Bitcoin

May 7, 2020

In [21]:
```
# https://www.statsmodels.org/stable/index.html
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
#plt.style.use('fivethirtyeight')
import itertools
```

In [22]:
```
final_date = pd.read_csv('datasetv1.csv')
final_date.head()
```

Out[22]:

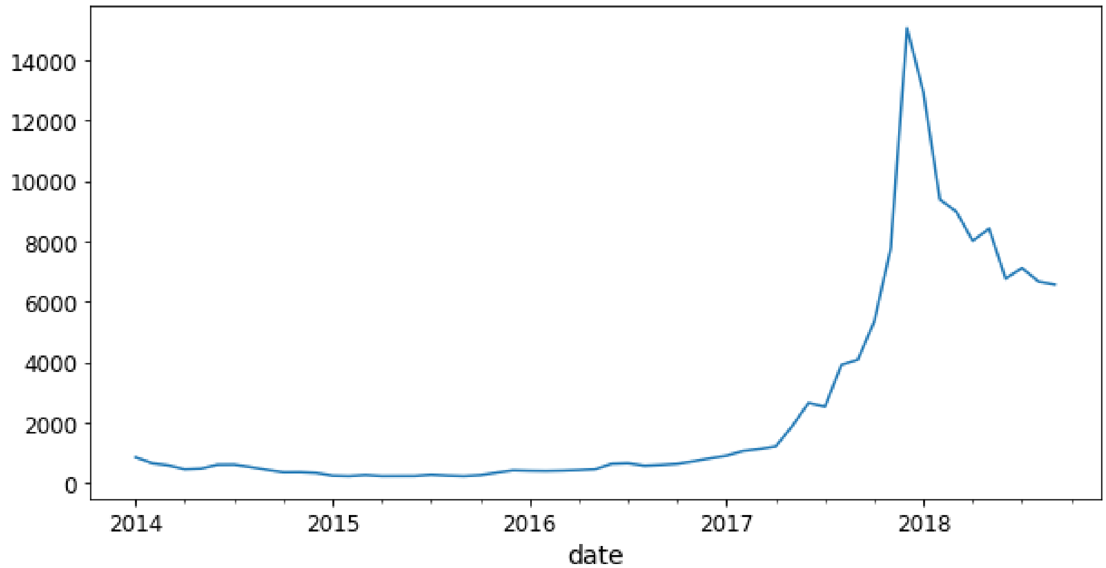
	date	bprice	goldprice	sp	forex	oil
0	2014/01/01	770.44	NaN	NaN	NaN	NaN
1	2014/01/02	808.05	1219.75	1831.98	1.3670	95.14
2	2014/01/03	830.02	1232.25	1831.37	1.3606	93.66
3	2014/01/04	858.98	NaN	NaN	NaN	NaN
4	2014/01/05	940.10	NaN	NaN	NaN	NaN

In [23]:
```
final_date[['goldprice', 'sp', 'forex', 'oil']]=final_date[['goldprice', 'sp', 'forex',
final_date = final_date.iloc[1:,:]
final_date['date']=pd.to_datetime(final_date['date'],format='%Y/%m/%d')
matplotlib.rcParams['axes.labelsize'] = 14
matplotlib.rcParams['xtick.labelsize'] = 12
matplotlib.rcParams['ytick.labelsize'] = 12
matplotlib.rcParams['text.color'] = 'k'
final_date.set_index('date',inplace=True)
```

