Time Series :

Project: Analyze The Data:
Covid 19
Morocco

Made by:

Boujbair Oussamae

DATA-INE2

Supervised by:

Pr.BADAOUI FADOUA

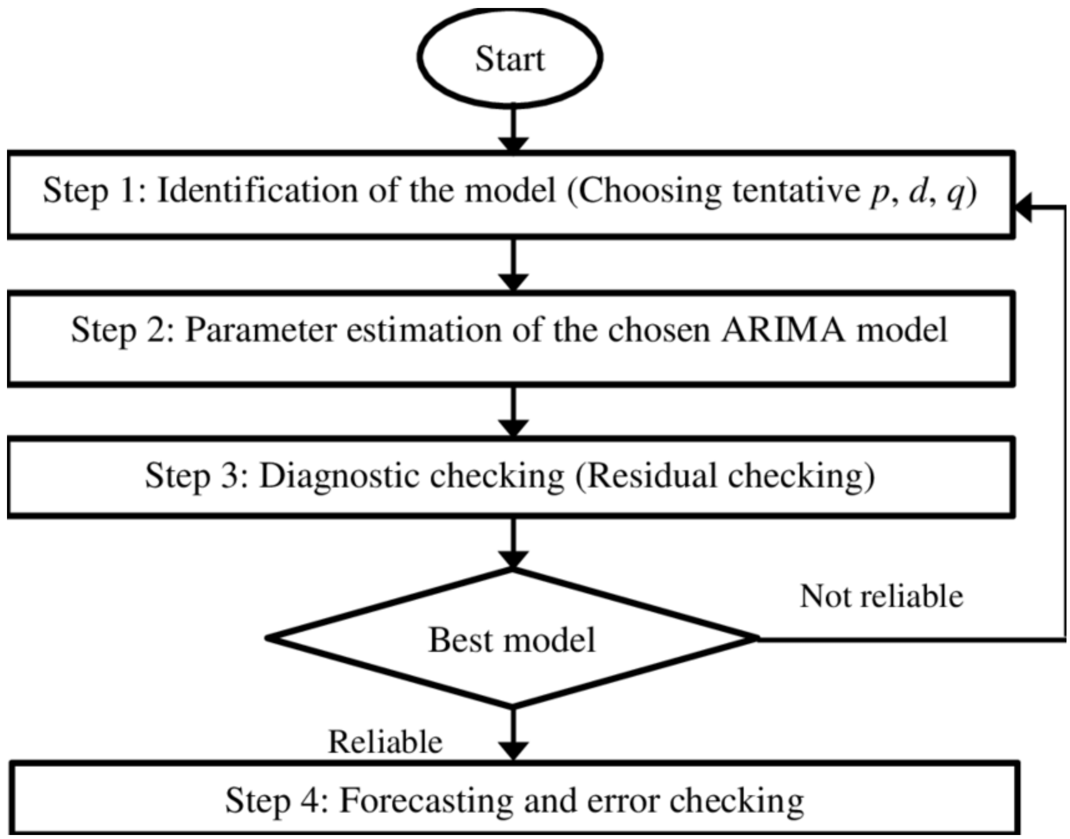
INSTITUT NATIONAL DES POSTES ET TELECOMMUNICATION

16 Junaury 2022
Rabat, Morocco

Part I

Abstract :

COVID-19 declared as a global pandemic by WHO, has emerged as the most aggressive disease, impacting more than 90% countries of the world. The virus started from a single human being in China, is now increasing globally at a rate of 3% to 5% daily and has become a never ending process. Some studies even predict that the virus will stay with us forever. Morocco is also not saved, and the virus is spreading as a community level transmitter. Therefore, it become really important to analyse the possible impact of COVID-19 in Morocco and forecast how it will behave in the days to come. In present work, prediction models based on Jenkins Box method, ARIMA to predict the time series, python libraries to analyze the data: **Pandas, Numpy, Seaborn, Matplotlib...**



Part II

Introduction :

The novel coronavirus (COVID-19) was widely reported to have first been detected in Wuhan (Hebei province, China) in December 2019. After the initial outbreak, COVID-19 continued to spread to all provinces in China and very quickly spread to other countries within and outside of Asia. At present, over 45 million cases of infected individuals have been confirmed in over 180 countries with in excess of 1 million deaths. Although the foundations of this disease are very similar to the severe acute respiratory syndrome (SARS) virus that took hold of Asia in 2003, it is shown to spread much more easily.

Although there are some similarities in epidemiology and clinical features between COVID-19, SARS-CoV, MERS-CoV and pandemic influenza viruses. The zoonotic origin of COVID-19 is not confirmed by researchers. Historically, the Middle East respiratory syndrome coronavirus (MERS-CoV) infection has been approved for transmission from dromedary camels to humans, and bats are the group of mammals that harbor the largest number Coronaviruses. That's why for COVID-19, the Human-Animal interaction has been questioned by researchers as a likely risk factorfor COVID19.

Morocco has also been exposed to the spread of the virus, given its proximity to Europe where the virus is already widespread. Morocco knows its first case of Coronavirus on March 02, 2020 and it registered until the date of January 13, 2022, 1025898 of contamination with COVID-19, including 961462 people healed and 14945 deaths.

1 What is a Time Series?

Time series is a sequence of observations recorded at regular time intervals. Depending on the frequency of observations, a time series may typically be hourly, daily, weekly, monthly, quarterly and annual. Sometimes, you might have seconds and minute-wise time series as well, like, number of clicks and user visits every minute etc. Why even analyze a time series? Because it is the preparatory step before you develop a forecast of the series. Besides, time series forecasting has enormous commercial significance because stuff that is important to a business like demand and sales, number of visitors to a website, stock price etc are essentially time series data. So what does analyzing a time series involve? Time series analysis involves understanding various aspects about the inherent nature of the series so that you are better informed to create meaningful and accurate forecasts.

Part III

Project :

2 Libraries used :

Pandas, Numpy, Seaborn, Matplotlib,statsmodels.tsa.seasonal, dateutil.parser ...

```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from dateutil.parser import parse
from statsmodels.tsa.stattools import acf, pacf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import math
import statsmodels.api as sm
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.stats.diagnostic import acorr_ljungbox as ljungbox
import matplotlib.gridspec as gridspec
from scipy.stats import chi2
from pandas.plotting import lag_plot


import os
print(os.listdir(r"C:\Users\Oussama\Desktop\python"))
```

```
['.ipynb_checkpoints', 'athlete_events.csv', 'c.csv', 'covid-19-datasets-1200x900.jpg', 'Covid_19_Dataset_Morocco.csv', 'Data_
F.csv', 'data_MAR_2.csv', 'df10.csv', 'df9.csv', 'mathm9nich.ipynb', 'new3.csv', 'noc_regions.csv', 'Olympics.ipynb', 'Olympics
_Boujbair.ipynb', 'our_data.csv', 'owid-covid-data.csv', 'projet_SC.ipynb', 'sere_temp1.ipynb', 'serie_temp - Copie (2).ipynb',
'serie_temp - Copie.ipynb', 'serie_temp.ipynb', 'Untitled.ipynb', 'Untitled1.ipynb', 'Untitled2.ipynb', 'Untitled3.ipynb', 'Unt
itled4.ipynb', 'Untitled5.ipynb', 'VosQuestions.ipynb']
```

3 Dataset :

3.1 Original Data

Covid-19 data of all countries of the world.
This data contains 67 columns: iso-code, continent, location,date, total-cases, new-cases.... and 153,630 rows.

```
data_covid =pd.read_csv('owid-covid-data.csv')
data_covid
```

