Statistical_analysis

April 22, 2018

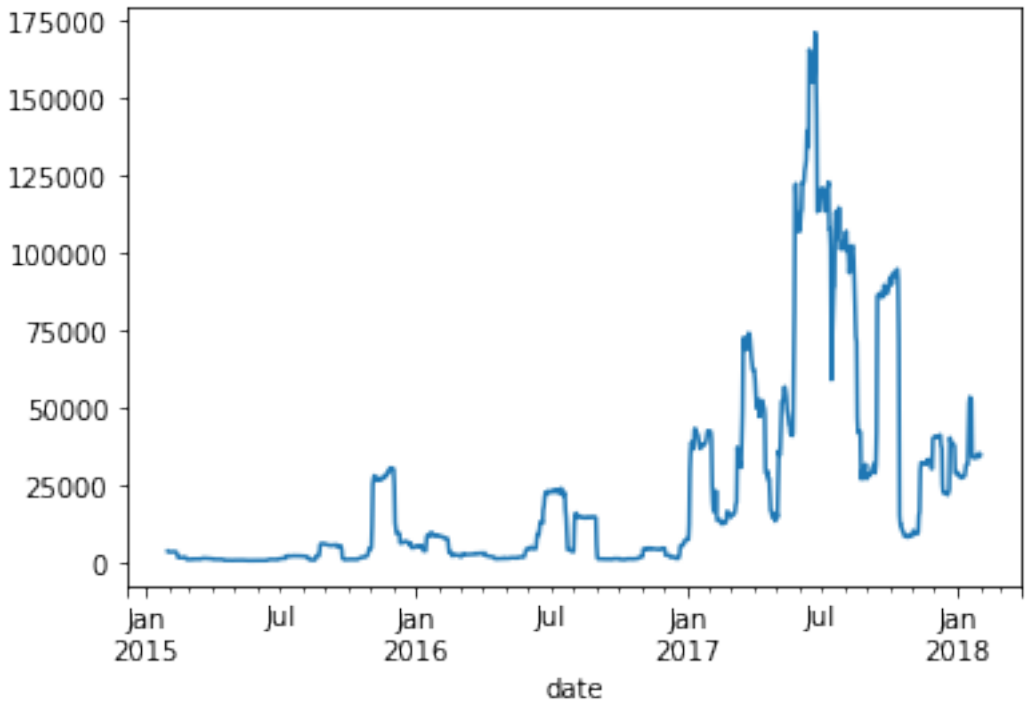
In [1]:
```
#Libraries

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os
from statsmodels.tsa import stattools
%matplotlib inline
```

```
C:\Users\User\Anaconda3\envs\rf\lib\site-packages\statsmodels\compat\pandas.py:56: FutureWarning
  from pandas.core import datetools
```

In [2]:
```
#Change paths, check all currencies

path = 'C:/Users/User/Desktop/crypto-analysis/train/'
filename = os.path.join(path,'train_btc.csv')
df = pd.read_csv(filename, sep = ',', encoding = 'UTF-8')
df['date'] = pd.to_datetime(df['date'], infer_datetime_format=True, dayfirst = True)
df = df.set_index('date')
headers = df.columns.tolist()
rows = df.index.tolist()
df1 = df.resample('D').mean()
```

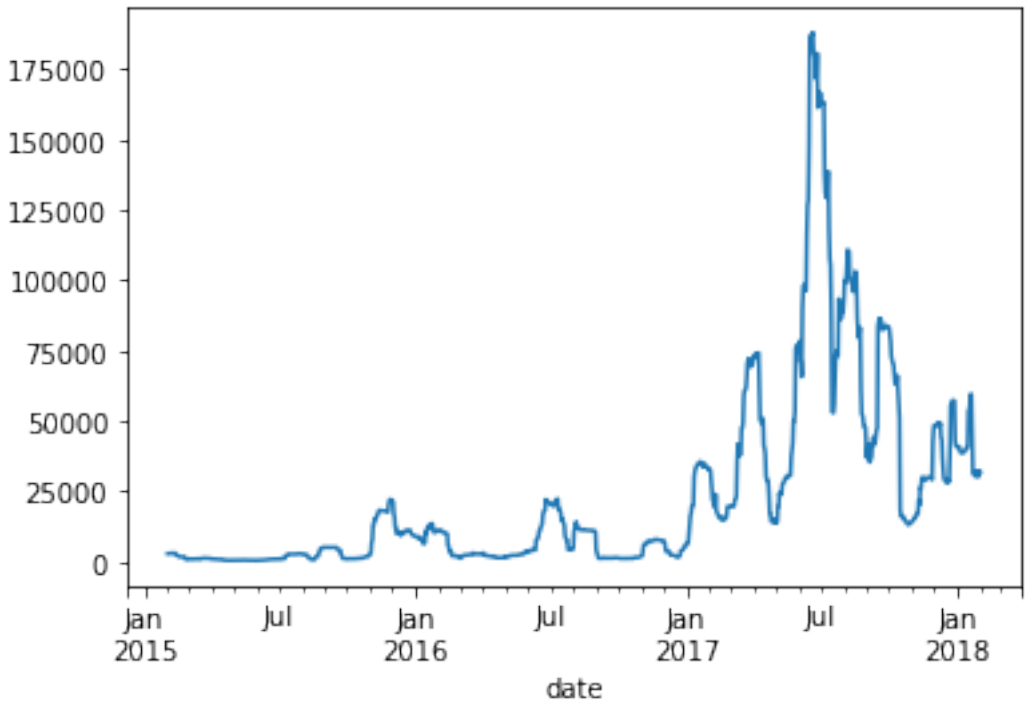
In [3]:
```
#How does the variance of the trading volume behave over time? Rather shaky...

df1.rolling(30).var()['volume'].plot()
```

Out[3]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x210a0d7f7f0>
```



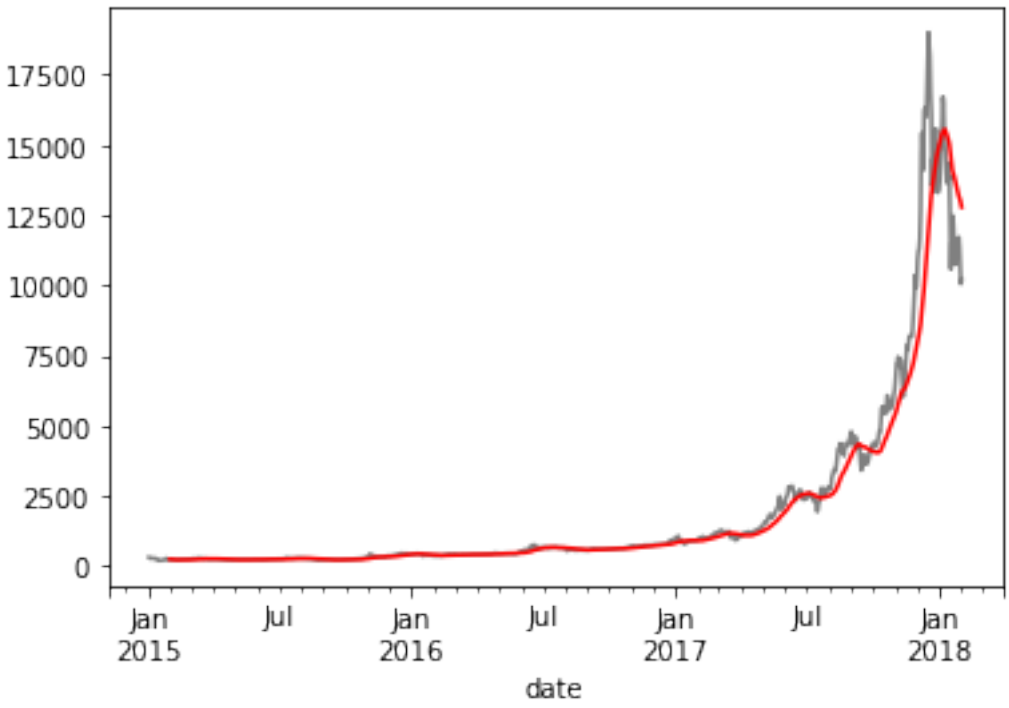
In [4]:
```
#What about the variance of the differenced volume? Similar results.

volume = df1.volume
volume_lagged = df1.volume.shift()
diffed_volume = volume - volume_lagged
diffed_volume.rolling(window = 30).var().plot()
```

Out[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x210973ddac8>
```



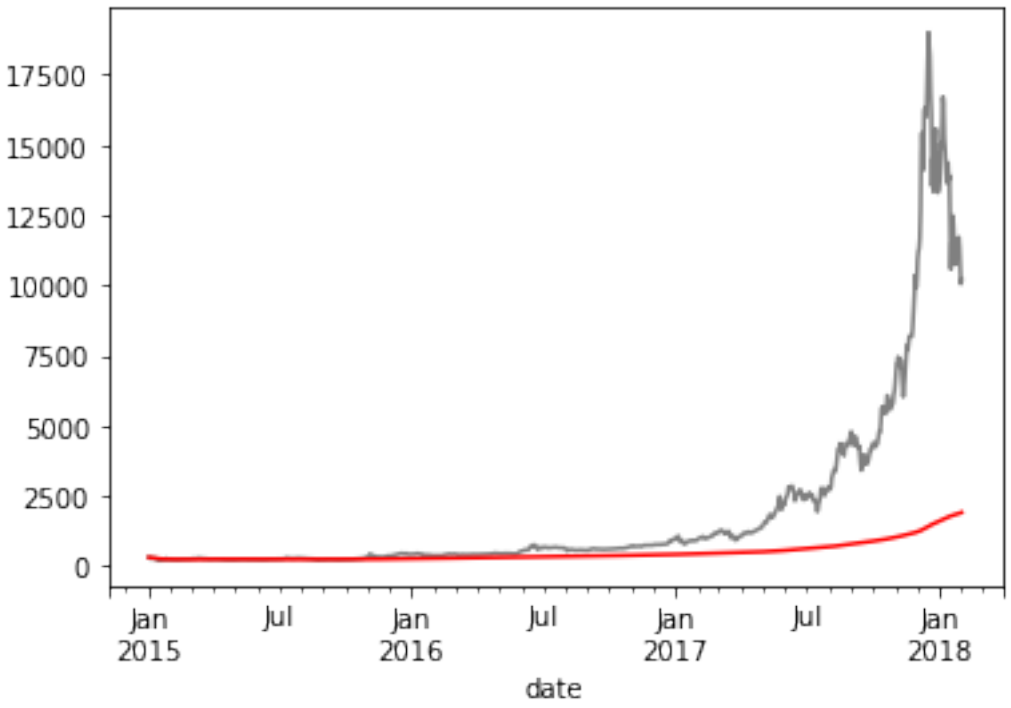
In [5]:
```
"""
The rolling window method (moving average in this case).
You can check different features here as well as different window sizes.
Or more generally, you can use different aggregations (e.g. count, min, max, variance, q
"""

df1['open'].plot(color='grey')
df1.rolling(30).mean()['open'].plot(color='red')
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x210a40d3780>

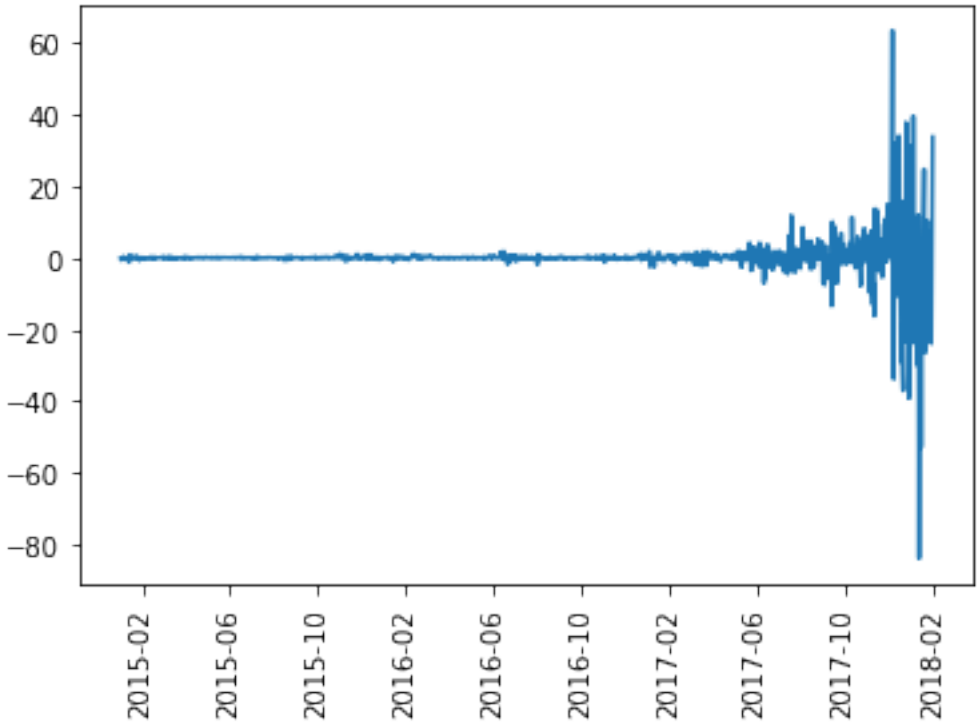


In [6]:
```
"""
Expanding windows are not exactly relevant with financial data, but I am adding it, for
You can think of it as long term memory, i.e. all your data points are equally important
Thus the red line is not as affected by the latest shenanigans as much as the rolling wi
"""

df1['open'].plot(color='grey')
df1.expanding().mean()['open'].plot(color='red')
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x210a40dacc0>
```




```
In [7]: df1['DayGain'] = df1.price_usd - df1.open
        plt.plot(df1['DayGain'])
        plt.xticks(rotation='vertical')

Out[7]: (array([735630., 735750., 735872., 735995., 736116., 736238., 736361.,
                736481., 736603., 736726.]), <a list of 10 Text xticklabel objects>)
```

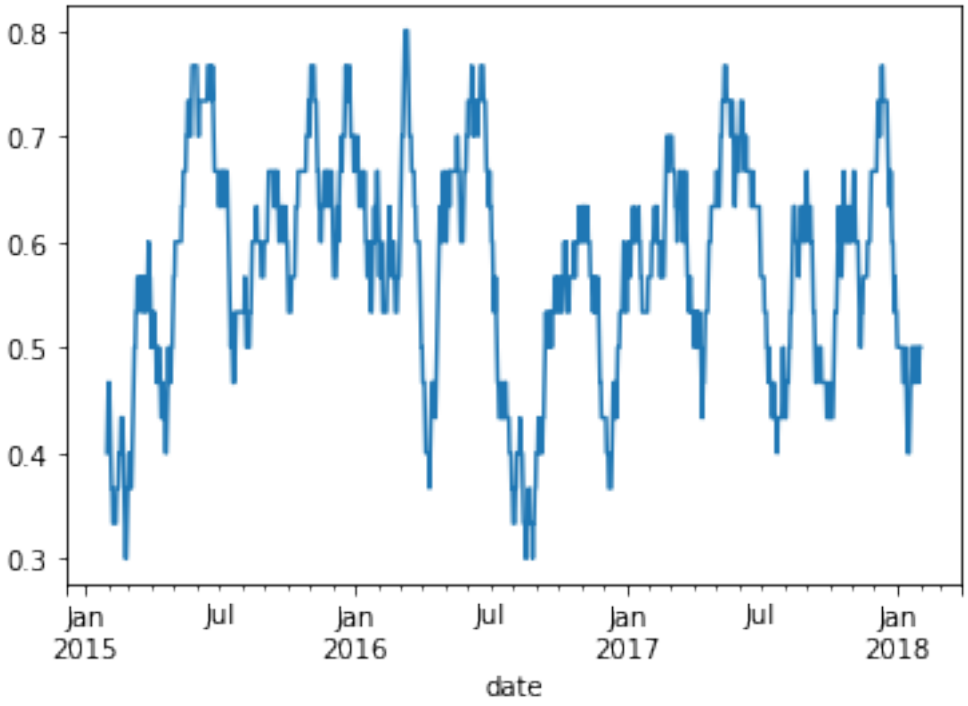



