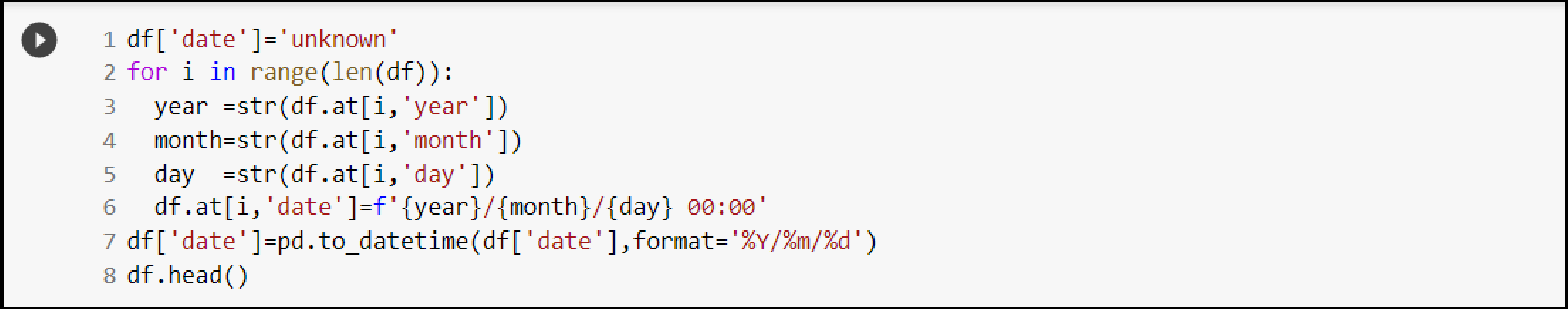
HW5 : Time Series

By : 6338125721 Boonthicha Sae-jia

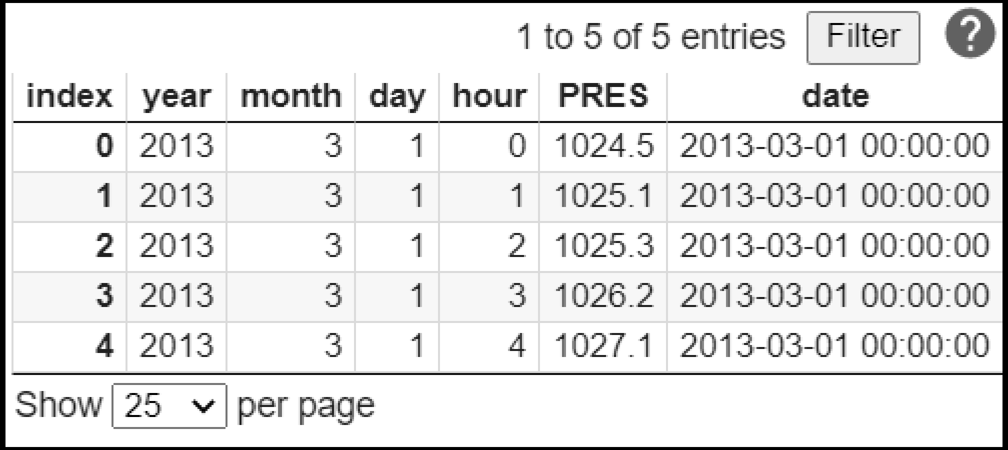
Code

https://colab.research.google.com/drive/1rL4B8_7a4D9MJHDxI-q8UoZ1WKoMAvad?usp=sharing

1 Select attributes



```python
df['date']='unknown'
for i in range(len(df)):
  year =str(df.at[i,'year'])
  month=str(df.at[i,'month'])
  day  =str(df.at[i,'day'])
  df.at[i,'date']=f'{year}/{month}/{day} 00:00'
df['date']=pd.to_datetime(df['date'],format='%Y/%m/%d')
df.head()
```



1 to 5 of 5 entries
Filter
?