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	...	female_sm
0	AFG	Asia	Afghanistan	2020-02-24	5.0	5.0	NaN	NaN	NaN	NaN	...	
1	AFG	Asia	Afghanistan	2020-02-25	5.0	0.0	NaN	NaN	NaN	NaN	...	
2	AFG	Asia	Afghanistan	2020-02-26	5.0	0.0	NaN	NaN	NaN	NaN	...	
3	AFG	Asia	Afghanistan	2020-02-27	5.0	0.0	NaN	NaN	NaN	NaN	...	
4	AFG	Asia	Afghanistan	2020-02-28	5.0	0.0	NaN	NaN	NaN	NaN	...	
...	...	...	...	...	...	...	...	...	...	...	...	
153625	ZWE	Africa	Zimbabwe	2022-01-06	220178.0	1121.0	1207.143	5108.0	16.0	15.857	...	
153626	ZWE	Africa	Zimbabwe	2022-01-07	221282.0	1104.0	1146.286	5136.0	28.0	18.857	...	
153627	ZWE	Africa	Zimbabwe	2022-01-08	221918.0	636.0	1100.571	5148.0	12.0	18.714	...	
153628	ZWE	Africa	Zimbabwe	2022-01-09	221918.0	0.0	1100.571	5148.0	0.0	18.714	...	
153629	ZWE	Africa	Zimbabwe	2022-01-10	223000.0	1082.0	987.571	5180.0	32.0	19.000	...	

153630 rows × 67 columns

3.2 Organize the Data for analysis:

Extract data specific to Morocco with specific columns

```
data_MAR=data_covid[data_covid['iso_code']=='MAR'].filter(['location','date','new_cases','total_cases','new_deaths','total_d
data_MAR.sample(5)
```

	location	date	new_cases	total_cases	new_deaths	total_deaths
95373	Morocco	2020-11-20	4706.0	316260.0	92.0	5182.0
95322	Morocco	2020-09-30	2470.0	123653.0	42.0	2194.0
95490	Morocco	2021-03-17	466.0	490088.0	8.0	8745.0
95243	Morocco	2020-07-13	191.0	15936.0	5.0	255.0
95305	Morocco	2020-09-13	2251.0	86686.0	25.0	1578.0

Reset the index of Data

```
def reset_my_index(df):
  res = df.reset_index(drop=True) # function to reverse order of row and resets index
  return(res)
reset_my_index(data_MAR)
```

	location	date	new_cases	total_cases	new_deaths	total_deaths
0	Morocco	2020-02-07	NaN	NaN	NaN	NaN
1	Morocco	2020-02-08	NaN	NaN	NaN	NaN
2	Morocco	2020-02-09	NaN	NaN	NaN	NaN
3	Morocco	2020-02-10	NaN	NaN	NaN	NaN
4	Morocco	2020-02-11	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...
699	Morocco	2022-01-06	6050.0	983629.0	11.0	14883.0
700	Morocco	2022-01-07	6428.0	990057.0	13.0	14896.0
701	Morocco	2022-01-08	7064.0	997121.0	8.0	14904.0
702	Morocco	2022-01-09	4963.0	1002084.0	7.0	14911.0
703	Morocco	2022-01-10	2622.0	1004706.0	4.0	14915.0

704 rows × 6 columns

Extract data with two columns: date and new-cases

```
data_MAR_2 = data_MAR.loc[:,['date','new_cases']]
data_MAR.dropna(subset = ["new_cases"], inplace=True) # drop empty rows
data_MAR_2.to_csv(r'C:\Users\Oussama\Desktop\python\data_MAR_2.csv', index = False) # save data_MAR_2 in a csv file
Data_F = pd.read_csv('data_MAR_2.csv',index_col='date') # take column date as an index of data
Data_F.to_csv(r'C:\Users\Oussama\Desktop\python\Data_F.csv', index = False) # # save Data_F in a csv file
Data_F.index = pd.to_datetime(Data_F.index)
Data_F.head()
```

	new_cases
date	
2020-03-02	1.0
2020-03-03	0.0
2020-03-04	0.0
2020-03-05	1.0
2020-03-06	0.0

4 Training data, Testing data :

we divide the data into two parts: Training data 80% and Testing data 20%.

```
train=Data_F.sample(frac=0.8,random_state=200)
train
```

	new_cases
date	
2021-09-25	1444.0
2020-12-30	2143.0
2021-07-17	2853.0
2021-05-24	90.0
2021-04-06	696.0
...	...
2021-07-22	1402.0
2021-06-30	776.0
2020-03-12	1.0
2020-05-25	99.0
2021-03-24	439.0

544 rows × 1 columns

```
test=data_F.drop(train.index)
test
```

	new_cases
date	
2020-03-02	1.0
2020-03-04	0.0
2020-03-08	0.0
2020-03-09	0.0
2020-03-19	14.0
...	...
2021-12-27	291.0
2022-01-04	4299.0
2022-01-05	5618.0
2022-01-08	7064.0
2022-01-10	2622.0

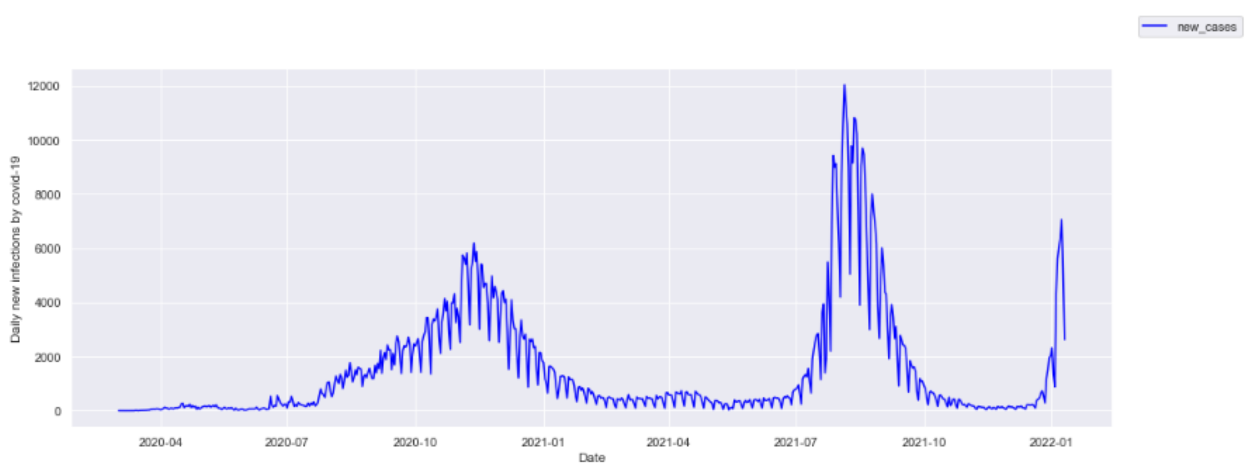
136 rows × 1 columns

5 Data visualization :

We analyze new cases affected by the covid over time from 03/02/2020 to 01/10/2022.
Let's use matplotlib to visualise the series.