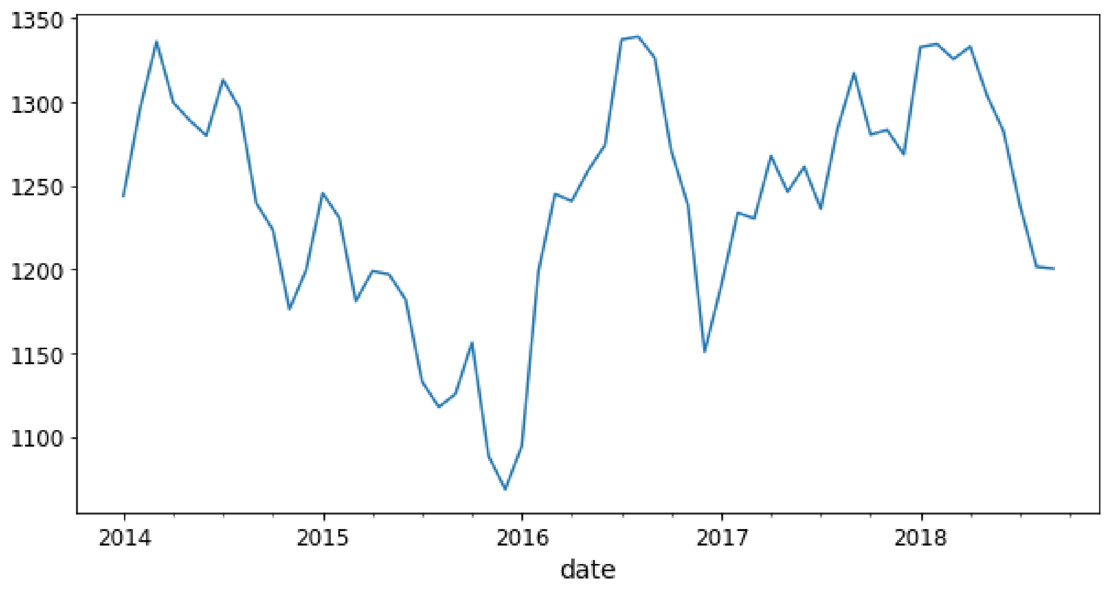
0.0.1 q4

In [24]:
```
for i in range(0,final_date.shape[1]):
    print('\033[1m ',final_date.columns[i],' \033[0;0m')
    y = final_date.iloc[0:,i].resample('MS').mean()
    y.plot(figsize=(10, 5))
    plt.show()
```