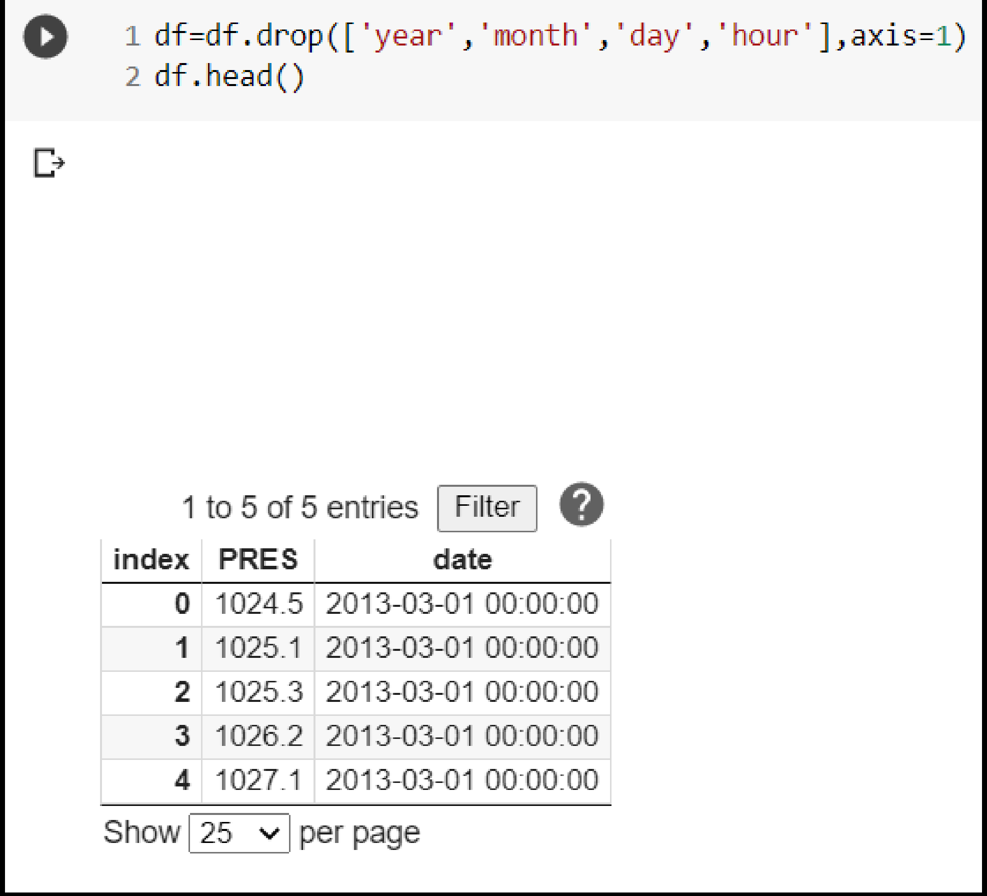
index	year	month	day	hour	PRES	date
0	2013	3	1	0	1024.5	2013-03-01 00:00:00
1	2013	3	1	1	1025.1	2013-03-01 00:00:00
2	2013	3	1	2	1025.3	2013-03-01 00:00:00
3	2013	3	1	3	1026.2	2013-03-01 00:00:00
4	2013	3	1	4	1027.1	2013-03-01 00:00:00

Show 25 per page

Select year, month, day, hour, and PRES as the attribute before create date from the first four attributes

2 Group data

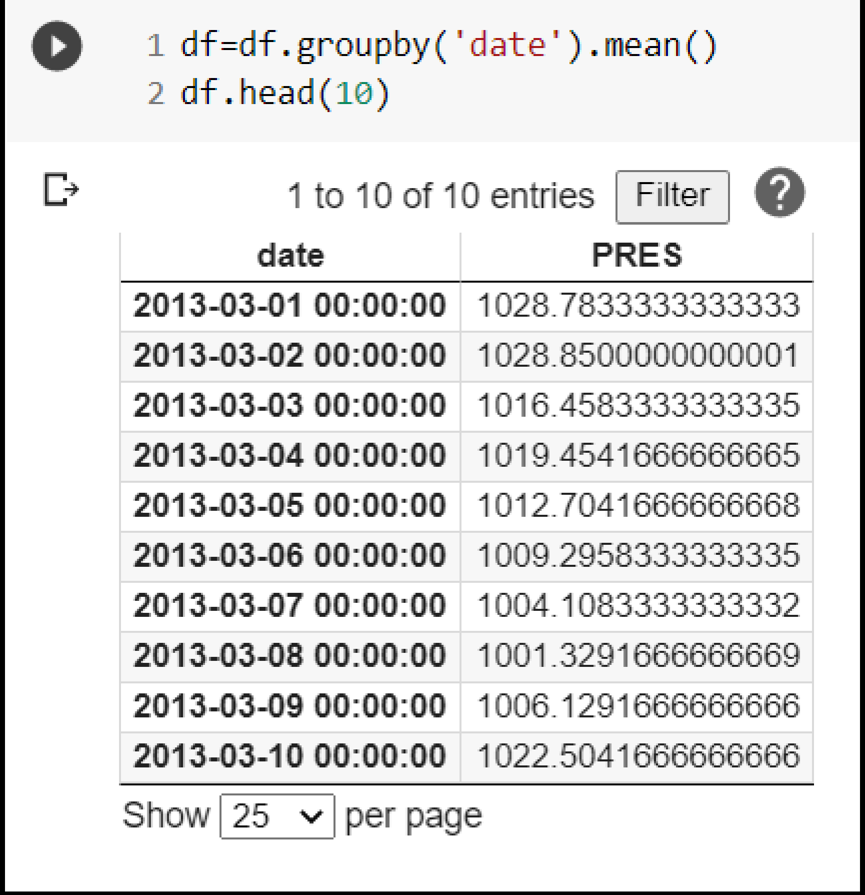
Aggregate pressure daily by average



```python
df=df.drop(['year','month','day','hour'],axis=1)
df.head()
```

index	PRES	date
0	1024.5	2013-03-01 00:00:00
1	1025.1	2013-03-01 00:00:00
2	1025.3	2013-03-01 00:00:00
3	1026.2	2013-03-01 00:00:00
4	1027.1	2013-03-01 00:00:00