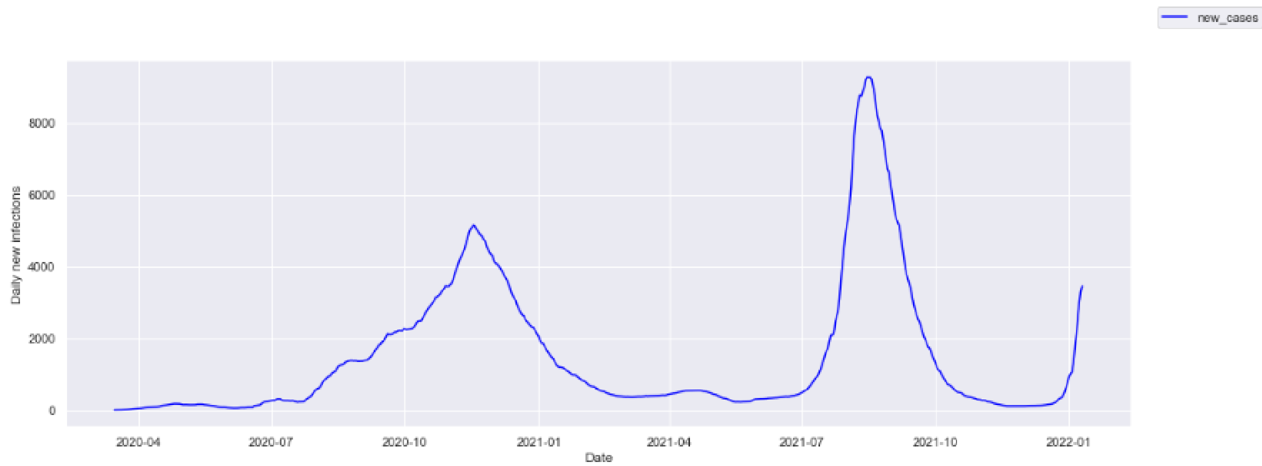
```
fig,ax = plt.subplots(figsize=(17,6))
rolling_avg = 1
ax.plot(Data_F.index,Data_F['new_cases'].rolling(window=rolling_avg).mean(),color='blue',label='new_cases')
ax.figure.legend()
sns.set()
sns.set_style("darkgrid")
ax.set_xlabel('Date')
ax.set_ylabel('Daily new infections by covid-19')
```

```
Text(0, 0.5, 'Daily new infections by covid-19')
```



```
fig,ax = plt.subplots(figsize=(17,6))
rolling_avg = 14
ax.plot(Data_F.index,Data_F['new_cases'].rolling(window=rolling_avg).mean(),color='blue',label='new_cases')
ax.figure.legend()
sns.set()
sns.set_style("darkgrid")
ax.set_xlabel('Date')
ax.set_ylabel('Daily new infections')
```

```
Text(0, 0.5, 'Daily new infections')
```



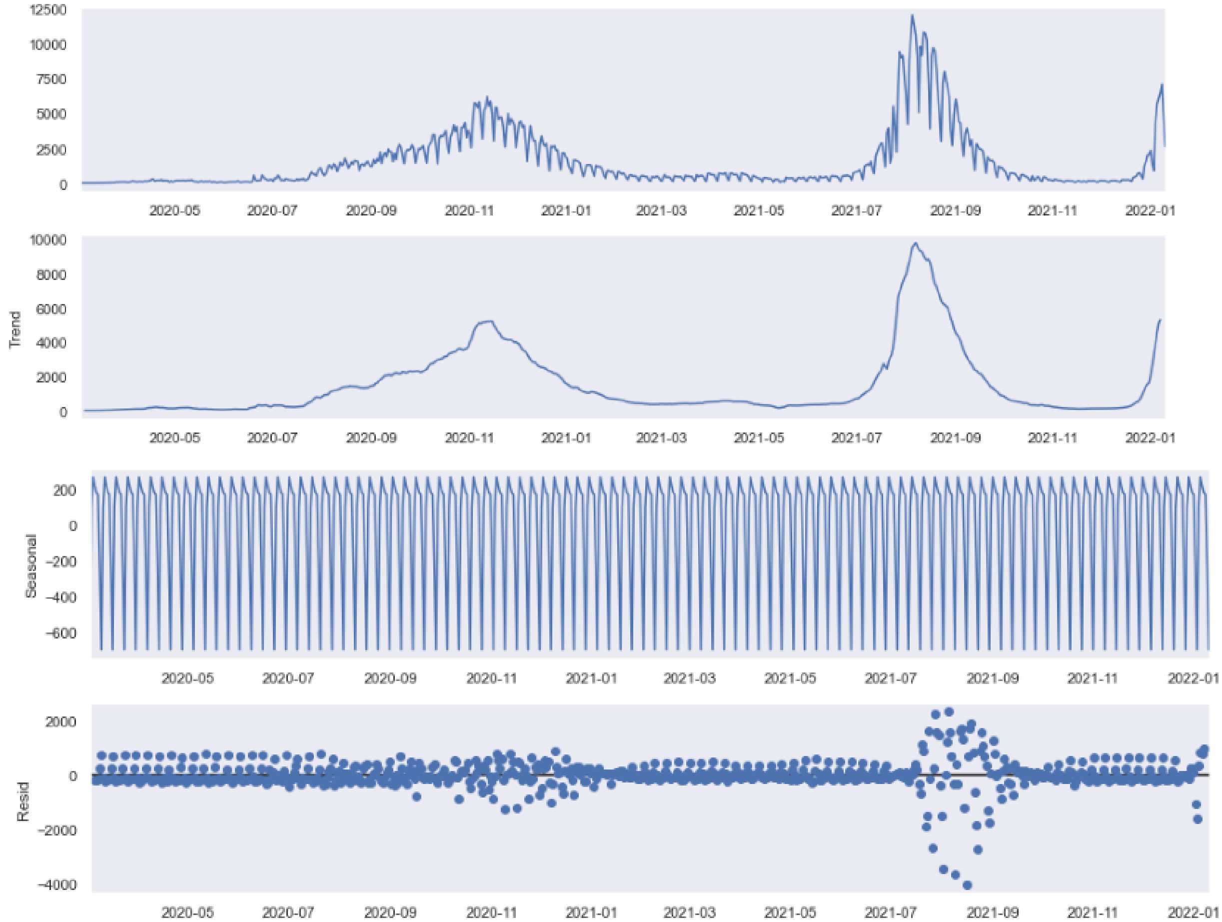
Since all values are positive, you can show this on both sides of the Y axis to emphasize the growth.

```
x = Data_F.index
y1 = Data_F['new_cases'].values

fig, ax = plt.subplots(1, 1, figsize=(19,6), dpi= 120)
plt.fill_between(x, y1=y1, y2=-y1, alpha=0.5, linewidth=2, color='seagreen')
plt.ylim(-12000, 12000)
plt.title('Daily new cases', fontsize=16)
plt.hlines(y=0, xmin=np.min(x), xmax=np.max(x), linewidth=.5)
plt.show()
```

6 Additive and multiplicative time series :

Depending on the nature of the trend and seasonality, a time series can be modeled as an additive or multiplicative, wherein, each observation in the series can be expressed as either a sum or a product of the components: Additive time series: Value = Base Level + Trend + Seasonality + Error Multiplicative Time Series: Value = Base Level x Trend x Seasonality x Error.

```python
dec=sm.tsa.seasonal_decompose(Data_F, model = 'additive')
fig = dec.plot()
plt.rcParams['figure.figsize'] = [10.0, 5.0]
```




```python
dec=sm.tsa.seasonal_decompose(Data_F, model = 'multiplicative')
fig = dec.plot()
plt.rcParams['figure.figsize'] = [9.0, 5.0]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-34-574bc4927d37> in <module>
----> 1 dec=sm.tsa.seasonal_decompose(Data_F, model = 'multiplicative')
      2 fig = dec.plot()
      3 plt.rcParams['figure.figsize'] = [9.0, 5.0]

E:\anaconda\lib\site-packages\pandas\util\_decorators.py in wrapper(*args, **kwargs)
    197                 else:
    198                     kwargs[new_arg_name] = new_arg_value
--> 199             return func(*args, **kwargs)
    200 
    201         return cast(F, wrapper)

E:\anaconda\lib\site-packages\statsmodels\tsa\seasonal.py in seasonal_decompose(x, model, filt, period, two_sided, extrapolate_
trend)
    133     if model.startswith('m'):
    134         if np.any(x <= 0):
--> 135             raise ValueError("Multiplicative seasonality is not appropriate "
    136                              "for zero and negative values")
    137 

ValueError: Multiplicative seasonality is not appropriate for zero and negative values
```