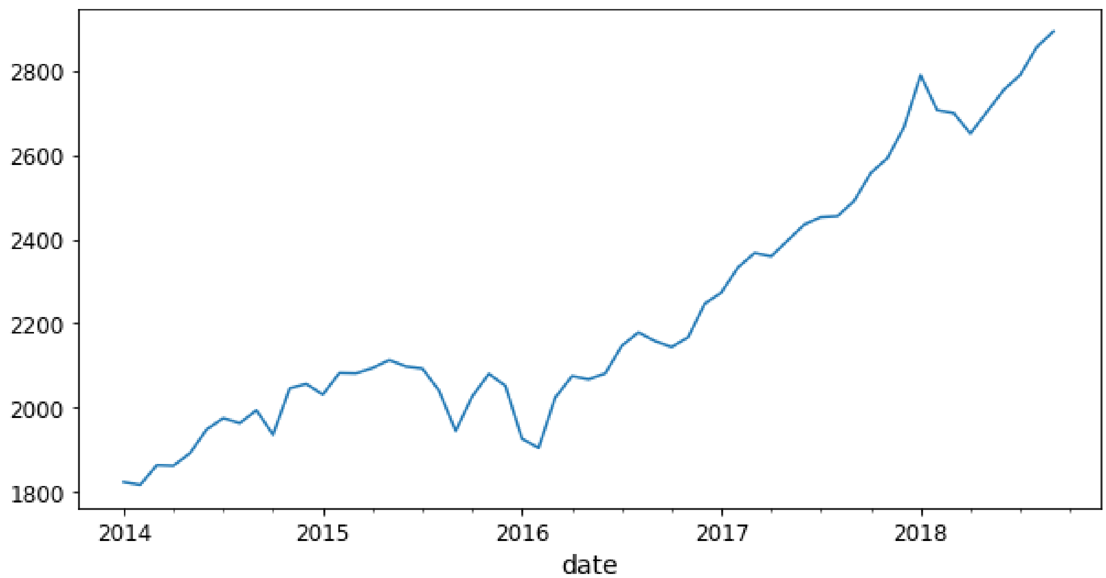
bprice



goldprice

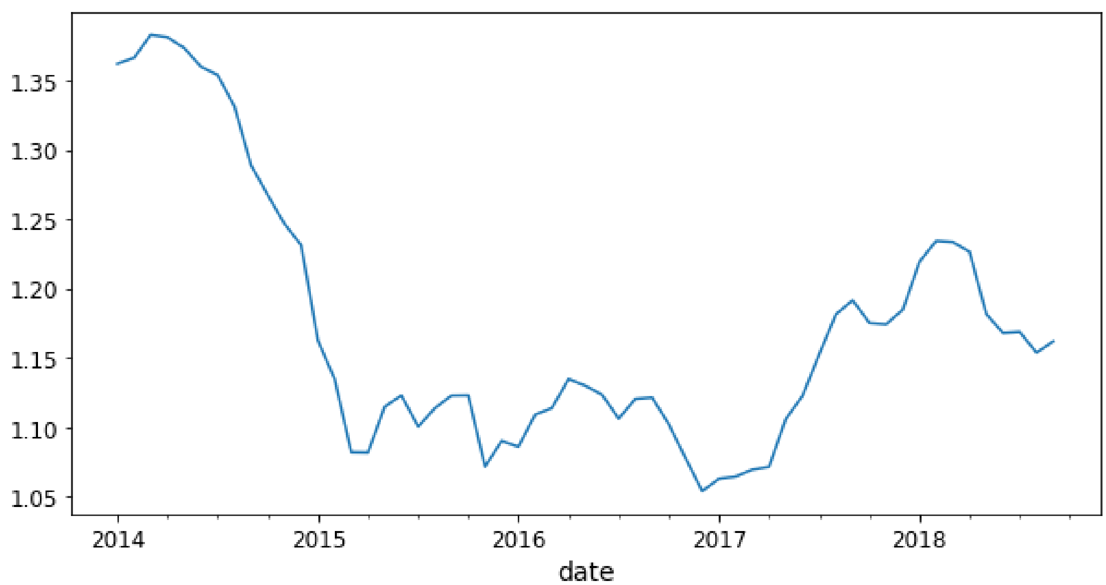


sp



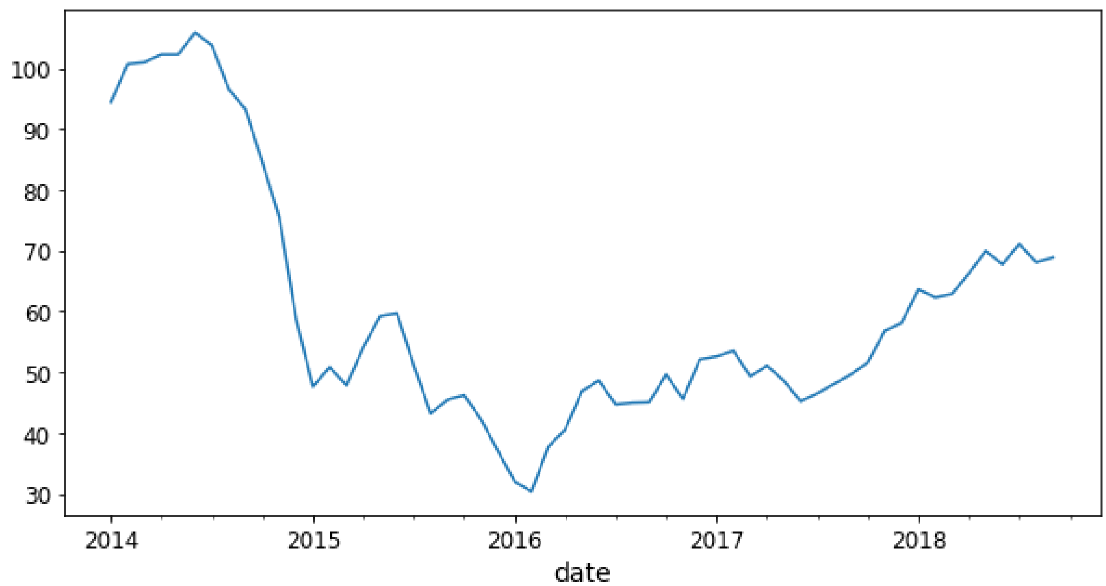
2800
2600
2400
2200
2000
1800
2014
2015
2016
2017
2018
date