Firstly, drop unnecessary columns

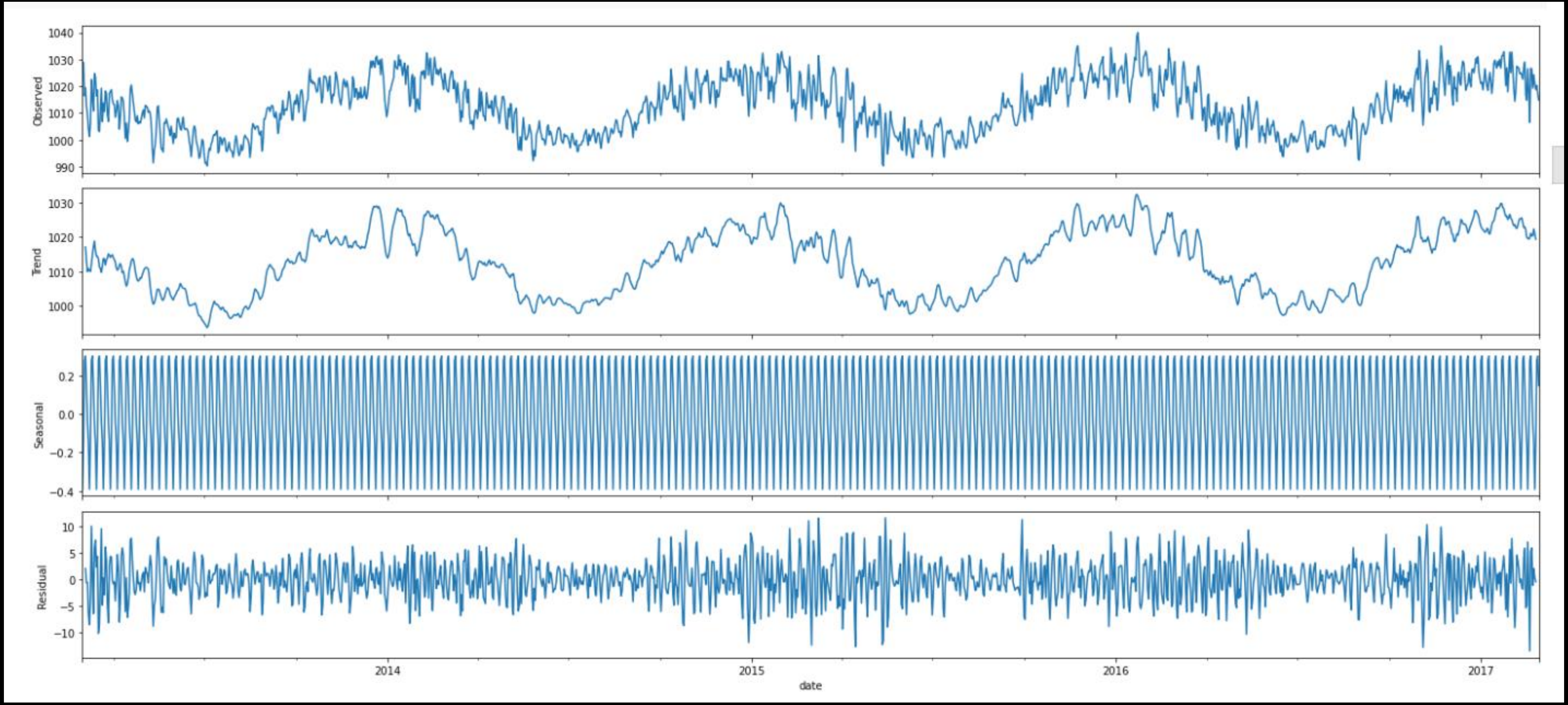


```python
df=df.groupby('date').mean()
df.head(10)
```

date	PRES
2013-03-01 00:00:00	1028.7833333333333
2013-03-02 00:00:00	1028.8500000000001
2013-03-03 00:00:00	1016.4583333333335
2013-03-04 00:00:00	1019.4541666666665
2013-03-05 00:00:00	1012.7041666666668
2013-03-06 00:00:00	1009.2958333333335
2013-03-07 00:00:00	1004.1083333333332
2013-03-08 00:00:00	1001.3291666666669
2013-03-09 00:00:00	1006.1291666666666
2013-03-10 00:00:00	1022.5041666666666

