# Quantitative methods in finance - a brief mention of time series seasonality

Eric Vansteenberghe

November 14, 2016

## Contents



Link to the source codes and the data sets used in this lecture

## 1 Introduction

This exercise is largely inspired by this website and the book **Econometric models and economic forecasts** from Robert Pindyck and Daniel Rubinfeld published in 1998.

## 2 Data set 1

We start with the simple data set proposed by Regis Bourbonnais you can download here. The file is called C3EX6.
As seen in the previous exercises, we download the data and clean it:

```
import pandas as pd
import os
os.chdir('/Users/skimeur/Google␣Drive/empirical_finance')

df=pd.read_excel('R_data/C3EX6.xls',sheetname='C3EX6',header=0,index_col=0)

dates = pd.date_range('2010-01', '2015-01', freq='Q')
```

df.index=dates

ventes=df['VENTES']

# 3 Decomposition principle

Seasonal adjustment rely on the fact that each time series $y_t$ can be decomposed as:

$$y_t = L_t \times S_t \times C_t \times I_t \tag{1}$$

with

- $L$ the long-term secular trend in series
- $S$ the seasonal component
- $C$ the long-term cyclical component
- $I$ the irregular component

The aim is to decompose the time series, isolate and combine $L_t \times C_t$ (estimate it) then we can deduct an estimate of $S_t \times I_t$.
If we have monthly data, we estimate:

$$\hat{L}_t \times \hat{C}_t = \frac{1}{12}\left(y_{t+6} + \ldots + y_{t-5}\right) \tag{2}$$

Then:

$$\hat{S}_t \times \hat{I}_t = \frac{y_t}{\hat{L}_t \times \hat{C}_t} \tag{3}$$

Then $I$ has to be eliminated, we average each $\hat{S}_t \times \hat{I}_t$ element is average per month ($\tilde{z}_1$ for January, $\tilde{z}_2$ February, etc.). Those terms should sum to 12. If they do not, we define $\bar{z}_i = \tilde{z}_i \frac{12}{\sum_j \tilde{z}_j}$.
Finally, the seasonally adjusted series are:

- $y_1^a = \frac{y_1}{\bar{z}_1}$
- $\ldots$
- $y_{12}^a = \frac{y_{12}}{\bar{z}_{12}}$

## 3.1 Decomposition with python

As described in the seasonal_decompose description, here, in fact we are using the additive model :

$$Y_t = T_t + S_t + \epsilon_t \tag{4}$$

So what is called here "*trend*", $T_t$ is in fact the time series without its seasonal component.
One should not mix this with the actual trend and cycle, so in an additive model, we defined in equation 1:

$$Y_t = L_t + S_t + C_t + I_t \tag{5}$$

so $T_t = L_t + C_t$
We can use the simple package:

**from** statsmodels.tsa.seasonal **import** seasonal_decompose

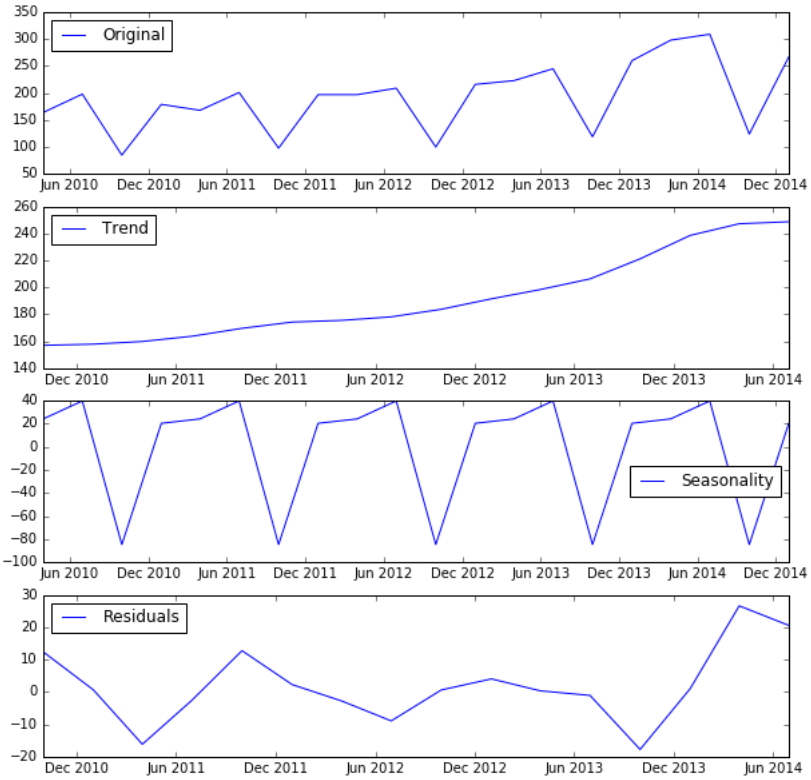
We can then exctrat the trend of the series, its seasonal component and the residuals:

```
decomposition = seasonal_decompose(ventes)

trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid
```

