Assignment 8 - Time series

June 29, 2019

0.1 Assignment 8

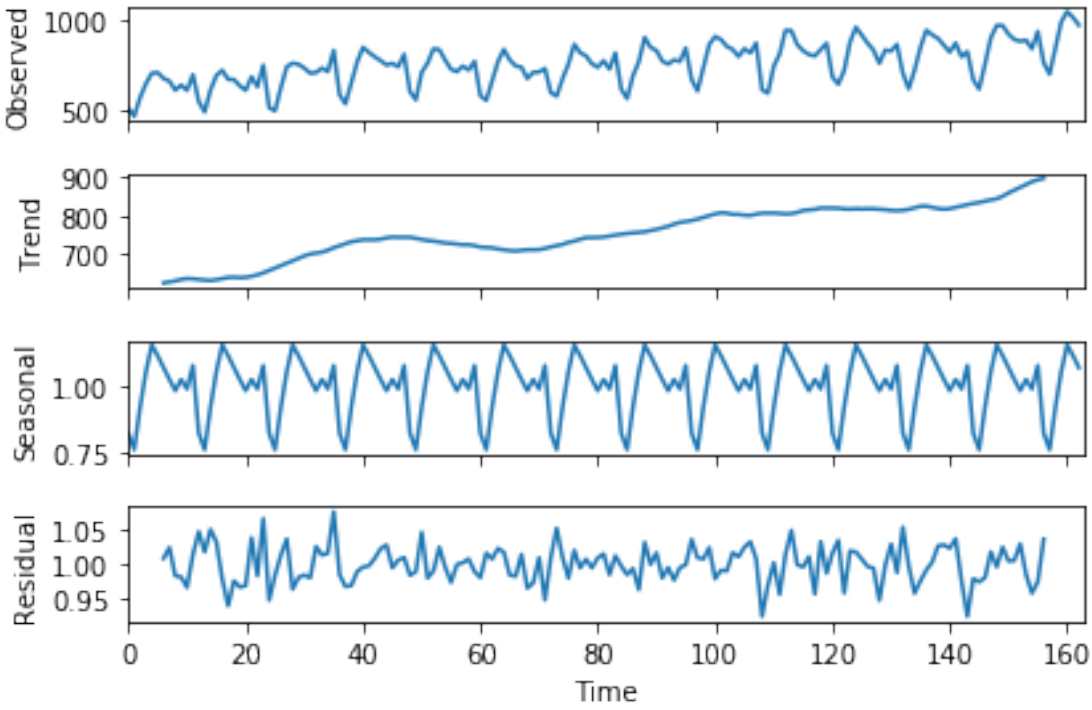
0.1.1 Problem

Using a suitable Python library, extrapolate the time series of the Paladini example over the next 6 months.