```
In [8]: #On 645 out of 1127 days the market close higher than opened.
        len(df1[df1.DayGain>0])
```

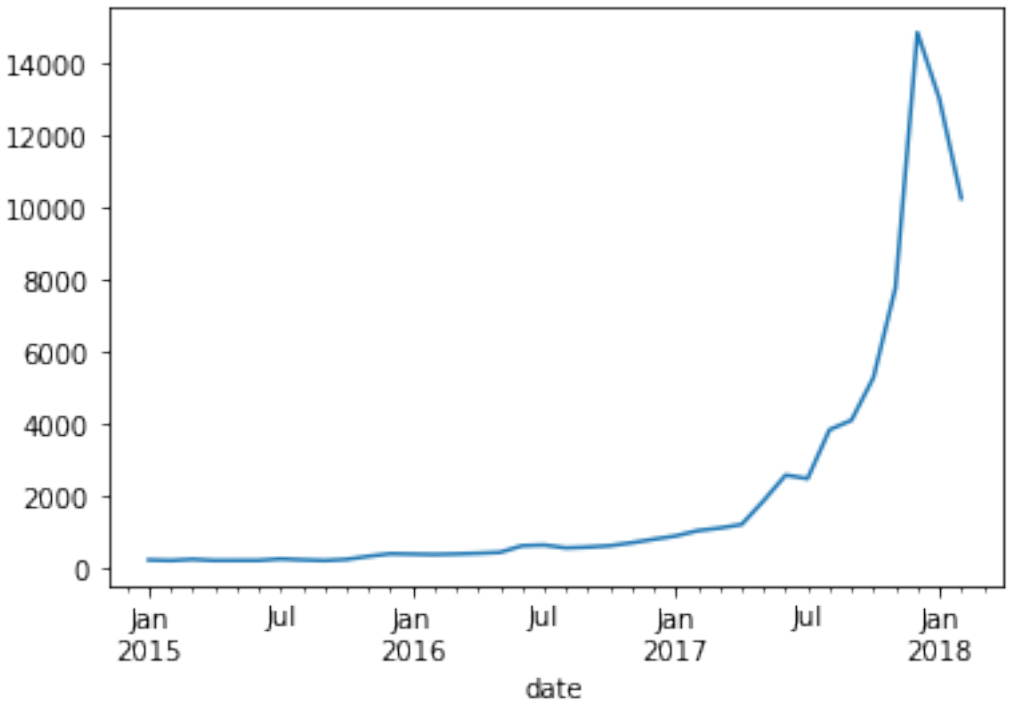
```
Out[8]: 645
```

```
In [9]: #Did the up days become more or less frequent over time?Here they seem randomely distrib
        df1.rolling(window = 30)['DayGain'].apply(lambda x: len([x_i for x_i in x if x_i > 0])/l
```

```
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x210a4335550>
```




```
In [10]: #Checking out the high values for every month.

         df.resample('M').mean()['high'].plot()

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x210a43f94e0>
```



In [11]:
```
#In the method parameter you can specify another type of correlation(kendall or spearma

correlations = df.corr(method='pearson')
```

In [12]:
```
#Nice interactive table of correlations across all features.

