

Qbus 6840 TUT 4 Time Series Decomposition





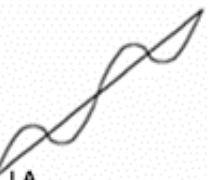




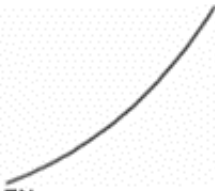
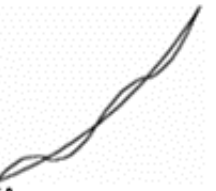
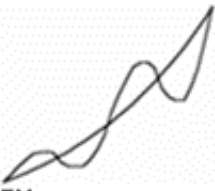
第四周在第三周的基础上继续分解 Time Series， 分解 seasonality

Additive Vs Multiplicative

当时间数列图显示的时间数列的季节变动大致相等时， 或时间数列图随时间推移等宽推进时， 采用**加法模型**

当时间数列图显示的时间数列的季节变动与时间数列的长期趋势大致成正比是， 采用**乘法模型**

乘法模型的 TS 图大致呈喇叭状或发射状

	Nonseasonal	Additive Seasonal	Multiplicative Seasonal
	(SIMPLE)		
Constant Level	 NN	 NA	 NM
Linear Trend	(HOLT)  LN	 LA	(WINTERS)  LM
Damped Trend (0.95)	 DN	 DA	 DM
Exponential Trend (1.05)	 EN	 EA	 EM

如果数据中的模式不很明显， 并且在加法模型和乘法模型之间进行选择还有困难， 则可以尝试两种模型， 然后选择准确度度量较小的模型。

Additive model 和 Multiplicative model 都是用来解释 TS 的组成部分

$$y_t = f(T_t, S_t, C_t, e_t)$$

- T_t : Trend component
- S_t : Seasonal component
- C_t : Cycle component
- e_t : Residual component

Additive model 的分解公式：

$$y_t = T_t + S_t + C_t + e_t$$

Multiplicative model 的分解公式：

$$y_t = T_t * S_t * C_t * e_t$$

一些化简处理

1. Tutorial 3 中用 rolling 函数计算的 Moving Average(Y_t)

For most cases, we will treat the moving average results as the combination of trend and cycle.

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

Multiplicative model 对应的：

$$T_t * C_t = \text{MovingAverage}(Y_t)$$

2. 省略 cycle component

In most cases, cycle C_t is hard to model, we will not take this component into account.

$$T_t = T_t + C_t$$

Multiplicative model 对应的：

$$T_t * C_t = T_t$$

计算 seasonal index

数学推导

由于 Tutorial 3 最后

```
1 | ts_res = ts_log - Trend
```

得到的 `ts_res` 代表的是 Seasonal Component $S_t + e_t$

根据 lecture3 Page 19 的说法，由于 season components 是 stationary 的 (Tutorial 3 最后的Dickey-fuller Test 测试)

- The usual assumption (under monthly data) is that

$$S_t = S_{t-12} = S_{t+12} \Rightarrow \hat{S}_t = \bar{S}_m$$

where \bar{S}_m is called seasonal index and they ($m = 1, 2, \dots, M$) are the **normalized** average of all observations in the month m of historical data, i.e.,

$$\sum_{m=1}^M \bar{S}_m = M.$$

根据 lecture3 P26 的计算过程

- Calculate the normalized constant by

$$c = \frac{12}{\text{sum of All month values}} \\ = 12 / (0.992 + \dots + 0.808) = 1.00026$$

- Month indices

$$\bar{S}_1 = 0.9446 * 1.00026 = 0.9449, \dots, \\ \bar{S}_{12} = 1.3926 * 1.00026 = 1.3930$$

- The seasonal component is

$$\{\hat{S}_1, \hat{S}_2, \dots, \hat{S}_{12}, \hat{S}_{13}, \dots, \hat{S}_{24}, \dots\} \\ = \{\bar{S}_1, \bar{S}_2, \dots, \bar{S}_{12}, \bar{S}_1, \dots, \bar{S}_{12}, \dots\}$$

