Time Series Practice

May 12, 2023

In this Juptyer notebook I will be performing some techniques that are useful for discerning patterns in a time series.

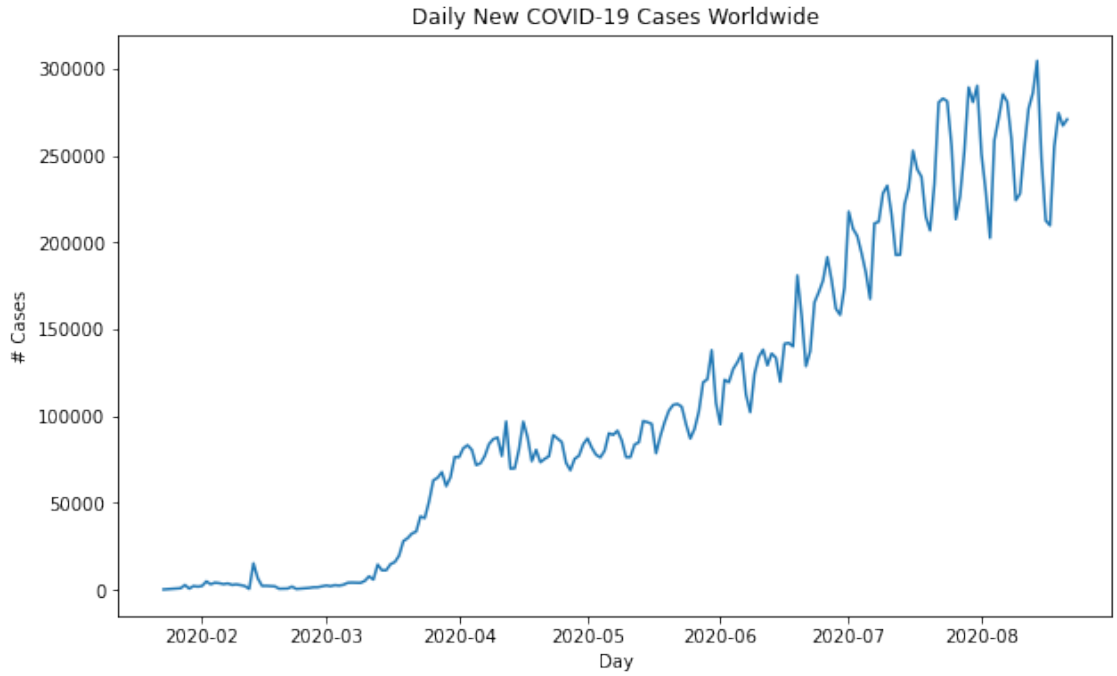
[1]:
```
#Importing necessary tools
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()

# Suppress all warnings
import warnings
warnings.filterwarnings("ignore")

import datetime as dt
from datetime import date
from datetime import datetime
from statsmodels.tsa.seasonal import seasonal_decompose, DecomposeResult
import statsmodels.api as sm
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
```

Loading some data for this I will be using Aug 2020 Covid-19 data. The number of daily new cases worldwide is a time series that arises naturally from this topical event, I will apply some of the techniques learned in class to this very time series to discern any patterns it may contain.

[2]:
```
#This function should returns a pd.Series of length 212, whose index is a pd.
 ↪DatetimeIndex

def load_data():
    df = pd.read_csv("assets/time_series_covid19_confirmed_global.csv")
    column1 = list(df.iloc[0,4:216].index)
    column2 = list(df.iloc[0,5:].index)
    new_cases =[]
    for x in range (0,212):
        y = column1[x]
```

```
        x = column2[x]
        new_cases.append(float(-(df[y].sum() - df[x].sum())))

    daily_new_cases = pd.Series(new_cases,index = column2)
    daily_new_cases.index = pd.to_datetime(daily_new_cases.index)
    return daily_new_cases
```