`forex`



1.35
1.30
1.25
1.20
1.15
1.10
1.05
2014
2015
2016
2017
2018
date

`oil`



0.0.2 q5

```
In [25]: X = final_date.iloc[:,1:]
         y = final_date.iloc[:,0]
         model = sm.OLS(np.array(y).reshape(-1,1),np.array(X).reshape(-1,4))
         results = model.fit()
         print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.714
Model:                            OLS   Adj. R-squared:                  0.713
Method:                 Least Squares   F-statistic:                     1071.
Date:                Wed, 03 Apr 2019   Prob (F-statistic):               0.00
Time:                        16:15:39   Log-Likelihood:                -15777.
No. Observations:                1723   AIC:                         3.156e+04
Df Residuals:                    1719   BIC:                         3.158e+04
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1           -10.4403      0.964    -10.835      0.000     -12.330      -8.550
x2             9.2304      0.205     45.112      0.000       8.829       9.632
x3         -7913.4446   1023.946     -7.728      0.000   -9921.755   -5905.134
x4            68.9103      4.575     15.062      0.000      59.937      77.884
==============================================================================
```

```
Omnibus:                      700.905   Durbin-Watson:                   0.017
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3576.148
Skew:                           1.870   Prob(JB):                         0.00
Kurtosis:                       8.986   Cond. No.                     4.75e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.75e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

The high R^2 shows us that the other variables are explaining about 70% of variation which is highly improbable because, the time series data are always highly correlated.

0.0.3 q6

```
In [26]: from statsmodels.tsa.stattools import kpss
         def rep_kpss(series,alpha=0.05,diff_max=6):
             diff = 0
             for i in range(0,diff_max):
                 pval = kpss(series,regression='c')[1]
                 if(pval>=alpha):
                     return(diff,0,pval,'level stationary')
                 pval = kpss(series,regression='ct')[1]
                 if(pval>=alpha):
                     return(diff,1,pval,'trend stationary')
                 diff +=1
                 series=series.diff().dropna()
                 #return(0)
         for i in final_date.columns:
             print(i)
             print(rep_kpss(final_date[i]))
```

```
bprice
(1, 0, 0.1, 'level stationary')
goldprice
(1, 0, 0.1, 'level stationary')
sp
(1, 0, 0.1, 'level stationary')
forex
(1, 0, 0.07134131156173776, 'level stationary')
oil
(1, 1, 0.1, 'trend stationary')
```

```
In [27]: kpss(final_date.iloc[:,4],regression='ct')
```

```
Out[27]: (1.38849613143664,
          0.01,
          25,
          {'10%': 0.119, '5%': 0.146, '2.5%': 0.176, '1%': 0.216})
```

0.0.4 q7

```
In [58]: y
```

```
Out[58]: date
         2014-01-03     830.02
         2014-01-04     858.98
         2014-01-05     940.10
         2014-01-06     951.39
         2014-01-07     810.58
         2014-01-08     859.95
         2014-01-09     860.89
         2014-01-10     884.67
         2014-01-11     930.90
         2014-01-12     873.26
         2014-01-13     857.96
         2014-01-14     851.83
         2014-01-15     874.71
         2014-01-16     847.37
         2014-01-17     828.22
         2014-01-18     843.76
         2014-01-19     878.68
         2014-01-20     871.05
         2014-01-21     874.29
         2014-01-22     863.95
         2014-01-23     854.35
         2014-01-24     825.12
         2014-01-25     861.85
         2014-01-26     880.15
         2014-01-27     814.53
         2014-01-28     833.94
         2014-01-29     837.51
         2014-01-30     845.85
         2014-01-31     848.29
         2014-02-01     853.02
                         ...
         2018-08-22    6357.59
         2018-08-23    6525.61
         2018-08-24    6692.62
         2018-08-25    6732.50
         2018-08-26    6707.63
         2018-08-27    6907.66
         2018-08-28    7076.74
```