- For monthly data, there are only 12 different \hat{S}_t values. For example, when $t = 37$, we know this is a January, so $\hat{S}_t = \bar{S}_{Jan}$ (ie. $m = 1$)

Index	year1	year2	year3	Average	Normalized
Ju1	0.975532	0.978562	1.02302	0.992371	0.992626
Aug	0.969948	0.969893	0.933196	0.957679	0.957924
Sep	0.916178	0.901172	0.885252	0.900867	0.901098
Oct	0.990678	0.980837	0.998444	0.98972	0.989973
Nov	1.07478	1.04341	1.03319	1.05046	1.05073
Dec	1.39218	1.41591	1.36988	1.39266	1.39301
Jan	0.921949	0.989406	0.922632	0.944662	0.944904
Feb	1.15115	1.12518	1.20782	1.16138	1.16168
Mar	1.11622	1.05857	1.12171	1.09883	1.09912
Apr	0.86493	0.929584	0.910456	0.901657	0.901888
May	0.773929	0.824791	0.796933	0.798551	0.798756
Jun	0.836715	0.79469	0.79285	0.808085	0.808292

lecture 里遇到的是 Multiplicative model 的数据,

$$\hat{s}_t = \text{mean}(s_t) \\ c = \frac{1}{\text{mean}(\hat{s}_t)} \\ \bar{s}_m = \hat{s}_t * c = \frac{\hat{s}_t}{\text{mean}(\hat{s}_t)}$$

类比成 Additive model 的话

$$\hat{s}_t = \text{mean}(s_t) \\ \bar{s}_m = \hat{s}_t - \text{mean}(\hat{s}_t)$$

s_t 指的是每个月的数据, $t = [1, 12]$

计算 seasonal index 需要我们先求 每个月 多年数据的平均值，再减去每个月平均值的平均值。

代码实现

1. 由于 `ts_res` 中存在 NaN 的值，在使用 `mean()` 函数时，数据里是不能有 NaN 值。我们需要进行数据的填充，一种方式是 Tutorial 中的使用 [nan_to_num函数](#) 用 0 填充NaN 数据。在该情景下，使用平均值用来填充更为合适

```
1 # Tutorial
2 ts_res_zero = np.nan_to_num(ts_res)
3
4 # 改进
5 ts_res_zero = ts_res.fillna(ts_res.mean())
```

2. 用 [reshape函数](#)将数据整形成 12 * 12 的方阵，每一行代表每一年12个月每个月的数据(S_t)

`numpy.reshape(a, newshape)`

- a 被整形的 array 或 matrix
- newshape: 整形后的 size, 如(2, 3) 二行三列

```
1 monthly_S = np.reshape(ts_res_zero, (12,12))
```

3. 用 [mean函数](#) 求每个月平均值

因为 第0年和第11年的数据是用 0 填充的 NaN 值，在计算平均值时不计算，所以选择[1:11,:] 第一行到第11行的所有列

`numpy.mean(a, axis=None)`

- a 要被求平均数的数据
- axis: 0 垂直方向求平均值(每列求平均), 1 水平方向求平均值(每行求平均)

- 4.减去每个月平均值的平均值

```
1 mean_allmonth = monthly_avg.mean()
2 monthly_avg_normalized = monthly_avg - mean_allmonth
3 print(monthly_avg_normalized.mean())
```

计算 seasonal adjusted data

数学推导

类比于 Page27 的 Multiplicative model:

$$T_t \times \widehat{C_t} \times e_t = \frac{Y_t}{\hat{S}_t} = \frac{Y_t}{\bar{S}_m}$$

Additive model:

$$T_t + \widehat{C_t} + e_t = Y_t - \hat{S}_t = Y_t - \bar{S}_m$$

代码实现

1. 用 `numpy.tile` 函数复制 \bar{S}_m (1年) 成 12 年的 matrix

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

`numpy.tile(A, reps)`

- A 被复制的 array
- reps: 复制的格式, e.g. 2 - 复制遍, (2, 3)复制成 2*3 个 A 组成的矩阵

2. 用 tutorial 3 里被 log 处理过的原数据 (Y_t) 减去 这个矩阵

```
1 seasonally_adjusted = ts_log - tiled_avg
```

更新 trend-cycle

在原数据中减去了 seasonality 的 Seasonally Adjusted Series ($T_t + \widehat{C_t} + e_t$) 我们需要再次进行 estimate

re-estimate 的方式:

- Tutorial4 里使用的 Rolling(也就是求 moving average)
- Lecture 3/4 中的 linear regression

```
1 T_final = seasonally_adjusted.rolling(12, center = True).mean().rolling(2,
    center = True).mean()
```

补充：直接用函数分解 y_t

在 tutorial 4 最后提供的网页中提供了直接调用库函数分解 y_t 的办法 [seasonal_decompose](#) 函数

`statsmodels.tsa.seasonal.seasonal_decompose(x, model)`

- x 待分解的 TS
- model: "additive" 或者 "multiplicative"
- 返回值 seasonal, trend, and resid attributes

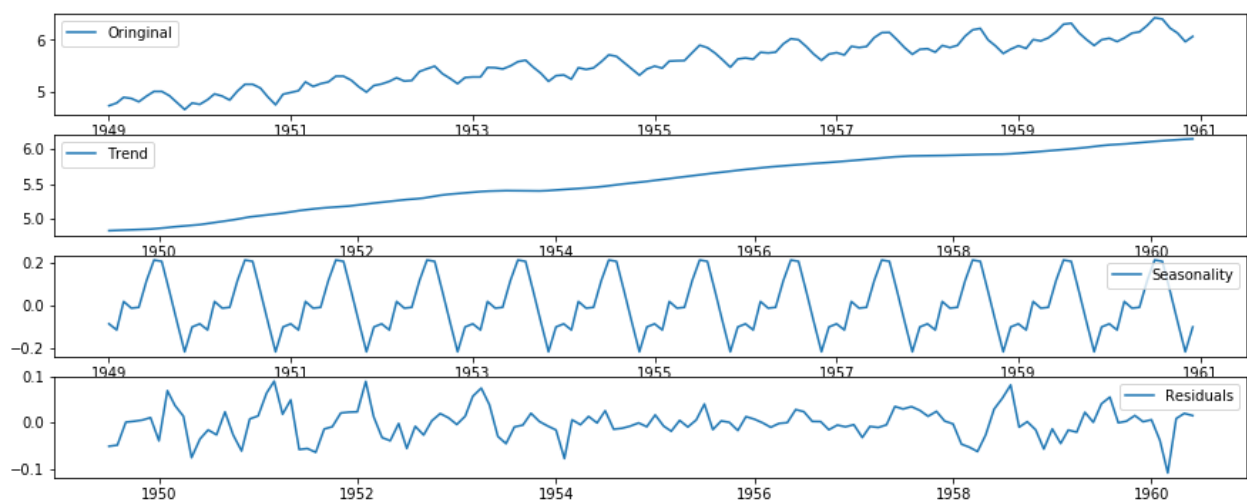
```
1 from statsmodels.tsa.seasonal import seasonal_decompose
2
3 decomposition = seasonal_decompose(ts_log)
4
5 trend = decomposition.trend
6 seasonal = decomposition.seasonal
7 residual = decomposition.resid
8
```

```

9 plt.subplot(411)
10 plt.plot(ts_log, label = "Original")
11 plt.legend(loc = 'best')
12 plt.subplot(412)
13 plt.plot(ts_log, label = "Trend")
14 plt.legend(loc = 'best')
15 plt.subplot(413)
16 plt.plot(ts_log, label = "Seasonality")
17 plt.legend(loc = 'best')
18 plt.subplot(414)
19 plt.plot(ts_log, label = "Residuals")
20 plt.legend(loc = 'best')

```

结果：



从时间序列中删除 Trend 和 Seasonality，得到 Residuals