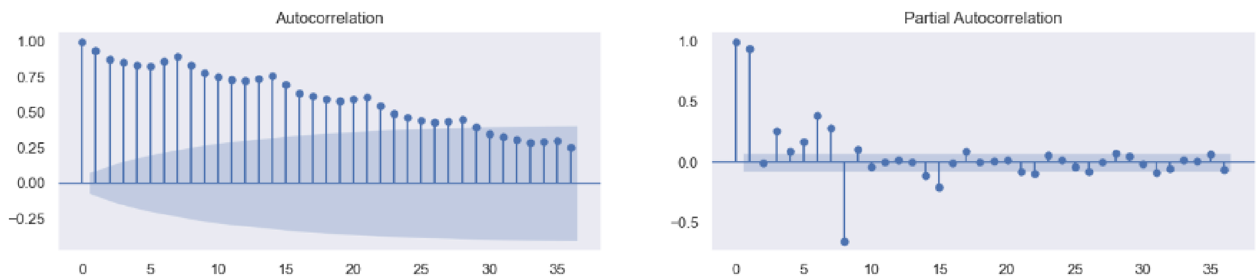
Additive models for seasonal-trend decomposition should have no problem with zero values.
the trend will be calculated adjust to the appropriate level, which can be near zero without any restriction.
We must work with additive model in our situation.

7 Simple and partial correlogram ACF, PACF :

Autocorrelation is simply the correlation of a series with its own lags. If a series is significantly autocorrelated, that means, the previous values of the series (lags) may be helpful in predicting the current value. Partial Autocorrelation also conveys similar information but it conveys the pure correlation of a series and its lag, excluding the correlation contributions from the intermediate lags.

Simple and partial correlogram for an order shift K≥36.

```
sns.set_style("dark")
fig, axes = plt.subplots(1,2,figsize=(16,3), dpi= 100)
plot_acf(df10.new_cases.tolist(), lags=36, ax=axes[0])
plot_pacf(df10.new_cases.tolist(), lags=36, ax=axes[1])
```



The time series is non-stationary as is visible from the autocorrelation plot and suggests an ARIMA model.

8 Box-Jenkins :

8.1 Definition :

The Box-Jenkins Model is a forecasting methodology using regression studies on time series data. The methodology is predicated on the assumption that past occurrences influence future ones.

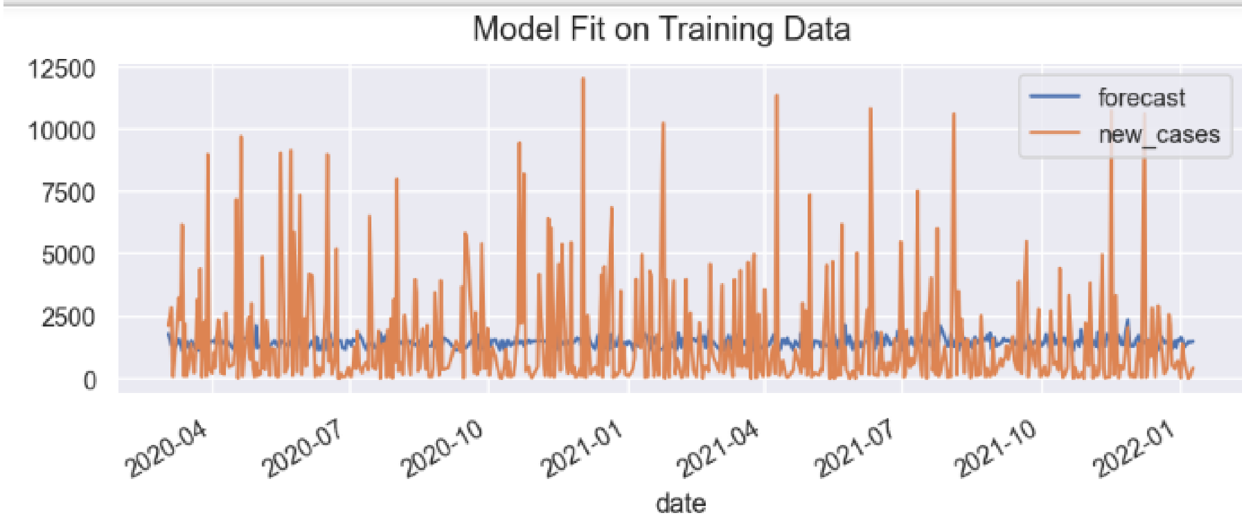
```python
def chi_square_table(p,dof):
    return chi2.isf(p,dof)

def chi_sq_critical_val(alpha,dof):
    pr=1-alpha
    val=chi2.ppf(pr,dof)
    return val

def eval_arima(series,order,lags,dynamic=False,alpha=0.05):

    plt.rcParams.update({'figure.figsize':(9,3), 'figure.dpi':120})

    #fit the model
    model=ARIMA(series,order=order)
    model_fit=model.fit(disp=-1)

    #print(type(model_fit))
    print(model_fit.summary())

    #display the fit of the model
    model_fit.plot_predict(dynamic=dynamic).suptitle("Model Fit on Data")
    plt.show()

    #get the residuals
    residuals=model_fit.resid
    #plot the residuals
    fig,ax=plt.subplots(1,2)

    residuals.plot(title='Residuals',ax=ax[0])
    residuals.plot(kind='kde',title='probability distribution of residuals',ax=ax[1])
    #print(model_fit.)
    plt.show()

    #are the residuals random?
    print(residuals.describe())
    #autocorrelation plots of residuals
    six_plots(residuals)

    #Significance Level at 5%
    #alpha=0.05

    #The Box-jenkins Method
    Q,p=ljungbox(residuals,range(1,lags),boxpierce=False)
    c=[]
    for i in range(len(Q)):
        dof=i+1
        c.append(chi_sq_critical_val(alpha,dof))
        #print('Chi-statistic(Q) :',Q[i],'  p-value:',p[i],'   critical value: ',c," KEEP H0" if Q[i]<c else "DNT KEEP H0")

    #plot Q versus c
    #accept if Q stays below the 45 deg line i.e Q<c
    arstr="ARIMA"+str(order)+""
    plt.plot(c,Q,label=arstr)
    plt.plot(c,c,label='c=Q')
    plt.xlabel('Q values')
    plt.ylabel('critical values')
    plt.title('Box-Jenkins Test')
    plt.legend()
    plt.show()
    return model_fit
```