0.1.2 Resolution

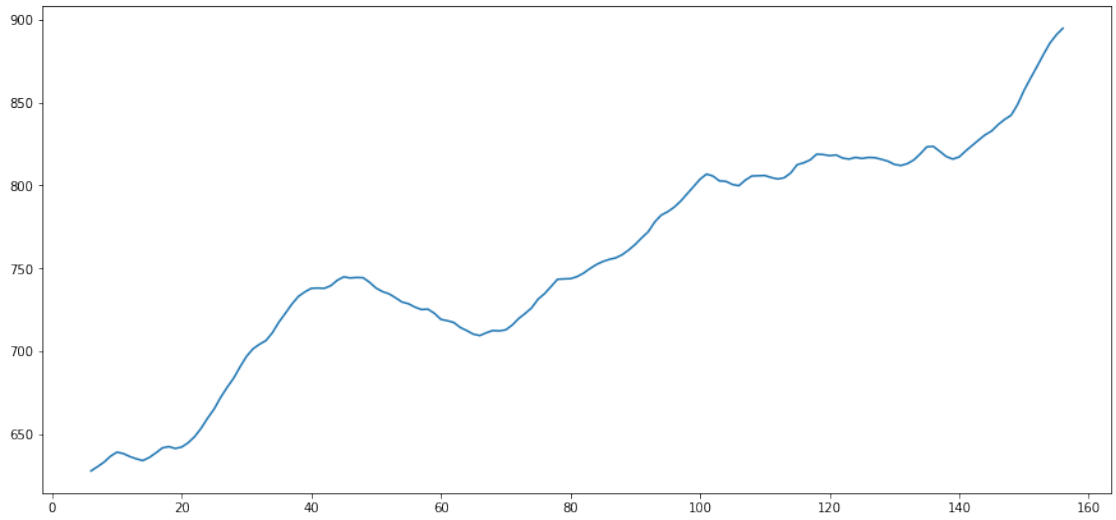
We can extrapolate the decomposition through the python library *statsmodels*. In particoular we can use the function *seasonal_decompose()* in order to extract and print the 4 variables.

```
In [1]: from random import randrange
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from statsmodels.tsa.seasonal import seasonal_decompose
        %matplotlib inline

In [2]: file = "Paladini  Associates example.csv"

        data = open(file, "r")
        data = data.read()
        data = data.split(",")
        data = [float(d) for d in data]
        data = np.array(data)

In [3]: m = data.shape[0]

In [4]: pal = seasonal_decompose(data, model='multiplicative', freq=12)
        pal.plot()
        plt.show()
```




```
In [5]: M = 12
        T = data.shape[0]
        qvt = np.zeros(data.shape[0])
        for i in range(int(M/2),T-int(M/2)):
            tmp = 0
            tmp = data[i-6]/2
            tmp += data[i+6]/2
            for j in range(i-5,i+6):
                tmp += data[j]
            qvt[i] = tmp/M
```

```
In [6]: x = np.arange(qvt.shape[0])
        plt.figure(figsize=(15,7))
        plt.plot(x[int(M/2):T-int(M/2)], qvt[int(M/2):T-int(M/2)])
        plt.show()
```




```
In [7]: z = np.polyfit(x[int(M/2):T-int(M/2)], qvt[int(M/2):T-int(M/2)], 1)

In [8]: q = np.zeros(data.shape[0])
        q[int(M/2):T-int(M/2)] = z[0] * x[int(M/2):T-int(M/2)] + z[1]

In [9]: v = np.zeros(data.shape[0])
        v[int(M/2):T-int(M/2)] = qvt[int(M/2):T-int(M/2)] / q[int(M/2):T-int(M/2)]

In [10]: srt = np.zeros(data.shape[0])
         srt[int(M/2):T-int(M/2)] = data[int(M/2):T-int(M/2)]/qvt[int(M/2):T-int(M/2)]

In [11]: s = np.zeros(12)
         for i in range(12):
             count = 0
             for j in range(i+12,T-int(M/2)-1,12):
                 s[i] += srt[j]
                 count += 1
             s[i] = s[i]/count

In [12]: xx = v[149:157]
         yy=np.arange(xx.shape[0])
         zz = np.polyfit(yy, xx, 2)

In [13]: t = np.arange(xx.shape[0],xx.shape[0]+7)
         a = zz[0] * (t**2) + zz[1]*t + zz[2]

In [14]: t = np.arange(157,163)
         qt = z[0]*t+z[1]

In [15]: p = a[:6] * qt * s[1:7]
```

