Time series analysis in Python

A step-by-step guid from basic to advanced

Khaled DELLAL

Time series analysis in Python

A step-by-step guid from basic to advanced

Khaled DELLAL

This book is available at https://leanpub.com/python-pedia

This version was published on 2025-06-10

Leanpub

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Tweet This Book!

Please help Khaled DELLAL by spreading the word about this book on Twitter!

The suggested hashtag for this book is #time_series_with_python.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#time_series_with_python

Contents

Trend analysis

Trend is the first thing a time series shows, at first glince we can notice that the series tends to go up, down or in an unclear direction. This chapter constitutes an answer to the question : How best we deal with **trend** effect ? But when can we say that a series is suffuring from a trend effect?

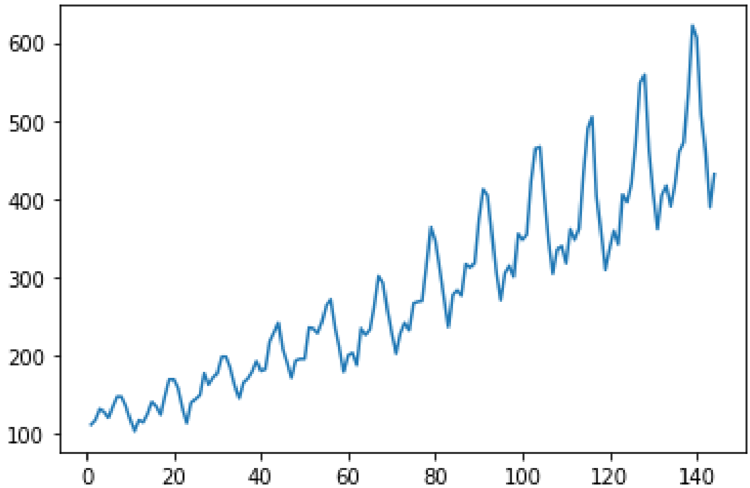
Linear trend effect

The main purpose of trend removal is to make a time series stationary in order to optimise the forecasting process.

The aim of this chapter is to perform the first step in time series forecasting: trend removal. We present trend effect removal when it takes a liear form.

To illustrate the how-to of this method we use a publicly avaialble historical data of Air Passengers between 1949 and 1960.

As example : we take the Air Passengers time series data as example, and we will remove its trend effect.



Number of passengers

For the linear trend is the simplest case of trend effect, first thing to do is
regressing the time series over the time variable.

Linear regression on trend

In the econometric jargon the variable of interest 'Y' is the monthly Air Passengers number. Explanatory variable is 't' here is the time vector.
Let's take a look at the two series :
Y : 112, 118, 132, …
t : 1, 2, 3, …

Reading the dataset

```
time_series=pd.read_csv("AirPassengers.csv")
Y=list(time_series["#Passengers"])
t=[t for t in range(1, len(time_series)+1)]
print("Y : ", Y)
print("t : ", t)
```

To perform a linear regression in Python we have to import the function linregress from the package scipy.stats:

Regression of Y over t

```
from scipy.stats import linregress

result=linregress(t, Y)
result
```

```
[13]: LinregressResult(slope=2.657183908045977, intercept=87.65277777777777, rvalue=0.923925411
      2768995, pvalue=4.020274506593391e-61, stderr=0.0923324695655315, intercept_stderr=7.7163
      46990275525)
```

Regression outputs

From the outputs we need only the slope and the intercept, the linear trend fuction has the following expression:

$Y = b \times t + a$

The coefficent ‘b’ represents the value of slope and ‘a’ represents the intercept value:

Storing ‘slope’ and ‘intercept’ values

```
b=result.slope
a=result.intercept
```

Linear trend removal

In this step we show how to store the trend-frees series, the operation is not other than substract the equation above from the orinial Y series.

First we proceed with creating the trend series:

```
# We create the trend series
trend=[b*i+a for i in t]
```

Let's name the free-trend series : $Y_f t$ and proceed with its creating as follows:

```
# We create the Y_ft series
Y_ft=[Y[i]-trend[i] for i in range(len(Y))]
```

Before plotting the de-trended series we create a function allowing us to save many lines of code later when we need some charts to be displayed.