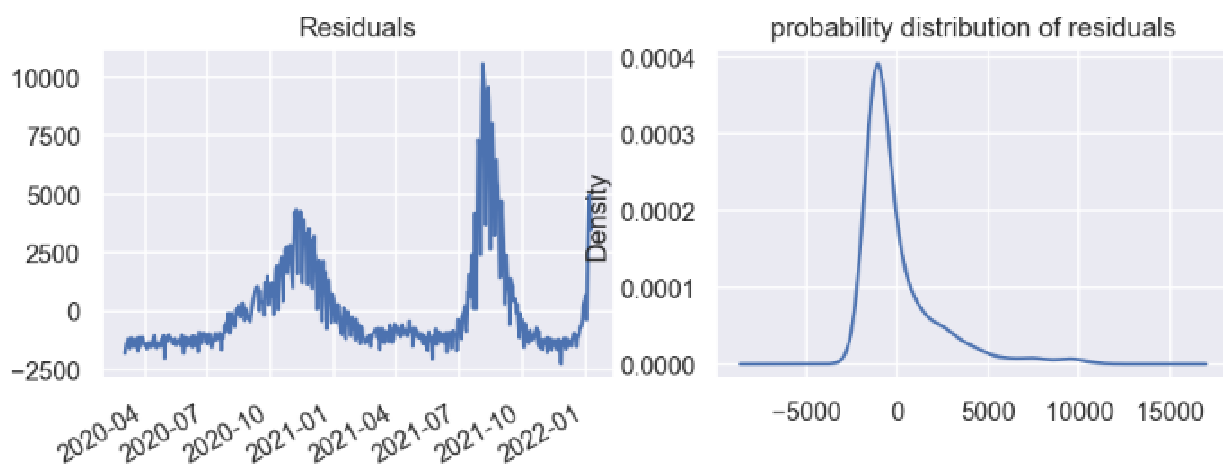
```python
from pandas.plotting import lag_plot

def six_plots(df):

    df=df.dropna()
    plt.rcParams.update({'figure.figsize':(9,5), 'figure.dpi':100})
    fontdict={'fontsize':9,'verticalalignment':'bottom'}
    fig,ax=plt.subplots(2,3)
    df.plot(ax=ax[0,0])
    df.hist(ax=ax[0,1]) #must be gaussian like
    sm.qqplot(df,ax=ax[0,2],line='45') # how close does the series fit the normal distribution. Quantile-Quantile
    lag_plot(df,ax=ax[1,0]) #Lag-1 plot to see autocorrelations
    plot_acf(df,ax=ax[1,1],title='') #acf plot
    plot_pacf(df,ax=ax[1,2],title='') #pacf plot

    left = 0.45
    bottom = -0.5
    top = 1.2

    ax[0,0].text(left, top, 'run sequence',
        horizontalalignment='left',
        verticalalignment='top',
        transform=ax[0,0].transAxes)
    ax[0,1].text(left, top, 'histogramme',
        horizontalalignment='left',
        verticalalignment='top',
        transform=ax[0,1].transAxes)
    ax[0,2].text(left, top, 'Q-Q',
        horizontalalignment='left',
        verticalalignment='top',
        transform=ax[0,2].transAxes)
    ax[0,2].set_xlabel('')
    ax[0,2].set_ylabel('')

    ax[1,0].text(left, bottom, 'Lag-plot',
        horizontalalignment='left',
        verticalalignment='bottom',
        transform=ax[1,0].transAxes)
    ax[1,1].text(left, bottom, 'ACF',
        horizontalalignment='left',
        verticalalignment='bottom',
        transform=ax[1,1].transAxes)
    ax[1,2].text(left, bottom, 'PACF',
        horizontalalignment='left',
        verticalalignment='bottom',
        transform=ax[1,2].transAxes)

    fig.tight_layout()
    fig.suptitle("")
    plt.show()
```

ARIMA(0,1,1)

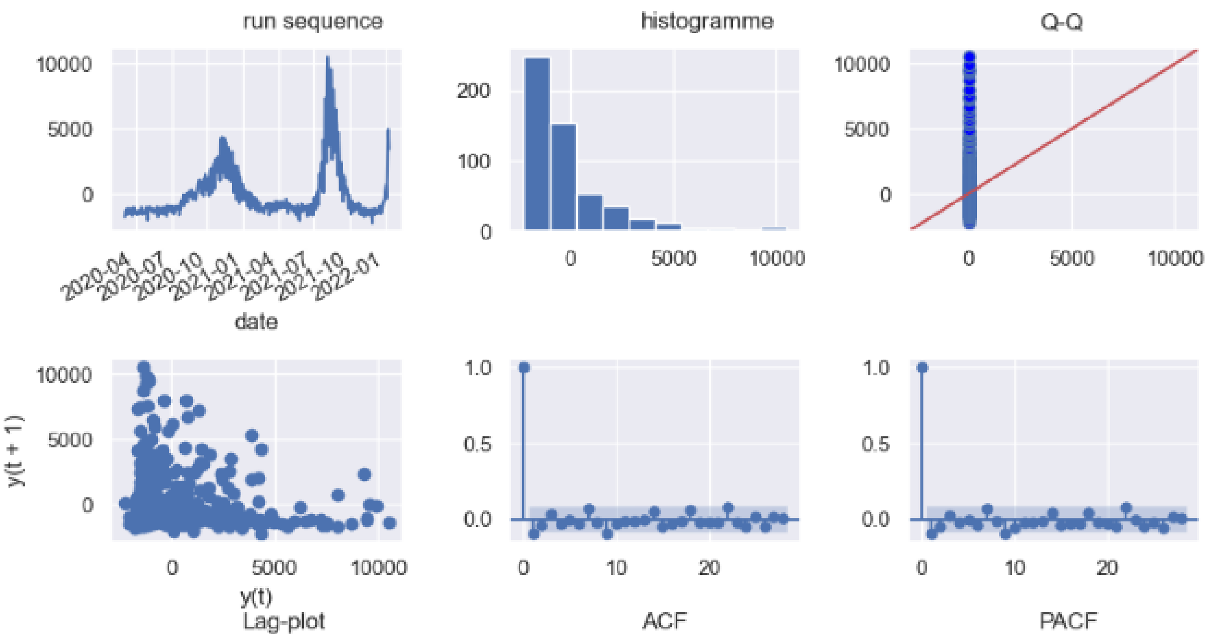
```python
arima_011=eval_arima(train,order=(0,1,1),lags=36)
```

