QBUS6840: Tutorial 4 – Time Series Decomposition II

Objectives

- Learn how to decompose time series data into trend, seasonal and cycle components

In our previous tutorial, we have learnt the data manipulation in Time Series, i.e. loading the dataset from a csv file, indexing the data, visualizing the data, moving average and how to check stationary with the help of moving average. In this week, we will further learn how to decompose a time series data by using trend-cycle decomposition.

Here are the main steps we may take:

(1) Analyze the stationarity
(2) Estimate the initial trend-cycle component T_t
(3) Estimate the seasonal component and normalized seasonal index $\overline{S_t}$
(4) Update the estimation of the trend-cycle component $\widehat{T_t}$
(5) If you want to use this decomposition method for forecasting, then you need to combine the normalized seasonal index $\overline{S_t}$ and final trend-cycle $\widehat{T_t}$ together as the forecasting result

In our previous tutorial, we have learnt the fundamental of the stationarity analysis and initial trend-cycle estimation. For today's tutorial, we will mainly focus on the steps 3-4 from above list.

1. Calculate initial trend-cycle estimation by moving average

Continue with the task 5 in our last tutorial. At the end of last tutorial, you have get time series data `ts_res` (which is approximately stationary). Generally, this `ts_res` is the seasonal component $S_t + e_t$. Although you have done this task with correct actions, you still need to pay attention to the following tips.

Firstly, when examining the seasonal peaks (reduce the trend), we use the log transform so that we could penalizes larger values more than the smaller. However, for some other cases, we may also apply different transformation methods such as exponent, or cube root etc.

Secondly, to calculate the initial trend-cycle estimate, a simple but effective way is to do moving average smoothing to reduce the seasonal fluctuation. For even seasonal period `m`, we must do a "2 by `m`"-MA. For any odd seasonal period `m`, we must do a `m`-MA.

Thirdly, the selection of decomposition type is important.
Basically, we have 2 main decomposition type: additive and multiplicative.

- If the seasonal variation is proportional to the trend, then we should

use **multiplicative**. If the seasonal peaks increase in magnitude when the trends magnitude increases and vice versa then it is **proportional**.

- Otherwise we should use **additive**.

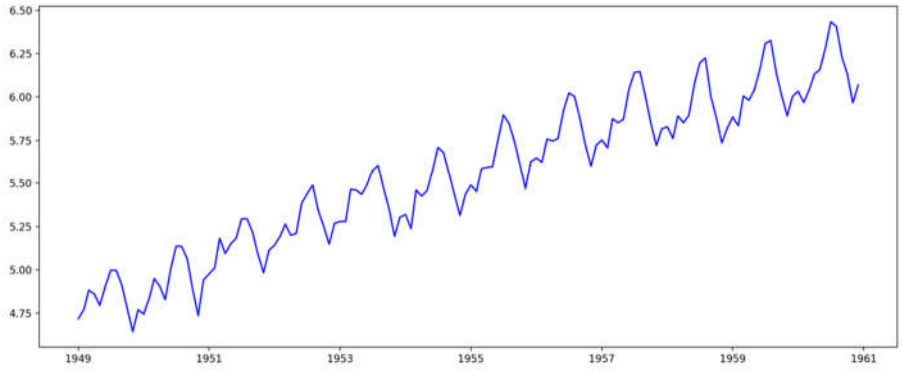
One thing we need to pay attention, for most cases, we will treat the moving average results as the combination of trend and cycle. That is:

$$T_t + C_t = \text{Moving Average}(Y_t)$$

Since in most cases, cycle C_t is hard to model, we will not take this component into account. Therefore:

$$T_t = T_t + C_t$$

Examine the following time-series curve and think about what is the suitable decomposition method?



Once we decide the decomposition method, we then need to calculate seasonal components. In last tutorial, for additive decomposition type, we use:

$$S_t + e_t = Y_t + T_t$$

```
ts_res = ts_log - Trend
plot_curve(ts_res)
```

Question: What if the trend is multiplicative/proportional?

2. Calculate seasonal index

We assume that the season components are relatively stationary throughout the dataset ($S_t = S_{t+m}$ and so on).

So now that we have identified the seasonal components we can eliminate them from the original data, which we can use to create an updated and more accurate trend-cycle estimation.