We can then plot the decomposition:

```
import matplotlib.pylab as plt

fig = plt.figure()
fig.set_figheight(10)
fig.set_figwidth(10)
ax1 = fig.add_subplot(4,1,1)
ax1.plot(ventes, label='Original')

ax2 = fig.add_subplot(4,1,2)
ax2.plot(trend, label='Trend')

ax3 = fig.add_subplot(4,1,3)
ax3.plot(seasonal,label='Seasonality')

ax4 = fig.add_subplot(4,1,4)
ax4.plot(residual, label='Residuals')
fig.savefig('bour_season.png')
```

We obtain:




## 3.2 Residual stationarity test

As define in our proposed source, we use the stationarity test function:

```
from statsmodels.tsa.stattools import adfuller
def test_stationarity(timeseries):

    #Determing rolling statistics
    rolmean = pd.rolling_mean(timeseries, window=12)
    rolstd = pd.rolling_std(timeseries, window=12)

    #Plot rolling statistics:
    orig = plt.plot(timeseries, color='blue',label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)

    #Perform Dickey-Fuller test:
    print('Results of Dickey-Fuller Test:')
```

```
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test␣Statistic','p−value','#Lags␣Used','Number␣of␣Observations␣Use
    for key,value in dftest[4].items():
        dfoutput['Critical␣Value␣(%s)'%key] = value
    print(dfoutput)
```

And apply it to our residuals:

```
ventes_decompose = residual
ventes_decompose.dropna(inplace=True)
test_stationarity(ventes_decompose)
```

We do not reject the null hypothesis $H_0$ of non-stationarity.

# 4 Data set 2

We focus on the coffee world prices with 240 observation for Brazil downloadable here. We import that data set as usual:

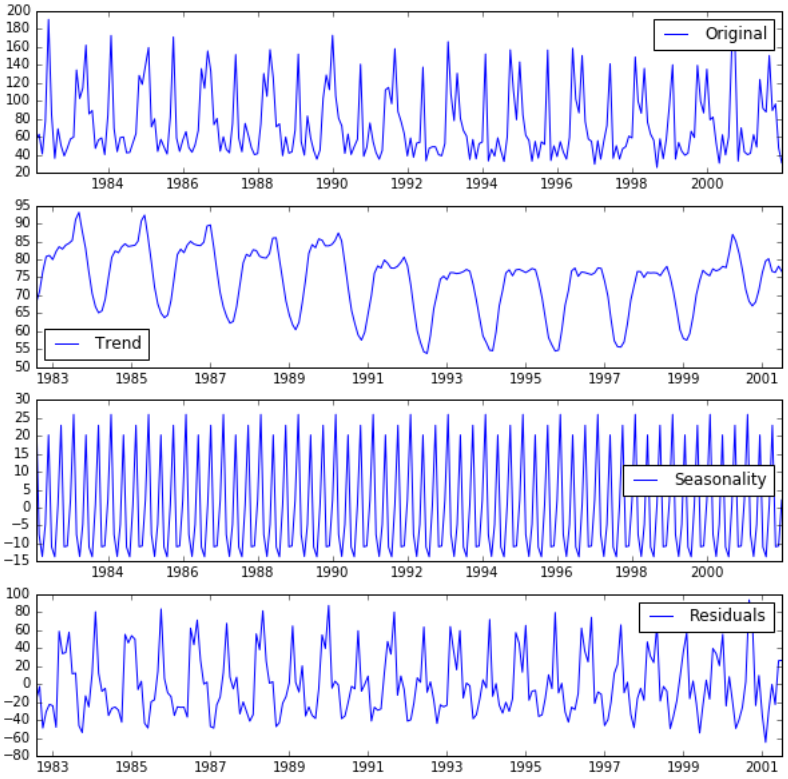
```
dfcoffee=pd.read_excel('R_data/coffeeprices.xls')

df=dfcoffee.iloc[31:51,1:14].unstack()
df=df.astype(float)

dates = pd.date_range('1982−01', '2002−01', freq='M')

df.index=dates
```