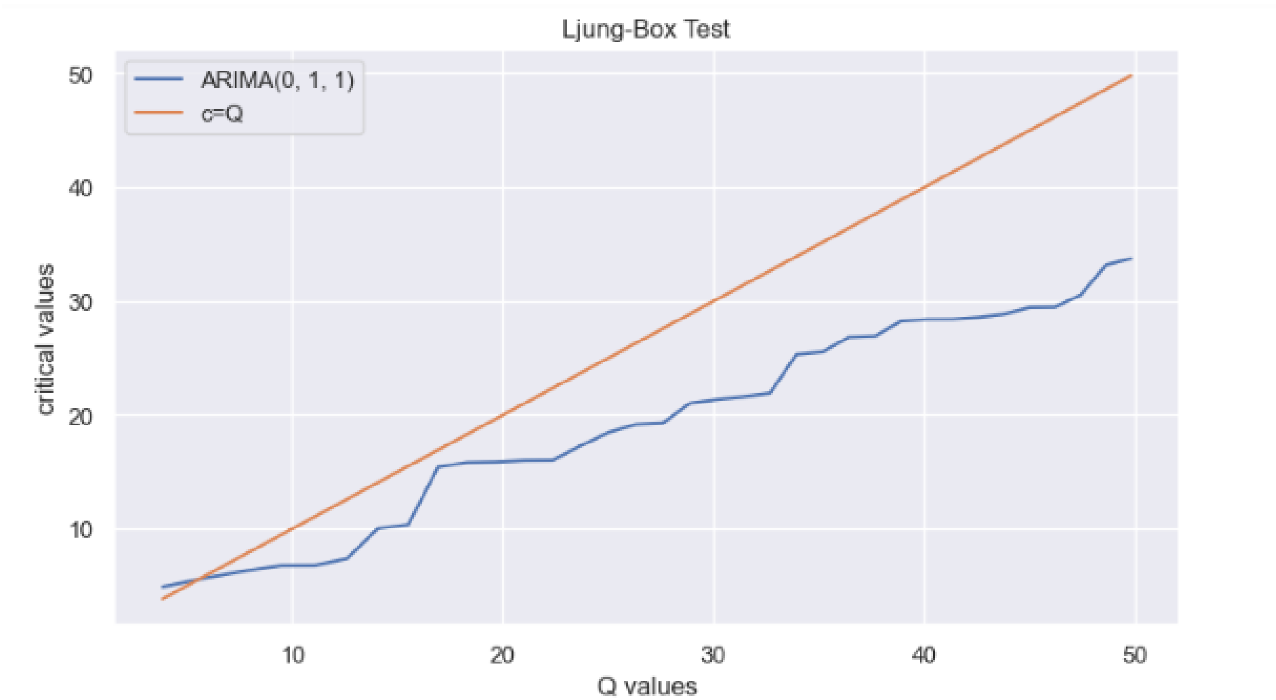
```
                             ARIMA Model Results
==============================================================================
Dep. Variable:           D.new_cases   No. Observations:                  543
Model:                ARIMA(0, 1, 1)   Log Likelihood               -4926.332
Method:                      css-mle   S.D. of innovations           2095.830
Date:               Fri, 14 Jan 2022   AIC                           9858.664
Time:                       19:17:24   BIC                           9871.556
Sample:                            1   HQIC                          9863.705

=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                -1.2682      0.572     -2.216      0.027      -2.390      -0.147
ma.L1.D.new_cases    -1.0000      0.005   -201.142      0.000      -1.010      -0.990
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
MA.1            1.0000           +0.0000j            1.0000            0.0000
```





```
count      543.000000
mean        10.525654
std       2108.252828
min      -2256.292962
25%      -1221.890611
50%       -885.669383
75%        408.407645
max      10548.108427
dtype: float64
```

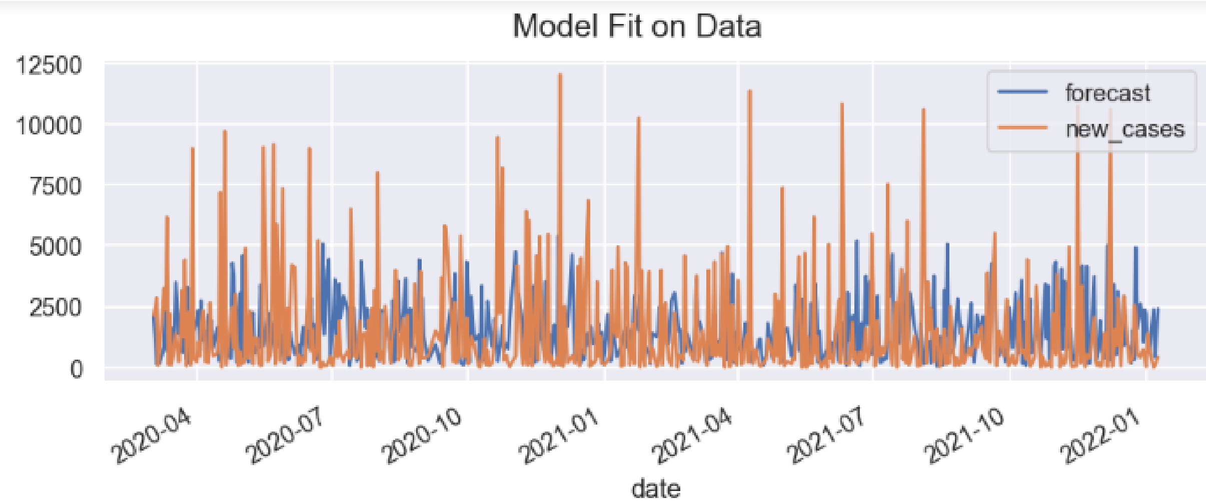


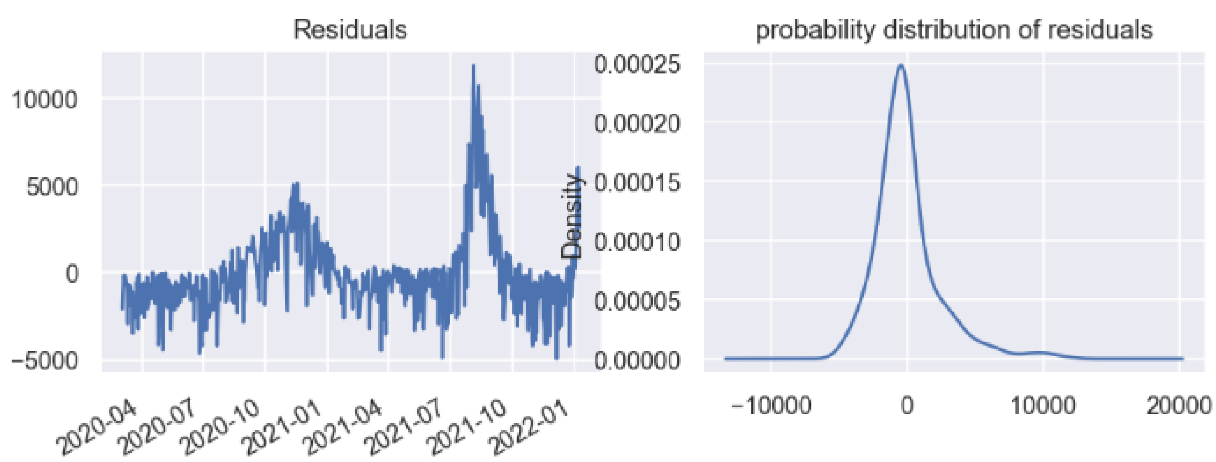


ARIMA(2,1,0)

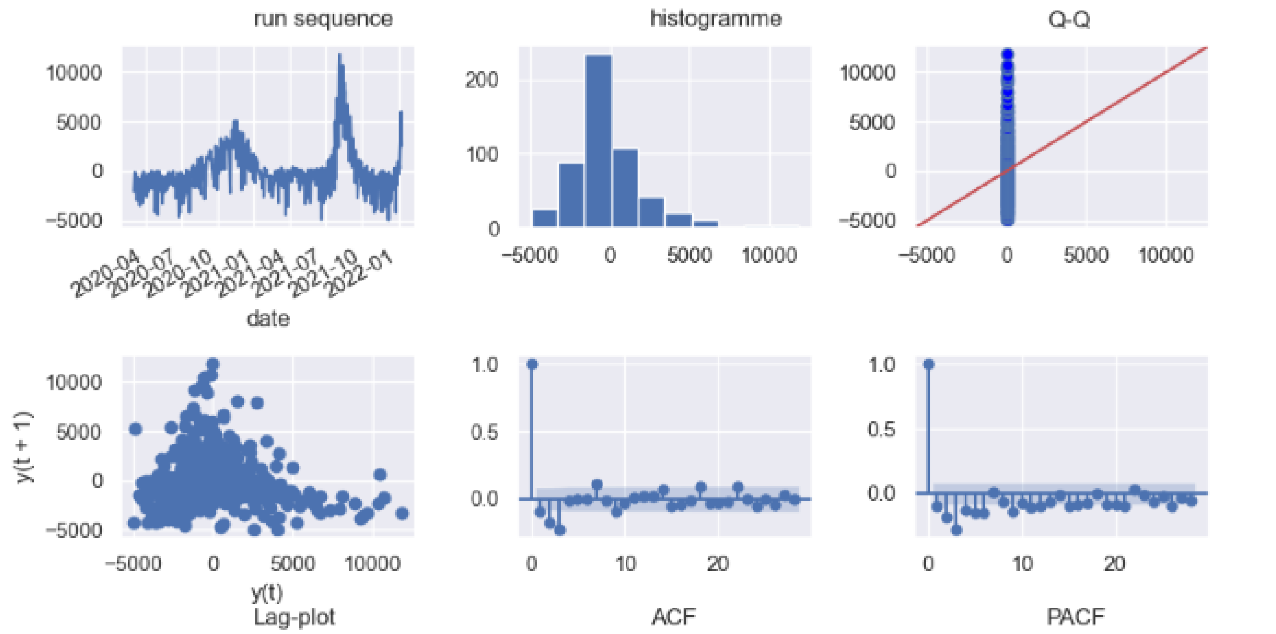
```
arima_210=eval_arima(train,order=(2,1,0),lags=36)
```