Use function groupby to group pressure in same date

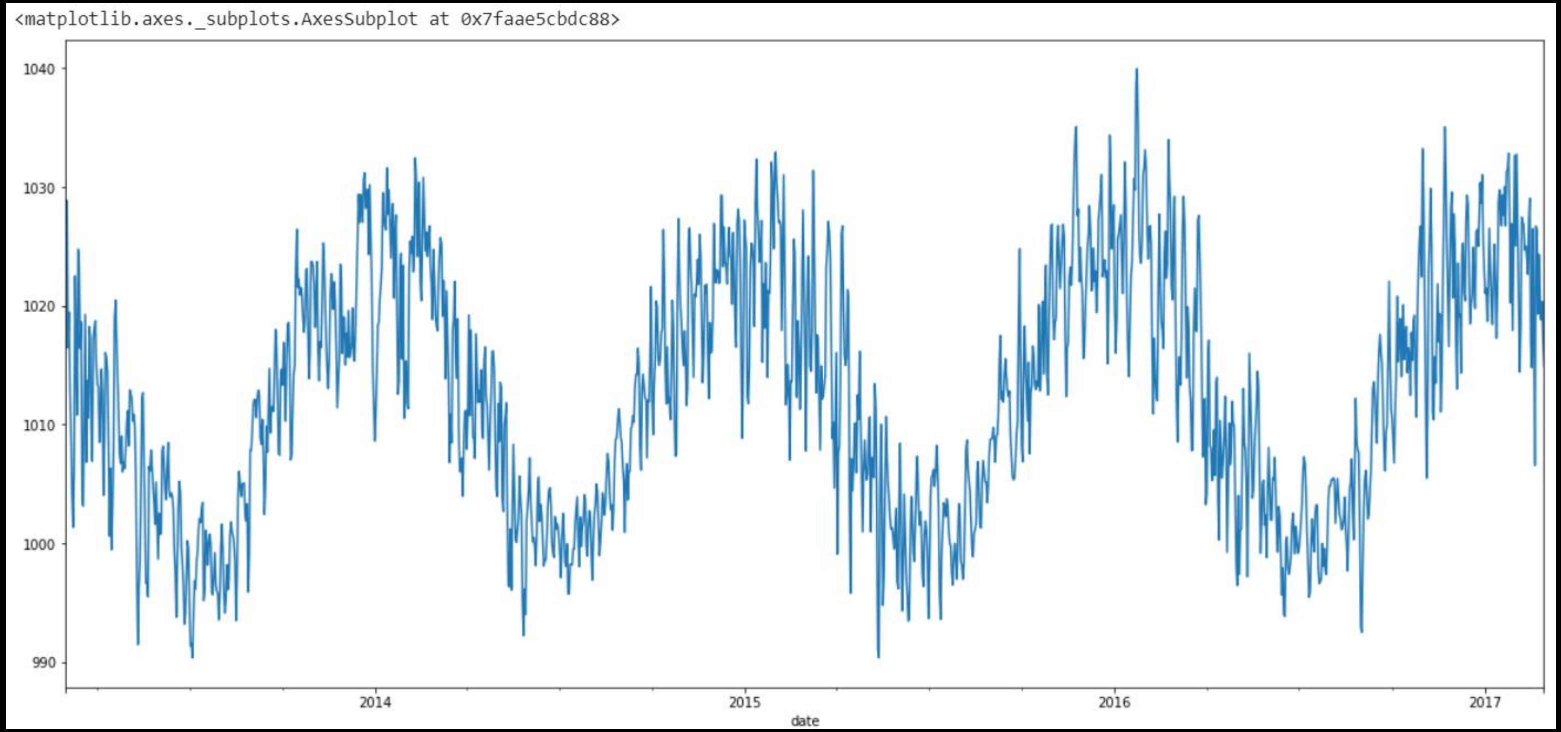
3 Visualize Date



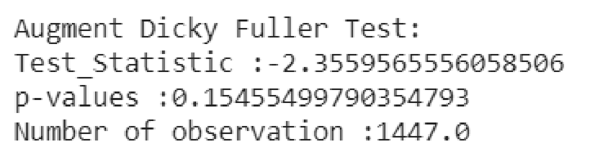
With sm from statsmodels.api, the observed plot is decomposed in trend, seasonal and residual

4 Calculate ACF and PACF

Before calculate ACF and PACF, the data should be stationary. Therefore, we take log and use the difference in log as the data that we will use the create the model.



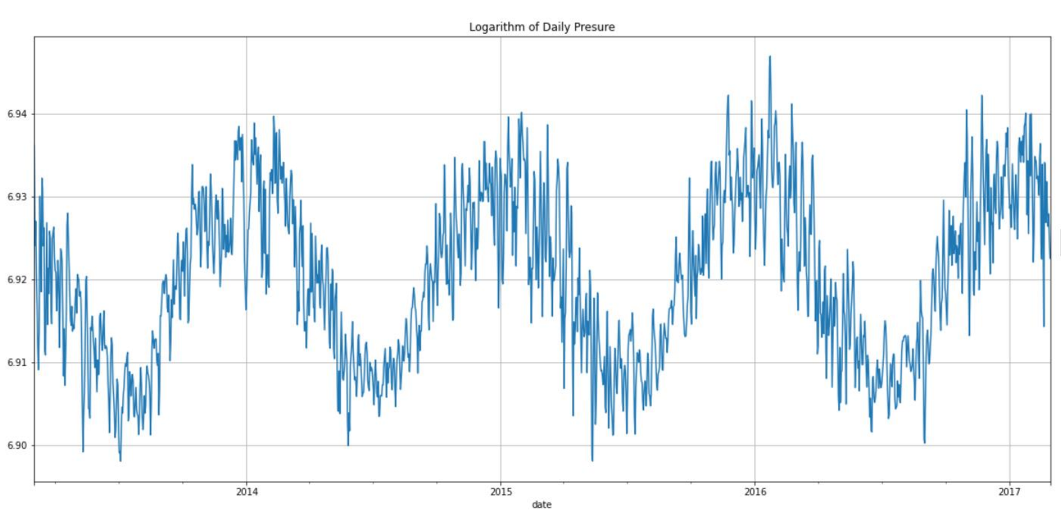
```
<matplotlib.axes._subplots.AxesSubplot at 0x7faae5cbdc88>
```

