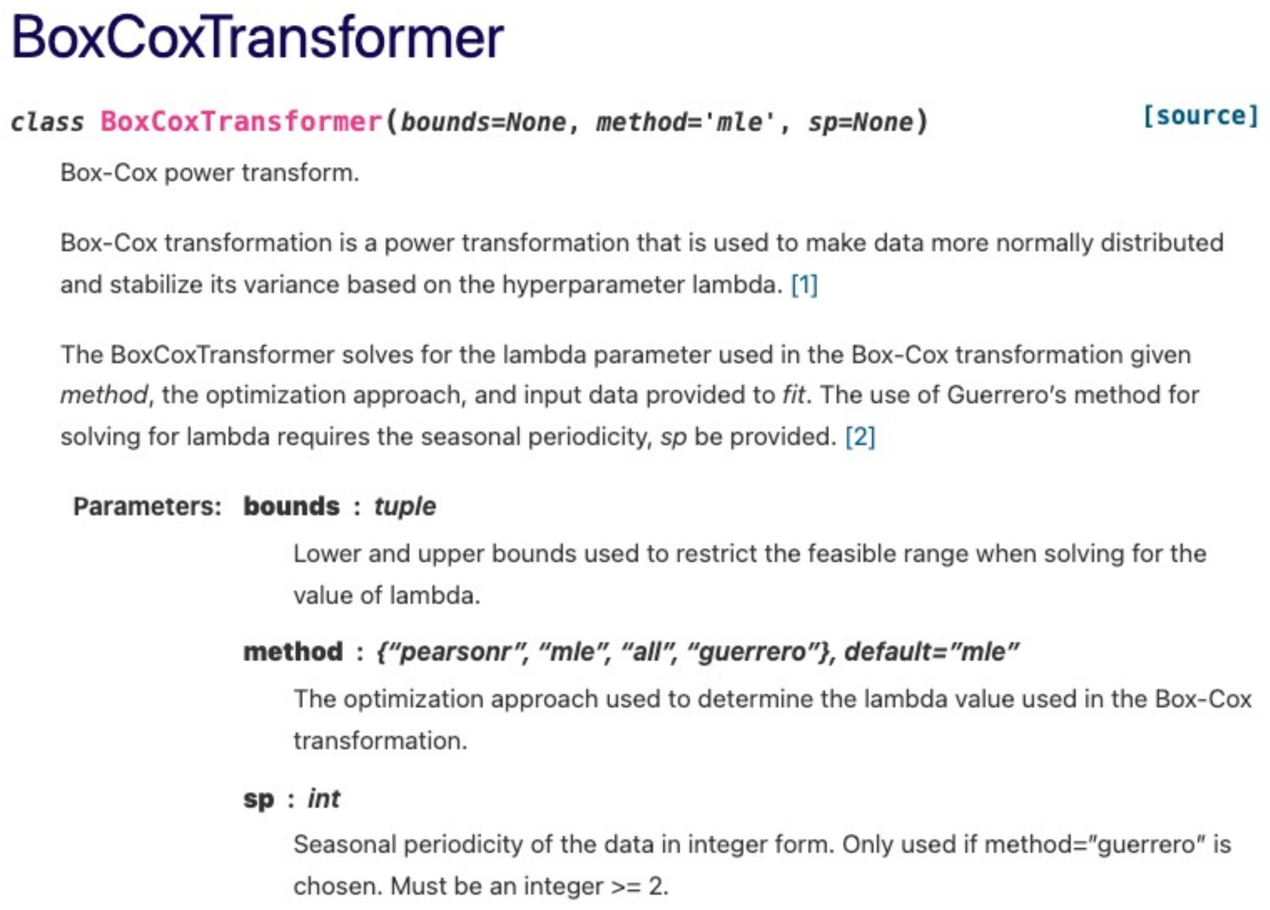


```
# Manually set lambda
df['y_boxcox'] = boxcox(df['y'], lmbda=0)
```

```
# Finds lambda that minimises the negative
# log-likelihood (i.e., MLE method)
df['y_boxcox'], lmbda = boxcox(df['y'], lmbda=None)
print(lmbda)
```

```
0.14802265137037945
```

Box Cox implementation in sktime



BoxCoxTransformer

class **BoxCoxTransformer**(*bounds=None*, *method='mle'*, *sp=None*) [source]

Box-Cox power transform.

Box-Cox transformation is a power transformation that is used to make data more normally distributed and stabilize its variance based on the hyperparameter lambda. [1]

The BoxCoxTransformer solves for the lambda parameter used in the Box-Cox transformation given *method*, the optimization approach, and input data provided to *fit*. The use of Guerrero's method for solving for lambda requires the seasonal periodicity, *sp* be provided. [2]

Parameters: **bounds** : *tuple*

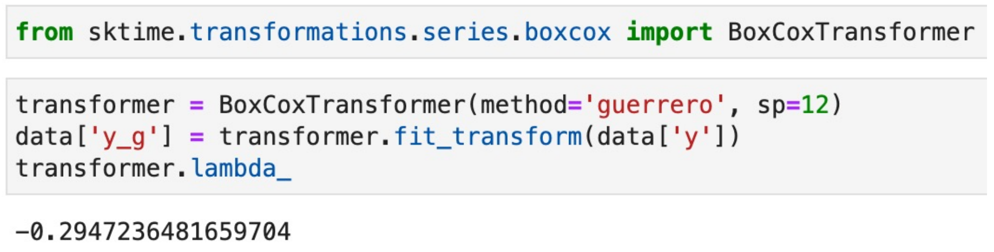
Lower and upper bounds used to restrict the feasible range when solving for the value of lambda.

method : *{"pearsonr", "mle", "all", "guerrero"}, default="mle"*

The optimization approach used to determine the lambda value used in the Box-Cox transformation.

sp : *int*

Seasonal periodicity of the data in integer form. Only used if method="guerrero" is chosen. Must be an integer >= 2.



```
from sktime.transformations.series.boxcox import BoxCoxTransformer
```

```
transformer = BoxCoxTransformer(method='guerrero', sp=12)
data['y_g'] = transformer.fit_transform(data['y'])
transformer.lambda_
```

```
-0.2947236481659704
```

Summary

Forecasting and decomposition methods sometimes work better if the variance is stable across the whole time series.

Log and power transforms can help stabilize the variance across the time series.

A Box Cox transform combines the log and power transform into a single method with a parameter λ.

The best λ is the one that makes the variance stable across the time series.

Multiple methods exist to automatically select the best λ, however, they don't always agree and manual sense checking is advised.