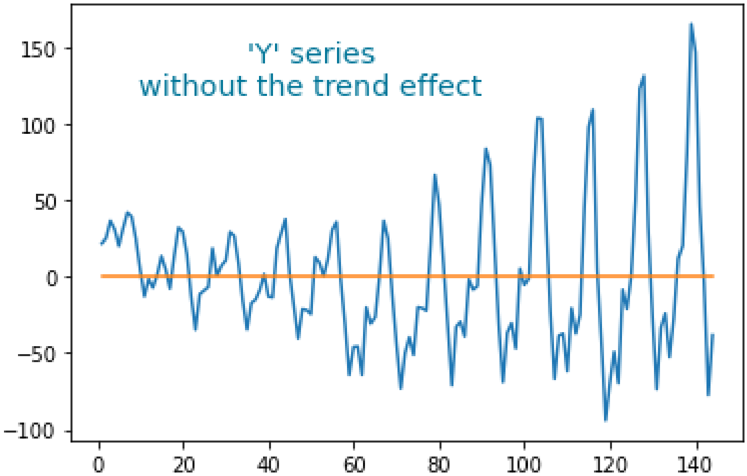
Function to plot the series

```
# We set a function allowing us to plot our time series

def plot_series(series, text='Time series', zero_base_lin\
e=True):
    """

    Parameters:
    ----------
    series : Name of the time series
    text : Text to be displayed on the chart
    zero_base_line : set to True when we dezire to show t\
he x-axis

```

```
    Returns:
    -------
    A displayed chart
    """

    t=[i for i in range(1, len(series)+1)]

    base_line=[0 for i in t]

    text_coordinates_X= int(len(series)/4)
    text_coordinates_Y= int(max(series))

    if zero_base_line:
        plt.plot(t, series)
        plt.plot(t, base_line)
        plt.text(text_coordinates_X,text_coordinates_Y,te\
xt ,
                 rotation=0,
                 horizontalalignment='center',
                 verticalalignment='top',
                 multialignment='center',
                 fontsize='xx-large',
                 color='#007598')
    else:
        plt.plot(t, series)
        plt.text(text_coordinates_X,text_coordinates_Y,te\
xt ,
                 rotation=0,
                 horizontalalignment='center',
                 verticalalignment='top',
                 multialignment='center',
                 fontsize='xx-large',
                 color='#007598')
    plt.show
```

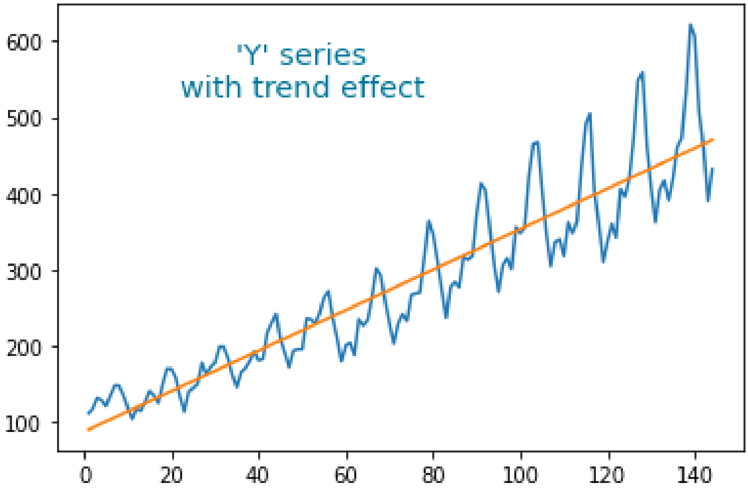
And here we plot the de-trended series:

```
plot_series(Y_ft, text="'Y' series\nwith trend effect", z\
ero_base_line=True)
```



The original time series having a trend effect was as in the following chart:

```
plot_series(Y, text="'Y' series\nwith trend effect", zero\
_base_line=False)
plot_series(trend, text="", zero_base_line=False)
```



Quadratic trend

The trend is quadratic when it has the following form :