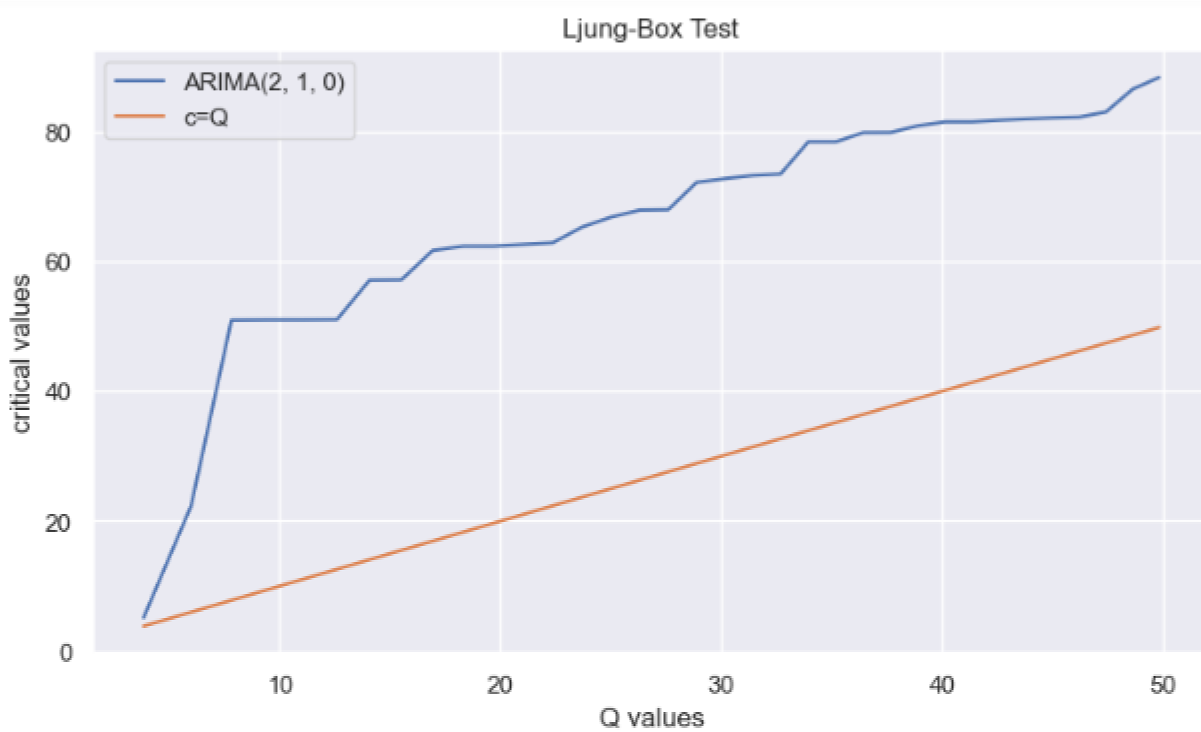
```
==============================================================================
Dep. Variable:            D.new_cases   No. Observations:                  543
Model:                 ARIMA(2, 1, 0)   Log Likelihood               -5003.074
Method:                       css-mle   S.D. of innovations           2426.543
Date:                Fri, 14 Jan 2022   AIC                          10014.148
Time:                        19:09:24   BIC                          10031.336
Sample:                             1   HQIC                         10020.868

=======================================================================================
                          coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------------
const                  -3.1103     49.148     -0.063      0.950     -99.439      93.218
ar.L1.D.new_cases      -0.7305      0.039    -18.522      0.000      -0.808      -0.653
ar.L2.D.new_cases      -0.3910      0.039     -9.929      0.000      -0.468      -0.314
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -0.9341           -1.2980j            1.5992           -0.3493
AR.2           -0.9341           +1.2980j            1.5992            0.3493
```





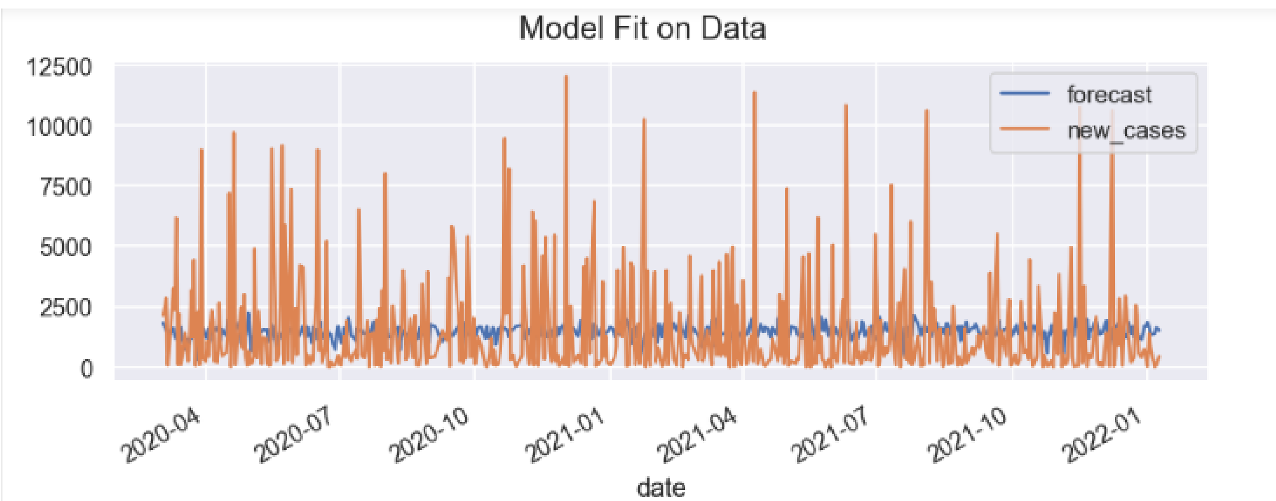
```
count       543.000000
mean          1.625014
std        2428.919819
min       -4947.830028
25%       -1344.488717
50%        -340.087701
75%         661.141954
max       11848.863532
dtype: float64
```