First take the average of the seasonal components for each month. To do

this, we may need to replace the `nan` components with zeros. Although this is not a good strategy, but simple enough to demonstrate the concept and algorithm. Then pad the data so that we have 144 months (12 years) of observations. Then take the average.

$$\widehat{S_t} = \text{mean}(S_t)$$

```
ts_res_zero = np.nan_to_num(ts_res)
monthly_S = np.reshape(ts_res_zero, (12,12))
monthly_avg = np.mean(monthly_S[1:11,:], axis=0)
```

Before you copy the above code, think about the following questions:

(1) What is the size of `monthly_S`?
(2) What is the meaning of "`1:11`"?
(3) If we don't want to use 0 to replace `nan`, what is your best suggestion? How to implement your suggestion in python code?

Then we should normalize our data in order to obtain a unique solution.
For additive decomposition, we should make the mean of seasonal index equal to zero. Therefore, we should take the mean of the seasonal index out of the original seasonal index.

$$\overline{S_t} = \widehat{S_t} - \text{mean}(\widehat{S_t})$$

```
#%% Normalize the seasonal index
mean_allmonth = monthly_avg.mean()
monthly_avg_normalized = monthly_avg - mean_allmonth
print(monthly_avg_normalized.mean())
```

Question: What if the time series is multiplicative / the magnitude of time series is proportionally changing?

3. Calculate the seasonal adjusted data

In order to calculate the seasonal adjusted data $T_t + \widehat{C_t} + e_t$, you need to have original data Y_t minus by the normalized seasonal index $\overline{S_t}$:

$$T_t + \widehat{C_t} + e_t = Y_t - \overline{S_t}$$

Since the size of original data Y_t and normalized seasonal index $\overline{S_t}$ does not match, we need to repeat the $\overline{S_t}$ over 12 years:

```
tiled_avg = np.tile(monthly_avg_normalized, 12)
```

Subtract the seasonal average from the original data:

```
seasonally_adjusted = ts_log - tiled_avg
```

Plot the seasonally adjusted data:

```
plt.figure()
fig, ax = plt.subplots(4, 1)
ax[0].plot(ts_log)
ax[1].plot(Trend)
ax[2].plot(ts_res)
ax[3].plot(seasonally_adjusted)
ax[0].legend(['log'], loc=2)
ax[1].legend(['trend'], loc=2)
ax[2].legend(['seasonality'], loc=2)
ax[3].legend(['seasonal adjusted'], loc=2)
```

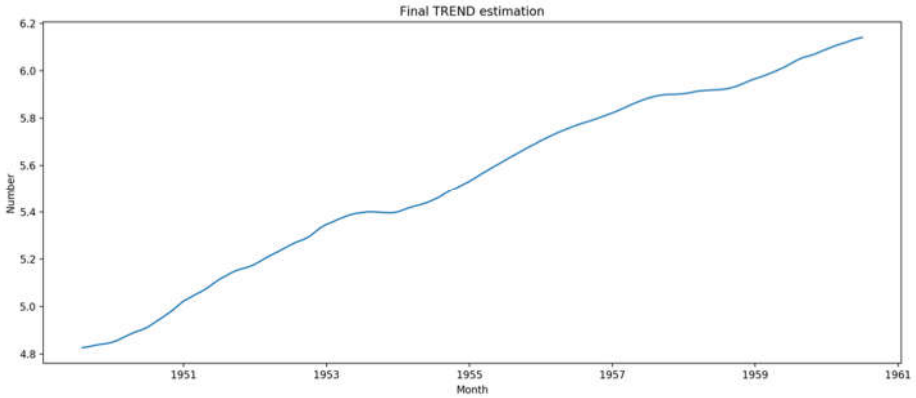
4. Update the trend-cycle

Finally, we need to re-estimate the trend-cycle component based on the seasonally adjusted data $T_t + \widehat{C_t} + e_t$.

In here, there are many ways to update the trend-cycle component $\widehat{T_t}$.
You can use a linear representation to fit the data, or use a moving average method to regenerate the trend-cycle. For simplicity, we use a "2 by m" moving average to update the trend-cycle:

```
T_final = seasonally_adjusted.rolling(12, center
=True).mean().rolling(2, center = True).mean()
plt.figure()
plt.plot(T_final)
plt.title('Final TREND estimation')
plt.xlabel('Month')
plt.ylabel('Number')
```

Check the output of your program and compare with this:



Now you have successfully extract the major components (trend-cycle, and seasonal index) from a time-series data.

If you are interested in topic and wish to learn more, please go to https://medium.com/@stallonejacob/time-series-forecast-a-basic-introduction-using-python-414fcb963000

When using pandas' mean, please make sure not counting NaNs in the detrended series.