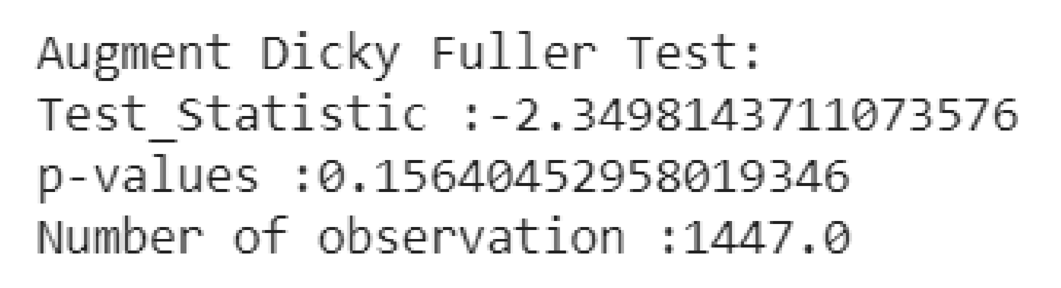


```
Augment Dicky Fuller Test:
Test_Statistic :-2.3559565556058506
p-values :0.15455499790354793
Number of observation :1447.0
```

The plot from actual data have p-values equal to 0.15455 which is more than >0.05

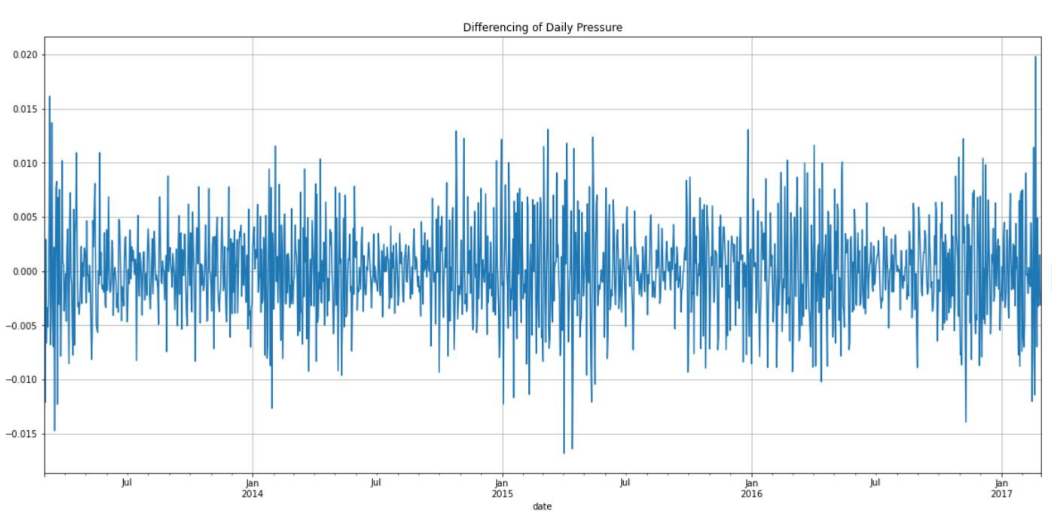
4 Calculate ACF and PACF

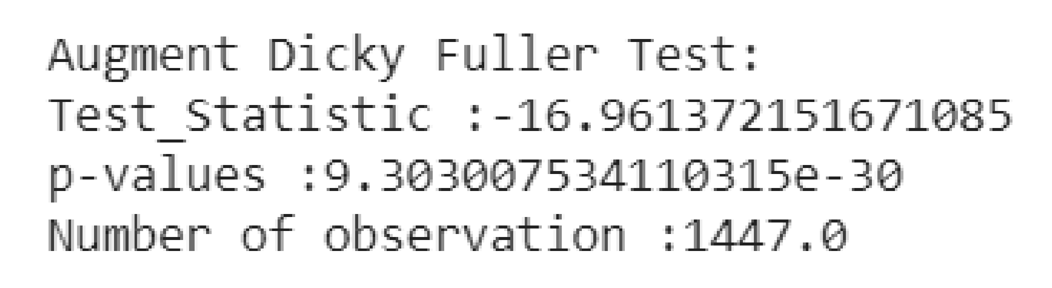




```
Augment Dicky Fuller Test:
Test_Statistic :-2.3498143711073576
p-values :0.15640452958019346
Number of observation :1447.0
```

After taking log, get p-values equal to 0.15640



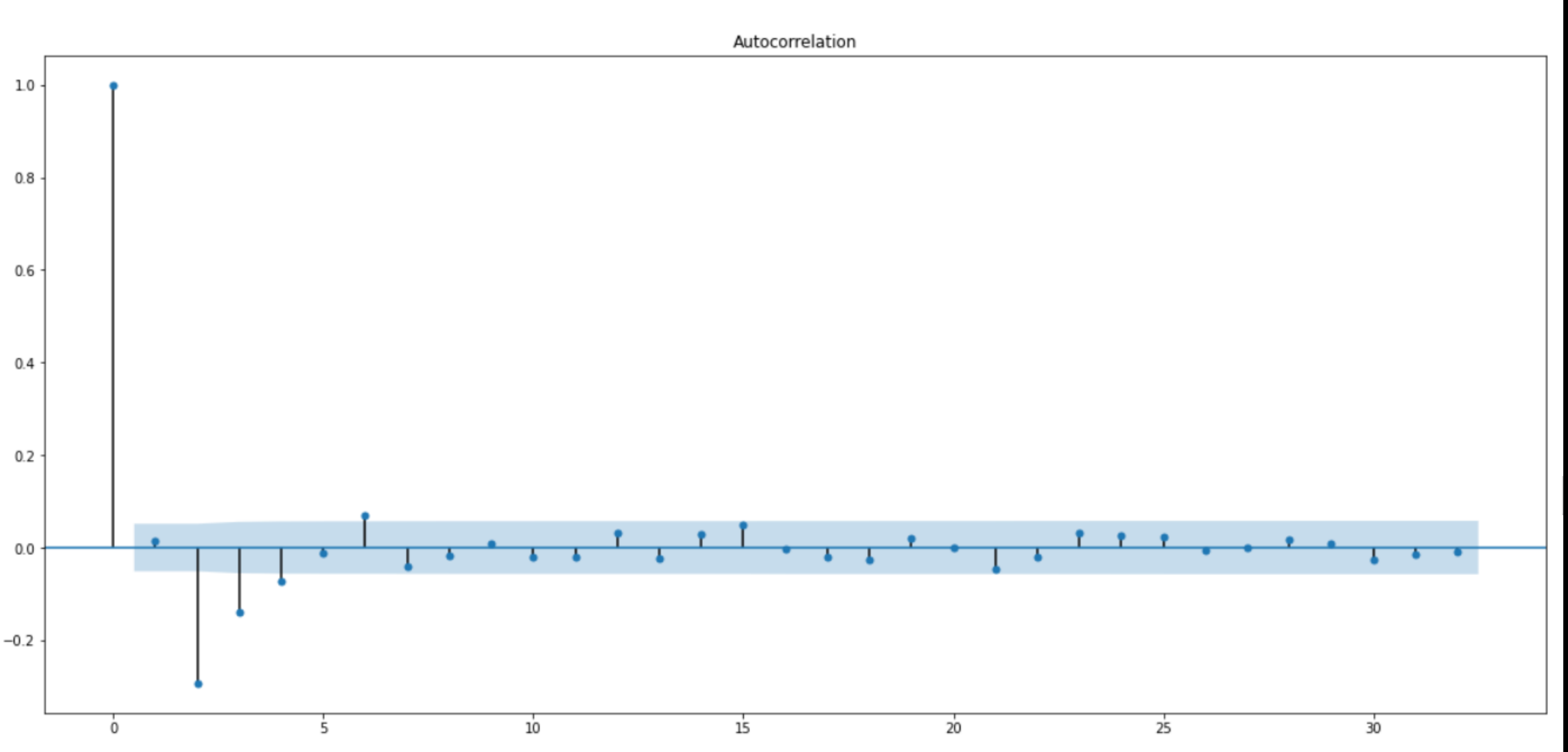


```
Augment Dicky Fuller Test:
Test_Statistic :-16.961372151671085
p-values :9.303007534110315e-30
Number of observation :1447.0
```

Using differencing, we get p-values less than 0.05

4 Calculate ACF and PACF

Use statsmodels library to get the array of ACF and PACF before looping to find the lag than ACF and PACF less than cut_point