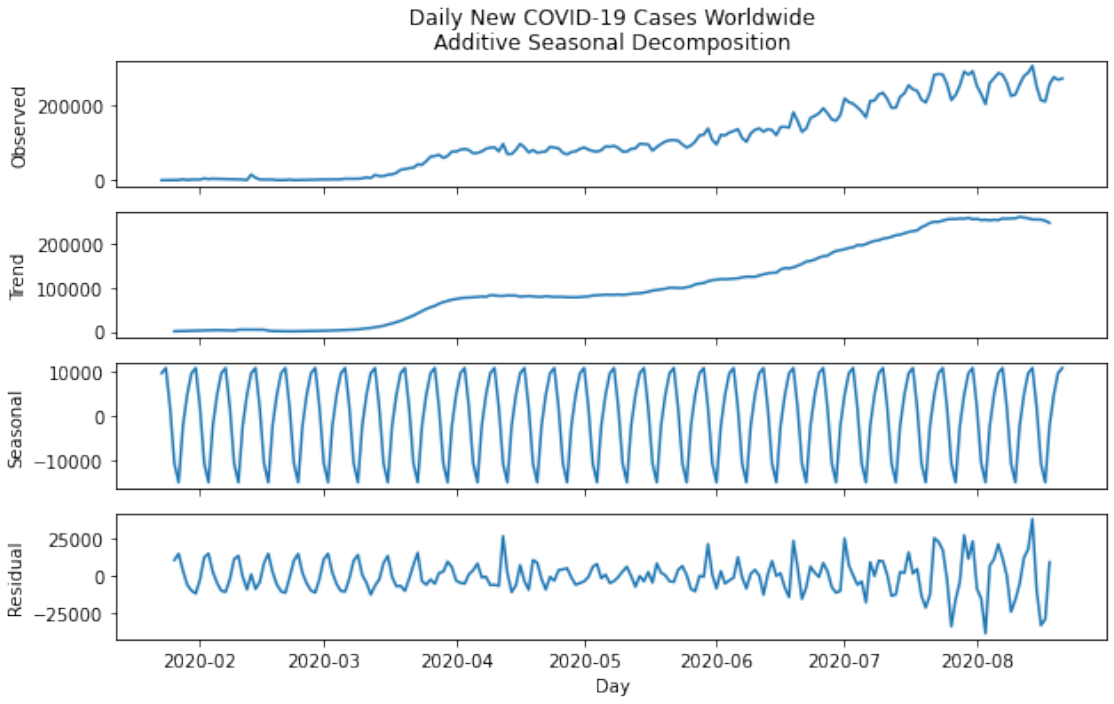
[3]:
```
# let's see the time series
fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(load_data())
ax.set_xlabel("Day")
ax.set_ylabel("# Cases")
ax.set_title("Daily New COVID-19 Cases Worldwide")

del fig, ax
```



[4]:
```
#With the times series prepared I will use seasonal decomposition to get a␣
 ↪sense of the possible patterns hidden in the data
from statsmodels.tsa.seasonal import seasonal_decompose, DecomposeResult

def sea_decomp(ser, model="additive"):
    """
    Takes in a series and a "model" parameter indicating which seasonal decomp␣
 ↪to perform
    """
```

```
    result = seasonal_decompose(ser,model=model)



    return result
```