```
2018-08-29    7035.81
2018-08-30    6982.40
2018-08-31    7013.97
2018-09-01    7192.30
2018-09-02    7295.13
2018-09-03    7261.49
2018-09-04    7358.50
2018-09-05    6687.01
2018-09-06    6498.62
2018-09-07    6396.27
2018-09-08    6183.38
2018-09-09    6238.54
2018-09-10    6305.57
2018-09-11    6282.92
2018-09-12    6328.93
2018-09-13    6486.62
2018-09-14    6492.37
2018-09-15    6515.90
2018-09-16    6497.37
2018-09-17    6251.16
2018-09-18    6334.20
2018-09-19    6388.98
2018-09-20    6491.64
Name: bprice, Length: 1722, dtype: float64
```

```
In [28]: X = final_date.iloc[:,1:].diff().dropna()
         y = final_date.iloc[1:,0]
         model_diff = sm.OLS(np.array(y).reshape(-1,1),np.array(X).reshape(-1,4))
         results_diff = model_diff.fit()
         print(results_diff.summary())
```

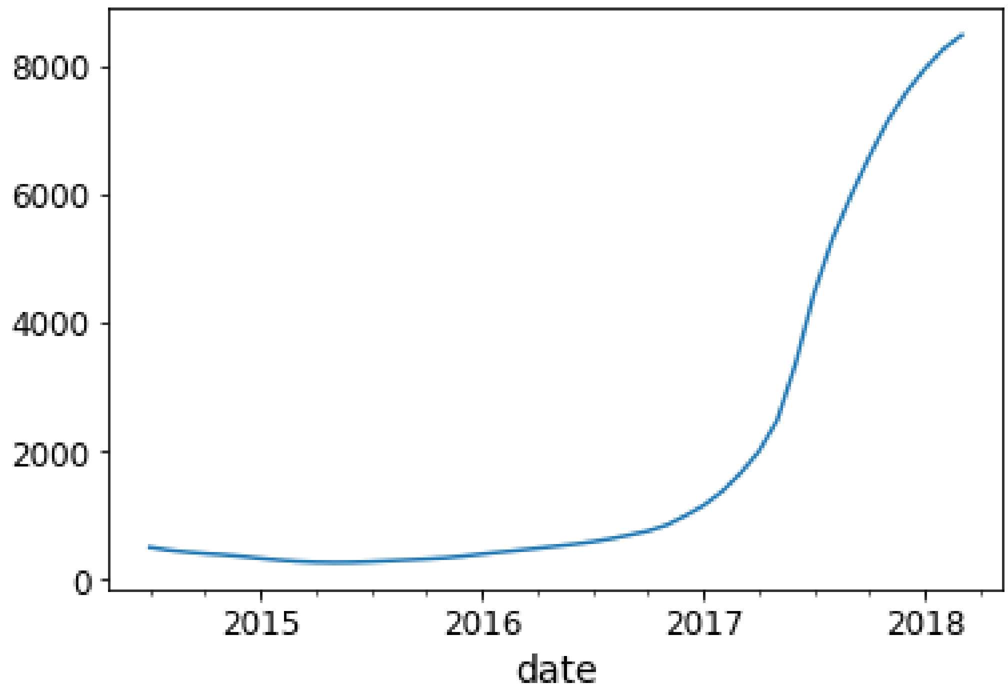
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.003
Model:                            OLS   Adj. R-squared:                  0.001
Method:                 Least Squares   F-statistic:                     1.376
Date:                Wed, 03 Apr 2019   Prob (F-statistic):              0.240