cmap = sns.diverging_palette(5, 250, as_cmap=True)

def magnify():
    return [dict(selector="th",
                 props=[("font-size", "7pt")]),
            dict(selector="td",
                 props=[('padding', "0em 0em")]),
            dict(selector="th:hover",
                 props=[("font-size", "12pt")]),
            dict(selector="tr:hover td:hover",
                 props=[('max-width', '200px'),
                        ('font-size', '12pt')])
]

correlations.style.background_gradient(cmap, axis=1)\
    .set_properties(**{'max-width': '80px', 'font-size': '10pt'})\
    .set_caption("Hover to magify")\
```

```
            .set_precision(2)\
            .set_table_styles(magnify())
```

Out[12]: <pandas.io.formats.style.Styler at 0x210a0d6cb38>

In [13]:
```
#Checking the correlation of  feature with itself, lagged.
#In the shift method, you can add any integer and shift the feature as back in time as

pd.DataFrame({'real':df1.open, 'lagged':df1.open.shift()}).corr()
```

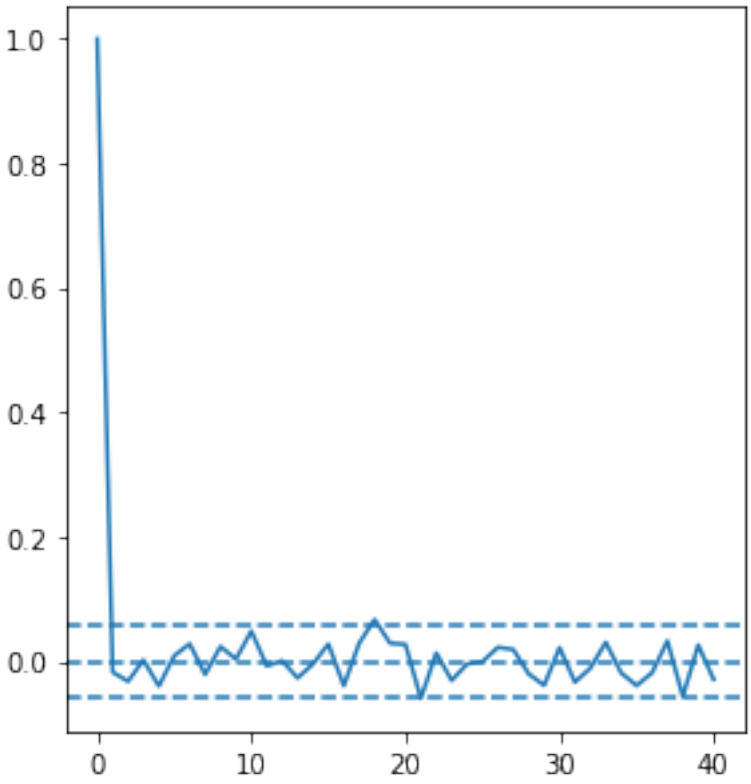
Out[13]:

	lagged	real
lagged	1.000000	0.997839
real	0.997839	1.000000

In [14]:
```
plt.rcParams["figure.figsize"]=(10,5) # Change the size of the plots
```

In [15]:
```
"""
Plotting the acf for log returns
Do not forget, acf plots are indicating internal structures, there is no need to keep a
Thus, we use the log returns, and not the actual prices.
For this case (BTC), neither significant correlations found over the time nor any parti
"""
acf_result = stattools.acf(df1['log_returns'])
plt.subplot(121)
plt.plot(acf_result)
plt.axhline(y=0,linestyle='--')
plt.axhline(y=-1.96/np.sqrt(len(df1['log_returns'])),linestyle='--')
plt.axhline(y=1.96/np.sqrt(len(df1['log_returns'])),linestyle='--')
```

Out[15]: <matplotlib.lines.Line2D at 0x210a478bda0>


1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40