In [16]: print(pal.trend)

```
[         nan          nan          nan          nan          nan
          nan 627.69583333 630.23333333 632.94583333 636.50416667
 639.0125     638.1375     636.32916667 634.9875     633.90833333
 635.80416667 638.51666667 641.55       642.36666667 641.22916667
 642.075      644.7        648.375      653.45       659.51666667
 665.02916667 672.         678.06666667 683.55       690.43333333
 696.79166667 701.37083333 704.1125     706.35833333 711.1125
 717.5        722.8375     728.35       732.95833333 735.72916667
 737.8875     738.0625     737.85833333 739.4625     742.72916667
 744.8        744.15833333 744.45       744.30416667 741.53333333
 738.03333333 735.90416667 734.5625     732.2        729.63333333
 728.58333333 726.57083333 725.08333333 725.34583333 722.8375
 719.1625     718.34583333 717.26666667 714.2625     712.36666667
 710.26666667 709.3625     711.05416667 712.425      712.22083333
 712.80416667 715.6625     719.62916667 722.72083333 726.075
 731.325      734.67916667 738.9375     743.34166667 743.60416667
 743.72083333 744.91666667 747.01666667 749.7875     752.2375
 754.075      755.3875     756.29166667 758.21666667 761.075
 764.4        768.36666667 772.04166667 777.90416667 782.075
 784.11666667 786.85833333 790.475      794.90833333 799.225
 803.74583333 806.86666667 805.64166667 802.725      802.4625
 800.59583333 799.8375     803.25       805.7        805.81666667
 805.99166667 804.76666667 803.89166667 804.65       807.5375
 812.55416667 813.6625     815.4125     818.88333333 818.65
 817.97916667 818.41666667 816.52083333 815.90833333 816.92916667
 816.2875     816.9        816.725      815.7625     814.625
 812.7        812.02916667 813.16666667 815.4125     819.11666667
 823.2875     823.57916667 820.60416667 817.45416667 815.90833333
 817.22083333 820.8375     824.1625     827.4        830.55
 832.85416667 836.675      839.825      842.30416667 848.77916667
 857.325      864.64583333 871.73333333 879.05416667 885.90833333
 890.95416667 894.8625              nan          nan          nan
          nan          nan          nan]
```