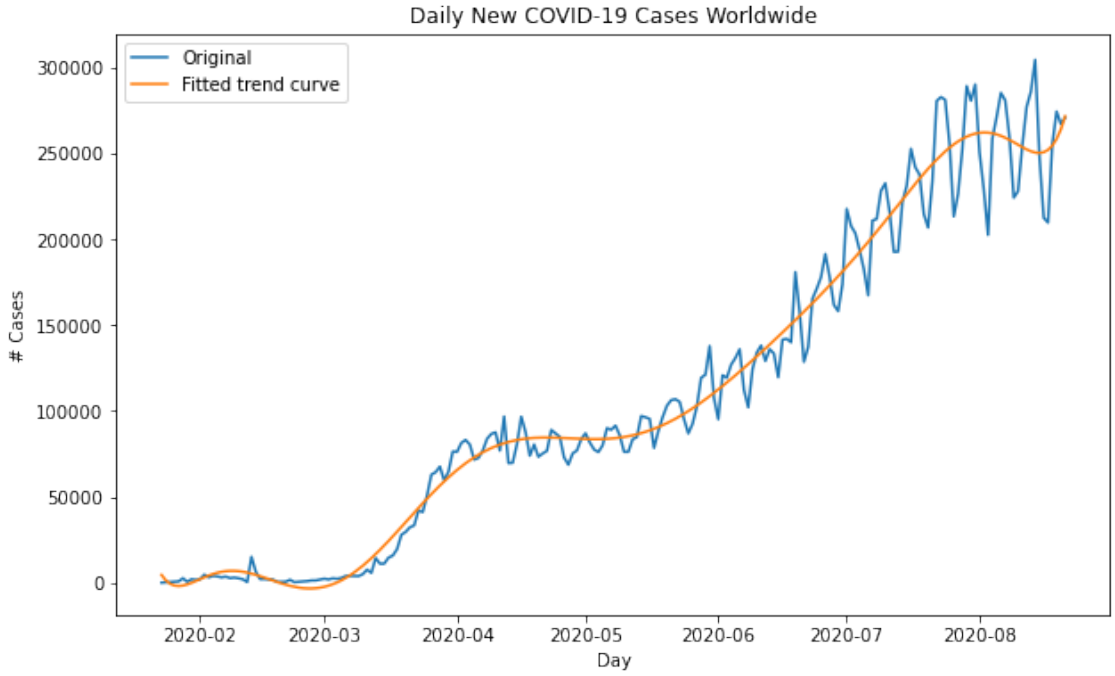
[5]:
```
#Let us see the seasonal decomposition
fig, axes = plt.subplots(4, 1, figsize=(10, 6), sharex=True)
res = sea_decomp(load_data(), model="additive")

axes[0].set_title("Additive Seasonal Decomposition")
axes[0].plot(res.observed)
axes[0].set_ylabel("Observed")

axes[1].plot(res.trend)
axes[1].set_ylabel("Trend")

axes[2].plot(res.seasonal)
axes[2].set_ylabel("Seasonal")

axes[3].plot(res.resid)
axes[3].set_ylabel("Residual")

axes[3].set_xlabel("Day")
fig.suptitle("Daily New COVID-19 Cases Worldwide", x=0.513, y=0.95)

del fig, axes, res
```



So, there is a non-linear trend hidden in this time series. Let's see if these trends can fit a regression model from there I will use the regression model to make predictions at each timestamp

[6]:
```
def fit_trend(ser, n):
    """
    Takes a series and fits an n-th order polynomial to the series.
    Returns the predictions.
    """

    ser_indx = ser.index.to_series()
    X_train = (ser_indx - ser_indx.iloc[0]).apply(lambda x: x.days).to_numpy()/
↪100
    y_train = ser.to_numpy()
    poly = PolynomialFeatures(n)
    X_train = poly.fit_transform(X_train.reshape(-1,1))
    L_reg = LinearRegression().fit(X_train,y_train.reshape(-1))
    trend_curve = L_reg.predict(X_train)


    return trend_curve
```

[7]:
```
#Lets see the regression line

fig, ax = plt.subplots(figsize=(10, 6))
ser = load_data()
preds = fit_trend(ser, 10)
ax.plot(ser.index, ser.values, label="Original")
ax.plot(ser.index, preds, label="Fitted trend curve")
ax.set_xlabel("Day")
ax.set_ylabel("# Cases")
ax.set_title("Daily New COVID-19 Cases Worldwide")
ax.legend()

del fig, ax, ser, preds
```



Next, I will use a weighted moving average (wma) as an alternate way of smoothing. If your interested, schedule me for an INTERVIEW. I will demonstrate how to assign greater weight to most recent events a crucial concept for time analysis.

Have a Great Day!

[ ]: