Tutorial 3

Time series 的分解，分解 Trend

Moving average

时间序列的数值由于受周期变动和随机波动的影响，起伏较大，不易显示出事件的发展趋势时，使用 Moving average 可以消除这些因素的影响，显示出事件的发展方向与趋势（即趋势线），然后依趋势线分析预测序列的长期趋势。

Simple Moving Average(SMA)

是某变数之前n个数值的未作加权算术平均。例如，收市价的10日简单移动平均指之前10日收市价的平均数。若设收市价为 p_1 至 p_n，则方程式为：

$SMA = \frac{p_1+p_2+\cdots+p_n}{n}$

dataFrame.rolling() 函数

在 python 中做 Moving average 用的是 dataFrame.rolling() 函数

`dataFrame.rolling(window, center = False)`

- window 移动窗口数
- center 窗口对应的时间坐标位置是否在中间，默认窗口的坐标在最右边(不理解先看例子)

例子1 解释 center 参数

生成两组周期分别为 4 和 5 的数据 t1 和 t2

```
t1 = pd.Series([1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4])
t2 = pd.Series([1,2,3,4,5,1,2,3,4,5,1,2,3,4,5,1,2,3,4,5])
```

```
t2.rolling(5).mean()
Out[2]:
0     NaN
1     NaN
2     NaN
3     NaN
4     3.0
5     3.0
6     3.0
7     3.0
8     3.0
9     3.0
10    3.0
```

```
11    3.0
12    3.0
13    3.0
14    3.0
15    3.0
16    3.0
17    3.0
18    3.0
19    3.0
dtype: float64
```

rolling 函数其实就是将数据滚动生成许多个 list：

如果 center = False, 窗口坐标在最右边

组1： [NaN NaN NaN NaN 1]

组2： [NaN NaN NaN 1 2]

组3： [NaN NaN 1 2 3]

组4： [NaN 1 2 3 4]

组5： [1 2 3 4 1]

...

> 如果要求 moving agverage 就对各个 list 中的数据进行求平均值（min）

```
t2.rolling(5,center = True).mean()
Out[3]:
0     NaN
1     NaN
2     3.0
3     3.0
4     3.0
5     3.0
6     3.0
7     3.0
8     3.0
9     3.0
10    3.0
11    3.0
12    3.0
13    3.0
14    3.0
15    3.0
16    3.0
17    3.0
18    NaN
19    NaN
dtype: float64
```

如果 center = True, 窗口坐标在中间

组1： [NaN NaN 1 2 3]

组2： [NaN 1 2 3 4]

组3： [1 2 3 4 5]

...

组-3： [1 2 3 4 5]

组-2： [2 3 4 5 NaN]

组-1： [3 4 5 NaN NaN]

```
t1.rolling(4,center = True).mean()
Out[9]:
0     NaN
1     NaN
2     2.5
3     2.5
4     2.5
5     2.5
6     2.5
7     2.5
8     2.5
9     2.5
10    2.5
11    2.5
12    2.5
13    2.5
14    2.5
15    2.5
16    2.5
17    2.5
18    2.5
19    NaN
dtype: float64
```

当窗口为偶数，center = True 时窗口对应的时间坐标位置在中间偏后的位置

组1： [NaN NaN 1 2]

组2： [NaN 1 2 3]

组3： [1 2 3 4]

组4： [2 3 4 1]

组5： [3 4 1 2]

...

组-2： [1 2 3 4]

组-1： [2 3 4 NaN]

例子2 解释 偶数周期数据用 “2 by m”-MA 计算

```
t1.rolling(4,center = True).mean().rolling(2,center = True ).mean()
Out[10]:
0     NaN
1     NaN
2     NaN
3     2.5
4     2.5
5     2.5
6     2.5
7     2.5
8     2.5
9     2.5
10    2.5
11    2.5
12    2.5
13    2.5
14    2.5
15    2.5
16    2.5
17    2.5
18    2.5
19    NaN
dtype: float64
```

> 奇数周期的 TS 求 MA 只需 window 和周期相同，偶数周期的 TS 求 MA 需要先 window = 周期 处理一次后再 window = 2 再处理一次推导

偶数周期的数据的 moving average：

设周期为4：

$\tilde{x_i} = \frac{\frac{1}{2}x_{i-2}+x_{i-1}+x_i+x_{i+1}+\frac{1}{2}x_{i+2}}{4}$

以 t2 为例:

window = 4 处理后

组1的 mean = NaN

组2的 mean = NaN

组3的 mean： $\frac{x_1+x_2+x_3+x_4}{4}$

组4的 mean： $\frac{x_2+x_3+x_4+x_5}{4}$

组5： $\frac{x_3+x_4+x_5+x_6}{4}$

再对这个分组结果 使用 window = 2 的 Rolling 处理：

新组1的 mean： NaN

新组2 的 mean: NaN

新组3 的 mean： $\frac{x_1+2x_2+2x_3+2x_4+x_5}{2}$

新组4 的 mean： $\frac{x_2+2x_3+2x_4+2x_5+x_1}{2}$

Making the time series stationary

第三周的 lecture 的内容主要是对 TS 的 Smooth， 只有 stationary 的 TS 我们才好进行下一步的预测

而影响 TS stationary 的两个主要原因是：

- Trend 均值可以随时间增长或减少
- Seasonality 周期性数值变化

Additive model 的分解公式：

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

Reduce Trend

用 Tranformation 降低 trend , 就是用 log sqrt (用于随时间均值增长的数据) 和 exp（用于随时间均值减少的数据）等方式处理数据（Exponential smooth）得到 y_t

```
ts_log= np.log(ts)
```

Reduce Seasonality

根据 lecture 3 Page 13 的说法，我们忽略掉 Cycle 的影响

> Trend and cycle components are often combined:
>
> Trend-cycle components (TC_t)
>
> Or equivalently assume
>
> C_t = 0 in Additive Model C_t = 1 in Multiplicative Model

用 moving average 为有噪声条件下的 trend 建模(moving average 用来 消除噪声影响)，得到 $\widehat{T_t \times C_t}$

类比 Page 15 的 Multiplicative Model 推导：

$$\widehat{S_t \times e_t} = \frac{y_t}{\widehat{T_t \times C_t}}$$

Additive Model 下：

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

用 Tranformation 处理过的原数据 y_t 减去 trend, 得到 $\widehat{S_t + e_t}$

```
Trend = ts_log.rolling(2, center = True .mean().rolling(12,center =
True).mean()

ts_res = ts_log - Trend
```

Checking the stationarity

检查处理过的数据是不是 stationary ， 可以通过Dickey-fuller Test 测试

这个函数在statsmodels.tsa.stattools 库中

Dickey-fuller Test

`adfuller(TS, regression, autolag='AIC')`

- TS：被检测的 TS
- regression：Dickey-fuller Test 的 type:

参数	Type		
'c'	Type 1	constant, no trend	$\Delta y_i = \beta 0 + \beta 1\ y_{i-1} + \varepsilon_i$
'ct'	Type 2	Constant and trend	$\Delta y_i = \beta 0 + \beta 1\ y_{i-1} + \beta 2\ i + \varepsilon_i$
'nc'	Type 0	no constant, no trend	$\Delta y_i = \beta 1\ y_{i-1} + \varepsilon_i$

每一 Type 下的 Dickey-fuller table 中的Critical Value 都是不同的。

因为Tutorial 的例子已经去掉了 trend, 但是数据一直在一个范围下浮动所以有 constant 选择 Type1

函数返回的是 Dickey-fuller Test 的各个统计值

Tutorial 提供了函数`test_stationarity`，将 Dickey-fuller Test 返回的统计值打印出来

Tutorial 例子经过 Dickey-fuller Test 的结果：

```
Results of Dickey-Fuller Test:
Test Statistic                  -3.779371
p-value                          0.003126
#Lags Used                      13.000000
Number of Observations Used    118.000000
Critical Value (1%)             -3.487022
Critical Value (5%)             -2.886363
Critical Value (10%)            -2.580009
dtype: float64
```

判断 stationary 的方法：Test Statistic 和 Critical Value 比较

如果 Test Statistic < Critical Value (1%) 则有 99% 可信度 series 是 stationary 的。