## 4.1 Decomposition

We decompose our time series:

```
decomposition = seasonal_decompose(df)

trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid
```

And we obtain:




As described in the seasonal_decompose description, here, in fact we are using the additive model :

$$Y_t = T_t + S_t + \epsilon_t \tag{6}$$

So what is called here "*trend*", $T_t$ is in fact the time series without its seasonal component.
One should not mix this with the actual trend and cycle, so in an additive model, we defined in equation 1:

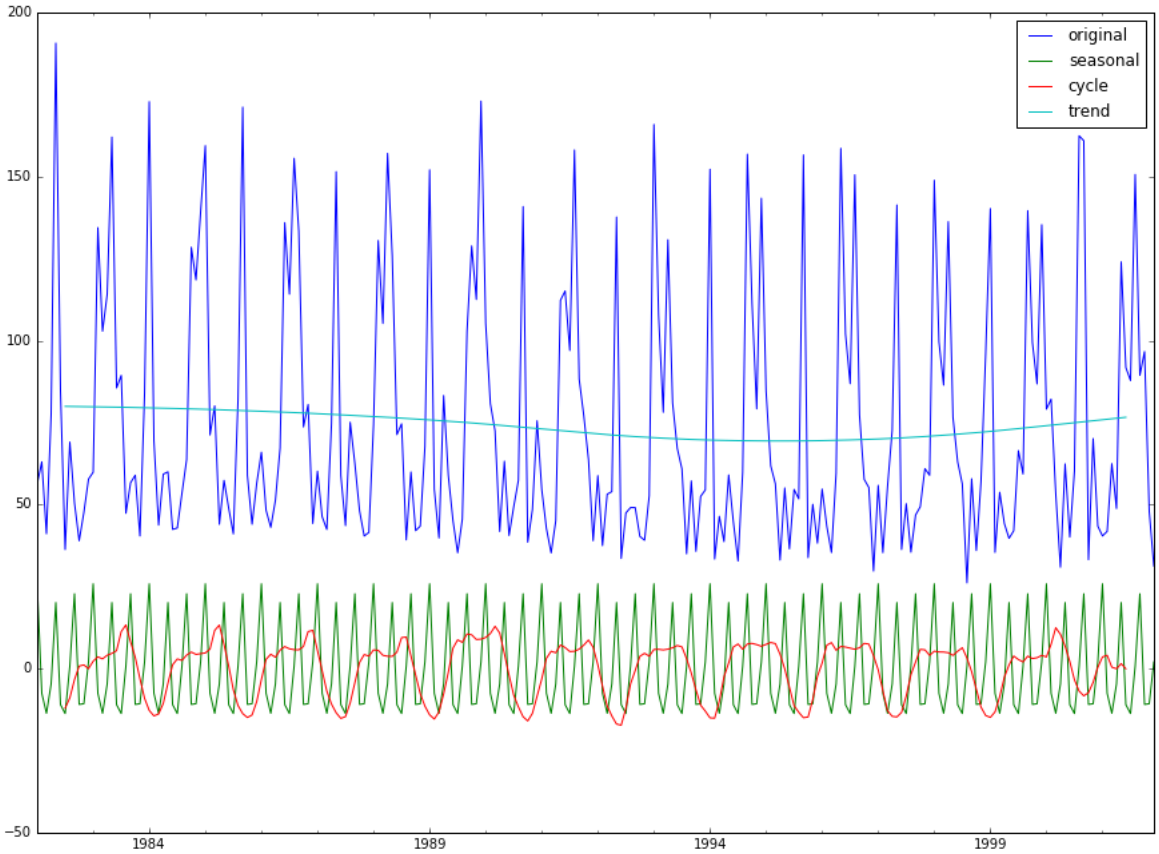
$$Y_t = L_t + S_t + C_t + I_t \tag{7}$$

so $T_t = L_t + C_t$
In order to separate the long-term secular trend from the long-term cyclical component, we can apply a Hodrick-Prescott filter on $T_t$, for monthly data we use $\lambda = 129600$:

**import** statsmodels.api as sm
C,L = sm.tsa.filters.hpfilter(trend.dropna(axis=0),129600)

dfcomposantes=pd.concat([df,seasonal,C,L],axis=1)
dfcomposantes.columns=['original','seasonal','cycle','trend']

We can then plot the different components:




## 5 X-13-ARIMA

Python Statsmodel package should allow the use of the United Sates Census Bureau seasonal adjustment program, X-13ARIMA-SEATS. Details here about the python package.