Time:                        16:15:40   Log-Likelihood:                -16842.
No. Observations:                1722   AIC:                         3.369e+04
Df Residuals:                    1718   BIC:                         3.371e+04
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1             1.0284     14.541      0.071      0.944     -27.491      29.548
x2            16.8869      8.238      2.050      0.041       0.730      33.044
x3          9640.7971   2.23e+04      0.433      0.665   -3.41e+04    5.33e+04
```

```
x4            63.6283    115.804      0.549      0.583    -163.503     290.759
==============================================================================
Omnibus:                      639.279   Durbin-Watson:                   0.009
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1909.172
Skew:                           1.942   Prob(JB):                         0.00
Kurtosis:                       6.395   Cond. No.                     2.81e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.81e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

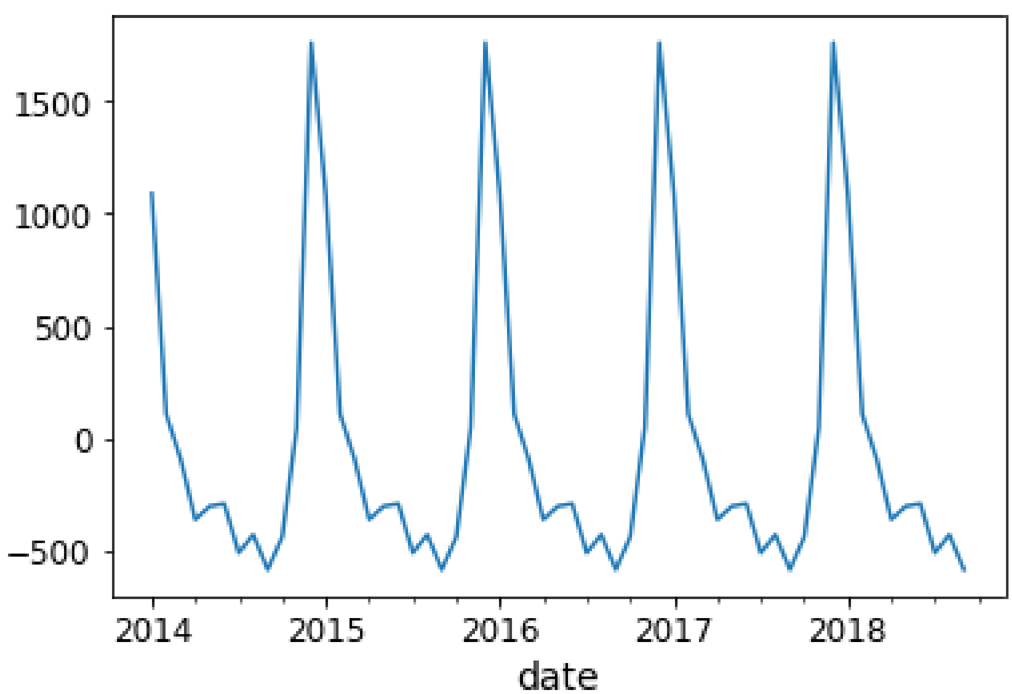
0.0.5 The R^2 suggests that there is no relationship between the series