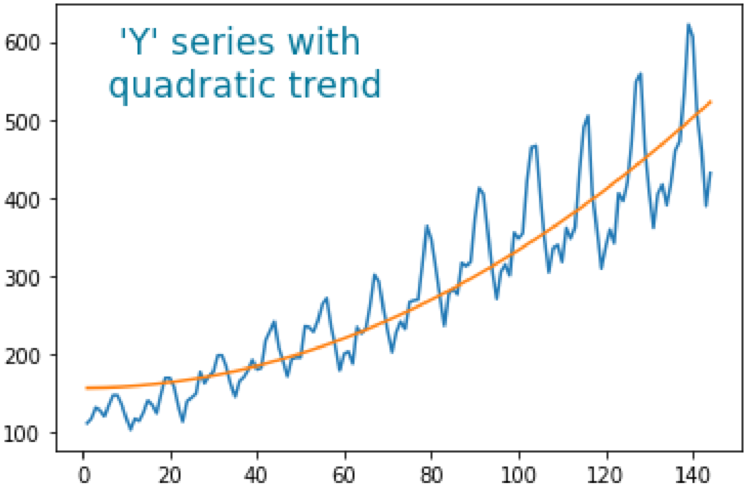
$quad_trend = b \times t^2 + a$

To get the estimation of b and a we have to do a regression between Y and t^2.

```
# We create a series of t^2 values
t_2=[i**2 for i in t]

# We regress Y on t_2 and store the 'result'
result=linregress(t_2, Y)

# We create the series of quadratic trend
quad_trend=[result.slope*t_2[i] + result.intercept for i \
in range(len(t_2))]
```

Y series and its quadratic trend are displayed as follows :

```
plot_series(Y, text="'Y' series with\n quadratic trend", \
zero_base_line=False)
plot_series(quad_trend, text="", zero_base_line=False)
```



Quadratic trend

Exponential trend

We say that the trend of the series is exponontial when the mathematical relationship between Y and t can be expressed like :

$$Y = e^{bt+a}$$

When can we say that the equation above is the adequate specification of the relationship between Y and t ?

To be able to estimate the coefficients b and a here we have to *linearize* that equation, to do that we proceed with a log-transformation as follows :

$$\ln Y = b \times t + a$$

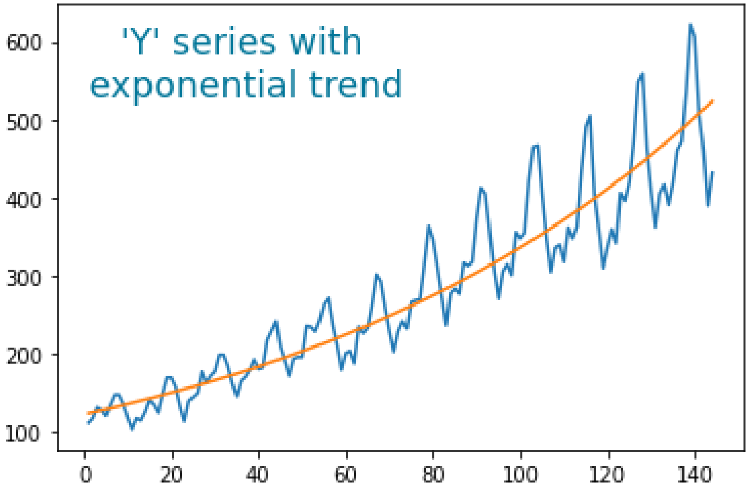
When correlation between lnY and t is higher then correlation between Y and t we can say that the relashionship is more exponential than linear.

The exponential trend series is claculated as follows:

```
# We regress log_Y on t

result=linregress(t, log_Y)

# We create the exponential trend series

exp_trend=[exp(result.slope*i + result.intercept) for i i\
n t]
```

The following chart shows the Y series with the exponential trend

```
plot_series(Y, text="'Y' series with\n exponential trend"\
, zero_base_line=False)
plot_series(exp_trend, text="", zero_base_line=False)
```



Exponential trend

Time series forecasting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/python-pedia.

Choice of the best trend

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/python-pedia.

Trend removal

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/python-pedia.

Forecasting setep

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/python-pedia.