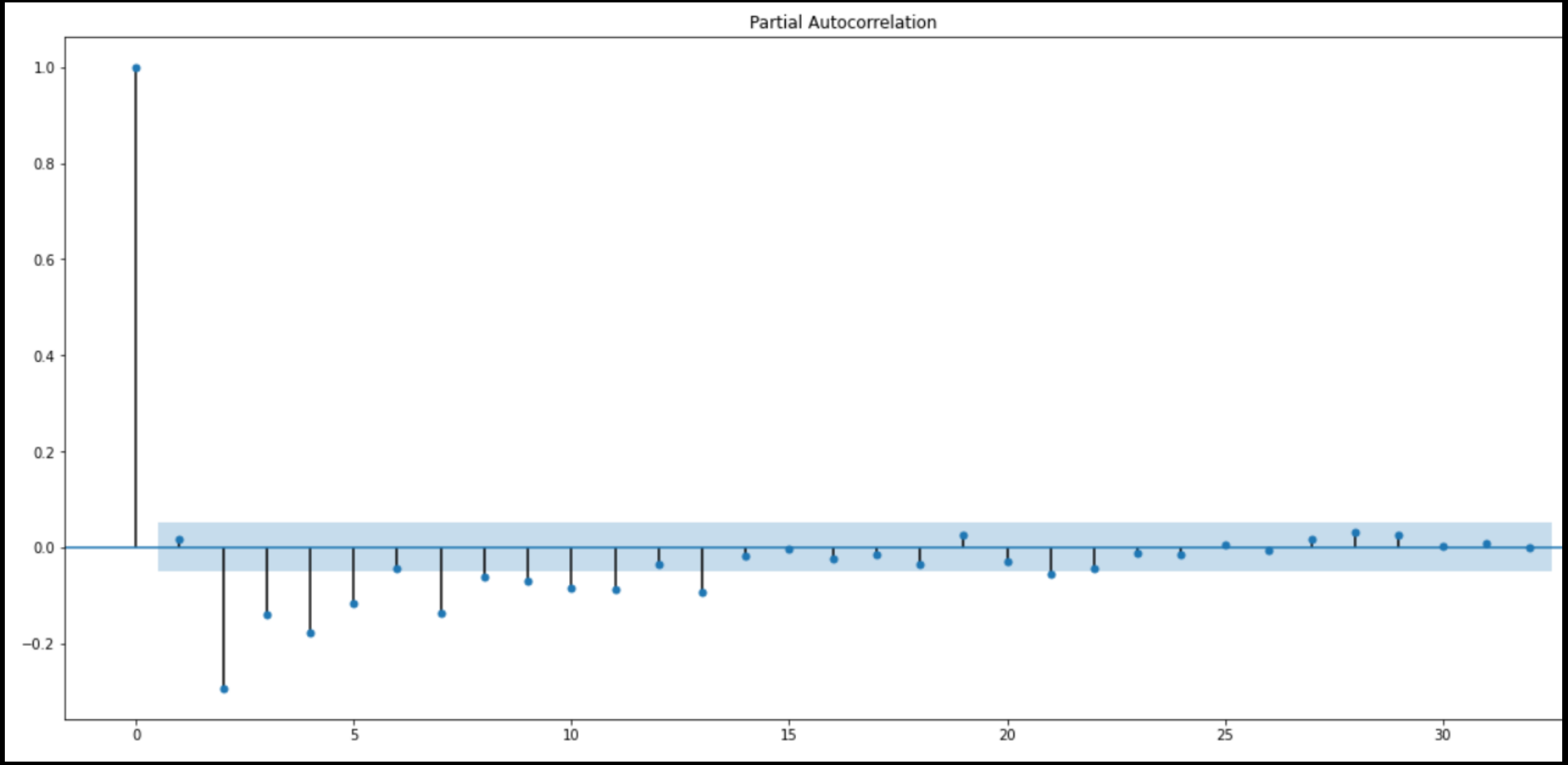


Then, q =1

The plot of ACF show that after 1st lag, value of ACF is less than cut_point

4 Calculate ACF and PACF

Use statsmodels library to get the array of ACF and PACF before looping to find the lag than ACF and PACF less than cut_point