In [17]: print(pal.seasonal)

```
[0.82382161 0.76156155 0.92124624 1.05494351 1.15456277 1.11418445
 1.06953315 1.02505042 0.98307259 1.02505984 0.98940811 1.07755577
 0.82382161 0.76156155 0.92124624 1.05494351 1.15456277 1.11418445
 1.06953315 1.02505042 0.98307259 1.02505984 0.98940811 1.07755577
 0.82382161 0.76156155 0.92124624 1.05494351 1.15456277 1.11418445
 1.06953315 1.02505042 0.98307259 1.02505984 0.98940811 1.07755577
 0.82382161 0.76156155 0.92124624 1.05494351 1.15456277 1.11418445
 1.06953315 1.02505042 0.98307259 1.02505984 0.98940811 1.07755577
 0.82382161 0.76156155 0.92124624 1.05494351 1.15456277 1.11418445
```

```
 1.06953315 1.02505042 0.98307259 1.02505984 0.98940811 1.07755577
 0.82382161 0.76156155 0.92124624 1.05494351 1.15456277 1.11418445
 1.06953315 1.02505042 0.98307259 1.02505984 0.98940811 1.07755577
 0.82382161 0.76156155 0.92124624 1.05494351 1.15456277 1.11418445
 1.06953315 1.02505042 0.98307259 1.02505984 0.98940811 1.07755577
 0.82382161 0.76156155 0.92124624 1.05494351 1.15456277 1.11418445
 1.06953315 1.02505042 0.98307259 1.02505984 0.98940811 1.07755577
 0.82382161 0.76156155 0.92124624 1.05494351 1.15456277 1.11418445
 1.06953315 1.02505042 0.98307259 1.02505984 0.98940811 1.07755577
 0.82382161 0.76156155 0.92124624 1.05494351 1.15456277 1.11418445
 1.06953315 1.02505042 0.98307259 1.02505984 0.98940811 1.07755577
 0.82382161 0.76156155 0.92124624 1.05494351 1.15456277 1.11418445
 1.06953315 1.02505042 0.98307259 1.02505984 0.98940811 1.07755577
 0.82382161 0.76156155 0.92124624 1.05494351 1.15456277 1.11418445
 1.06953315 1.02505042 0.98307259 1.02505984 0.98940811 1.07755577
 0.82382161 0.76156155 0.92124624 1.05494351 1.15456277 1.11418445
 1.06953315 1.02505042 0.98307259 1.02505984 0.98940811 1.07755577
 0.82382161 0.76156155 0.92124624 1.05494351 1.15456277 1.11418445
 1.06953315]
```

```
In [18]: print(pal.resid)
```

```
[       nan        nan        nan        nan        nan        nan
 1.00827969 1.02396061 0.98323503 0.98167711 0.96655699 1.01391922
 1.04688528 1.01761425 1.05002576 1.03214748 0.97896444 0.94011649
 0.976082   0.96699759 0.96925517 1.0380475  0.98315299 1.06571409
 0.94823701 0.98408436 1.01312037 1.03631647 0.96413921 0.98092987
 0.98437851 0.98047061 1.02543552 1.01317632 1.0158042  1.07560501
 0.98507252 0.96793962 0.96929201 0.98936716 0.99584777 0.9976436
 1.00853568 1.02231393 1.02772462 0.99480713 1.00587241 1.00961573
 0.98406022 0.98915703 1.04602013 0.98011519 0.99045097 1.02450846
 0.99568532 0.9738434  0.99961938 1.0020803  1.00757591 0.98947523
 0.98065426 1.01596783 1.00744693 1.02281787 1.01705586 0.98449764
 0.98261903 1.01417767 0.96549589 0.97319572 1.00942324 0.94856329
 1.00953695 1.0517874  1.00987621 0.97990093 1.0200025  0.99476181
 1.00725942 0.99457939 1.00912071 1.01481937 0.98497594 1.01195903
 0.99740281 0.98489481 0.99382396 0.96334503 1.03151815 0.99967225
 1.01632786 0.98030182 0.99516057 0.97705247 0.9951003  0.99913669
 1.03558779 1.00931009 1.00750189 1.02367678 0.97912118 0.99121434
 0.99029726 1.01660895 1.01067479 1.02442107 1.03226611 1.00768685
 0.92489215 0.96956228 1.00213222 0.95643805 1.01439067 1.04860022
 0.99931906 0.99590713 1.01076487 0.95723305 1.03244862 0.98714374
 1.0159246  1.03437543 0.95850033 1.01901022 1.01749795 1.00748236
 0.99667893 0.9941658  0.94793496 0.99839604 1.02898677 0.98799242
 1.05328482 0.99760573 0.95824568 0.97602616 0.99161419 1.00294825
 1.0264278  1.02780283 1.02291871 1.03659635 0.97604579 0.92488568
 0.97906386 0.97450657 0.98081991 1.01764406 0.99620251 1.02443009
```

```
 1.00464862 1.00540945 1.02919898 0.98503676 0.95832989 0.97484221
 1.0359369         nan         nan         nan         nan         nan
        nan]
```

```
In [19]: print(pal.observed)
```

```
[ 511.7  468.3  571.9  648.2  705.6  709.1  676.9  661.5  611.8  640.5
  611.1  697.2  548.8  492.1  613.2  692.3  721.7  672.   670.6  635.6
  611.8  686.   630.7  750.4  515.2  498.4  627.2  741.3  760.9  754.6
  733.6  704.9  709.8  733.6  714.7  831.6  586.6  536.9  654.5  767.9
  848.4  820.4  795.9  774.9  750.4  759.5  740.6  809.9  603.4  558.6
  711.2  760.9  840.   835.8  777.   727.3  714.   744.8  723.1  770.7
  581.   555.8  665.7  770.7  836.5  779.1  745.5  739.2  676.2  710.5
  711.9  731.5  598.5  578.9  675.5  756.   865.2  819.   800.8  758.1
  737.8  774.9  728.   817.6  618.1  565.6  691.6  768.6  903.   847.7
  830.9  772.1  755.3  779.1  770.   844.2  671.3  607.6  737.8  863.1
  908.6  891.1  853.3  836.5  797.3  840.7  816.9  872.2  613.9  595.
  744.1  812.   941.5  940.1  863.1  829.5  808.5  800.1  836.5  870.8
  684.6  644.7  721.   877.1  959.7  916.3  870.8  832.3  760.2  833.7
  827.4  864.5  705.6  619.5  723.1  847.7  942.9  917.   897.4  859.6
  821.8  872.2  795.9  824.6  669.9  618.1  756.   901.6  968.8  968.8
  921.2  891.1  882.   887.6  840.   935.9  763.7  700.   844.2  989.1
 1045.8 1012.9  970.9]
```