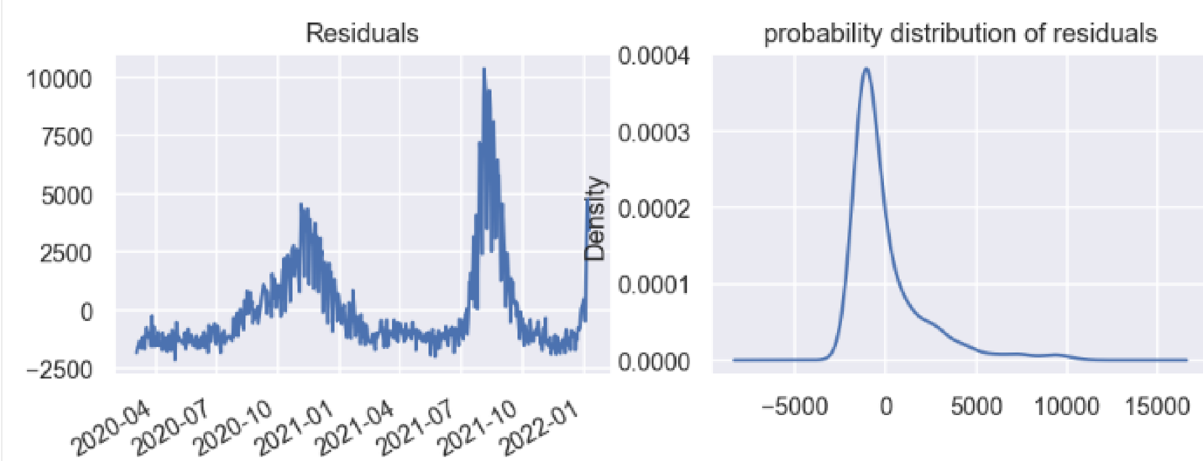




ARIMA(2,1,1)

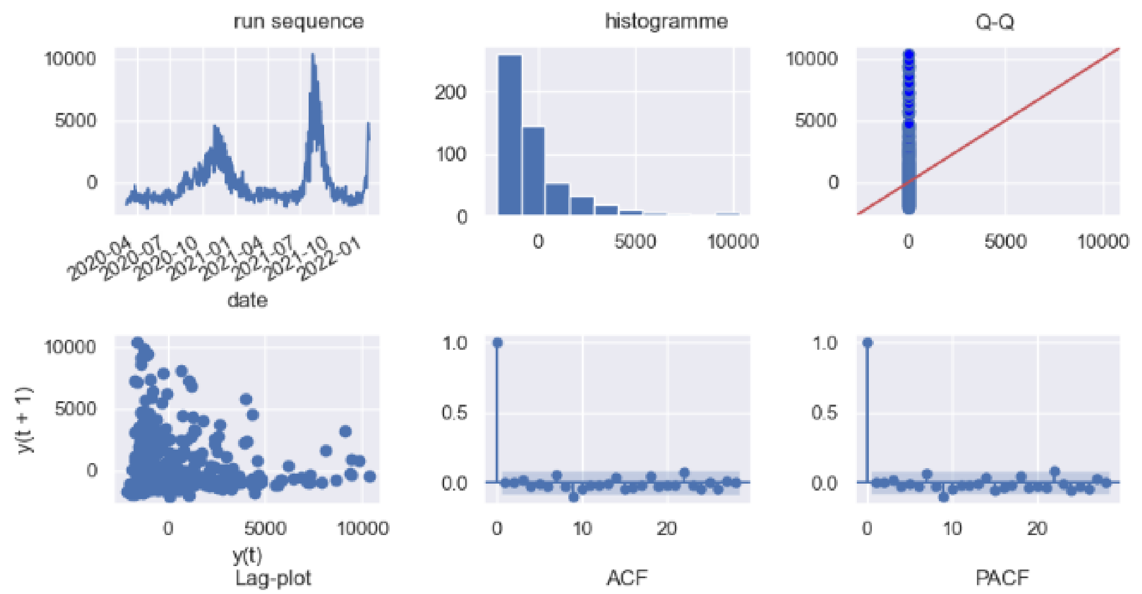
```
                             ARIMA Model Results
==============================================================================
Dep. Variable:            D.new_cases   No. Observations:                  543
Model:                 ARIMA(2, 1, 1)   Log Likelihood               -4923.395
Method:                       css-mle   S.D. of innovations           2084.054
Date:                Fri, 14 Jan 2022   AIC                           9856.790
Time:                        19:15:56   BIC                           9878.276
Sample:                             1   HQIC                          9865.191

=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
const                -1.2663      0.498     -2.543      0.011      -2.242      -0.290
ar.L1.D.new_cases    -0.0973      0.043     -2.270      0.023      -0.181      -0.013
ar.L2.D.new_cases    -0.0465      0.043     -1.084      0.278      -0.130       0.038
ma.L1.D.new_cases    -0.9999      0.005   -194.101      0.000      -1.010      -0.990
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -1.0477           -4.5199j            4.6397           -0.2863
AR.2           -1.0477           +4.5199j            4.6397            0.2863
MA.1            1.0001           +0.0000j            1.0001            0.0000
-----------------------------------------------------------------------------
```

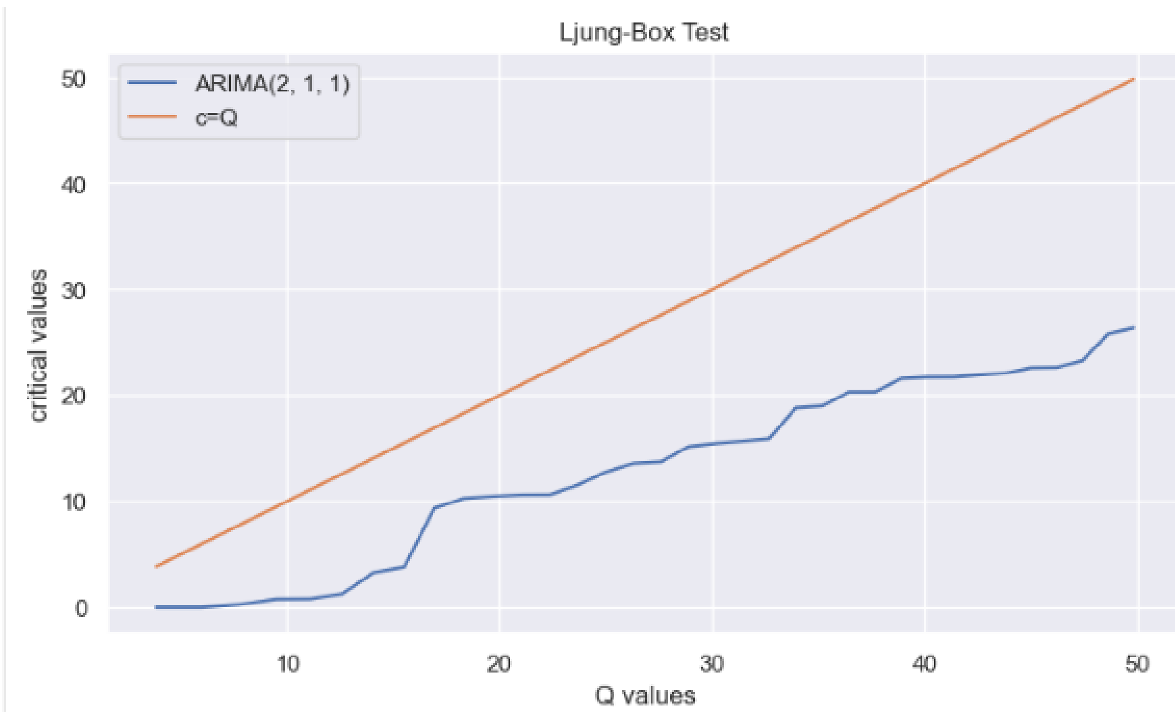




```
count      543.000000
mean        11.488239
std       2096.493706
min      -2130.558425
25%      -1236.539560
50%       -818.225365
75%        386.139932
max      10360.912884
dtype: float64
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
```





ARIMA(2,1,1) looks like a good choice.

Here is the GitHub repo link:

https://github.com/Boujbair/Time-Series-Covid19-Morocco

END