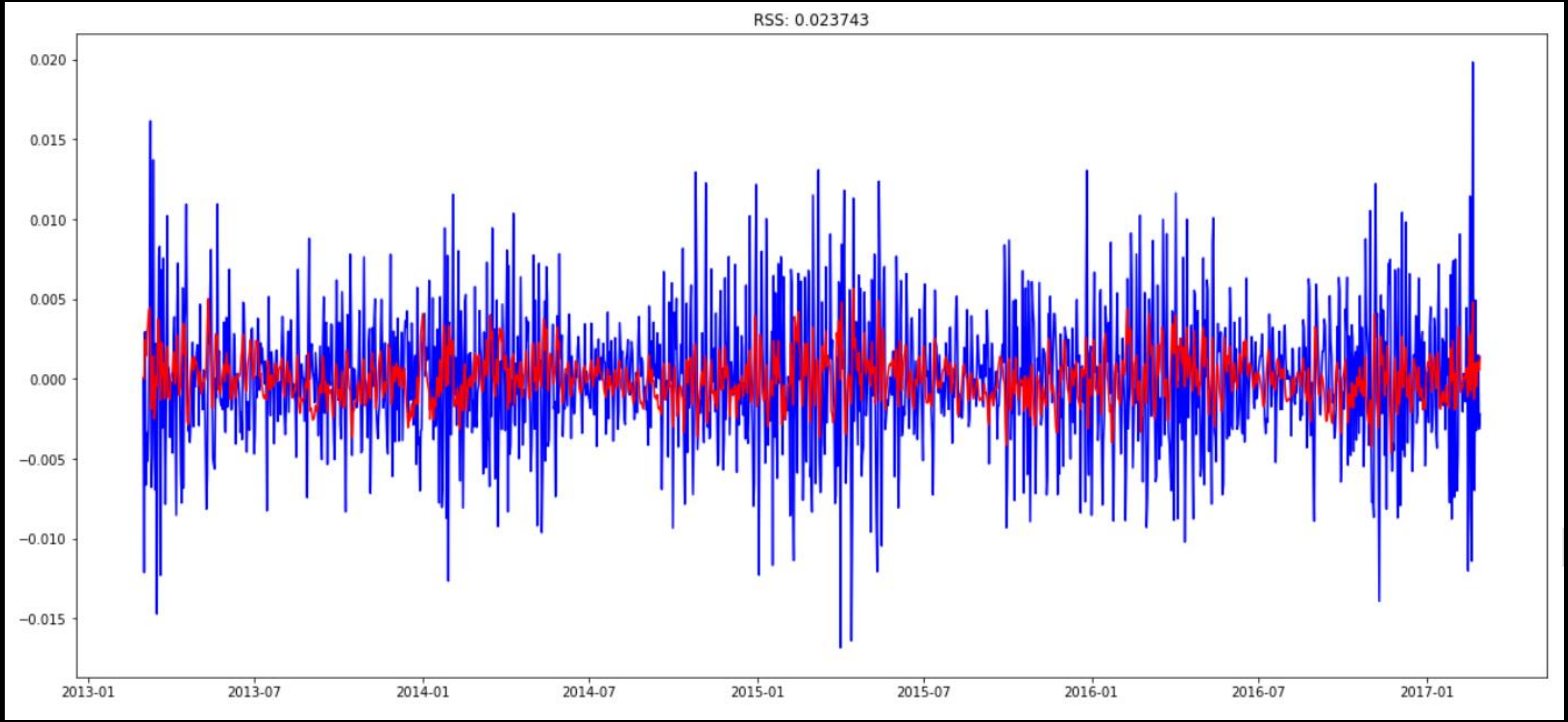


Then, p=1

The plot of ACF show that after 1st lag, value of PACF is less than cut_point

5 Apply ARIMA

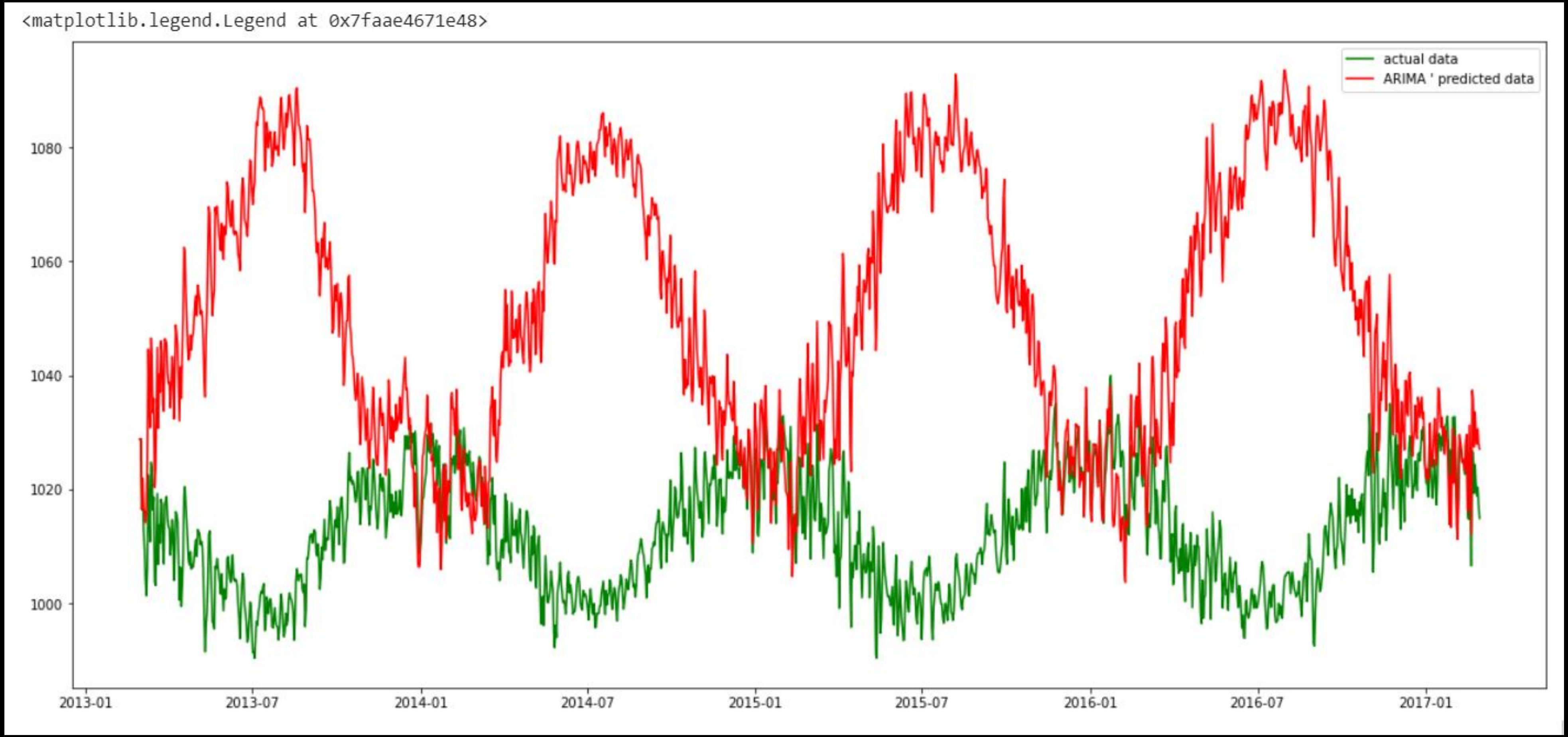
a) Create Model from p,q and d



Create model_1 using p=1,d=1,q=1 and get the less RSS score when compares to other model

5 Apply ARIMA

a) Change the data into the actual data and create the plot



```
<matplotlib.legend.Legend at 0x7faae4671e48>
```

The plot between predicted data and actual data seems to be inverse

6 Evaluate the result

Even though RSS score from model_1 is the smallest, model_1 can’t predicted the trend correctly. If we cut some period in the end and predict, the would predict more correctly or following the trend