```
In [29]: for i in range(0,final_date.shape[1]):
             print('\033[1m',final_date.columns[i],'\033[0;0m')
             y1 = final_date.iloc[0:,i].resample('MS').mean()
             decomposition = sm.tsa.seasonal_decompose(y1, model='additive')
             print(' \033[1m Trend Plot \033[0;0m')
             decomposition.trend.plot()
             plt.show()
             print(' \033[1m seasonal Plot \033[0;0m')
             decomposition.seasonal.plot()
             plt.show()
             print(' \033[1m resid Plot \033[0;0m')
             decomposition.resid.plot()
             plt.show()
             print(' \033[1m observed Plot \033[0;0m')
             decomposition.observed.plot()
             plt.show()
```

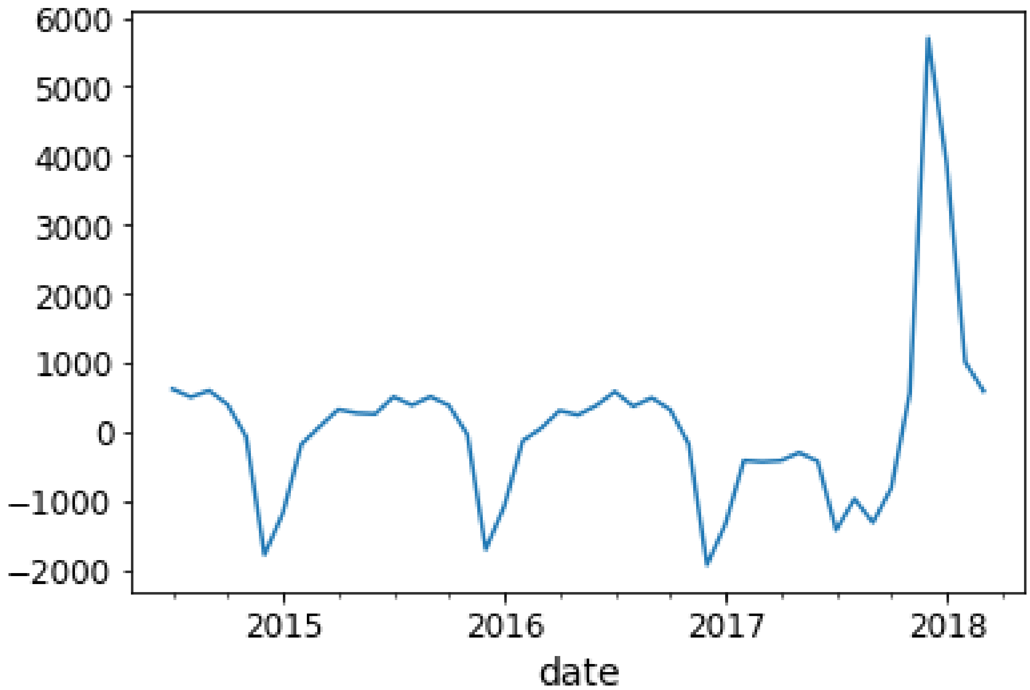
```
 bprice
  Trend Plot
```



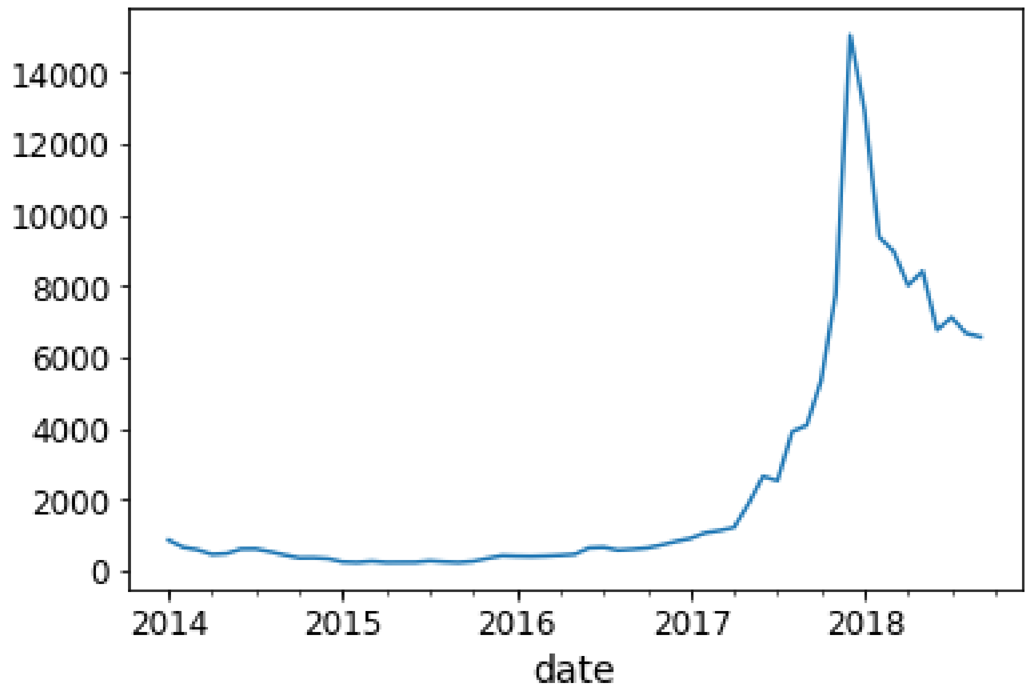
seasonal Plot



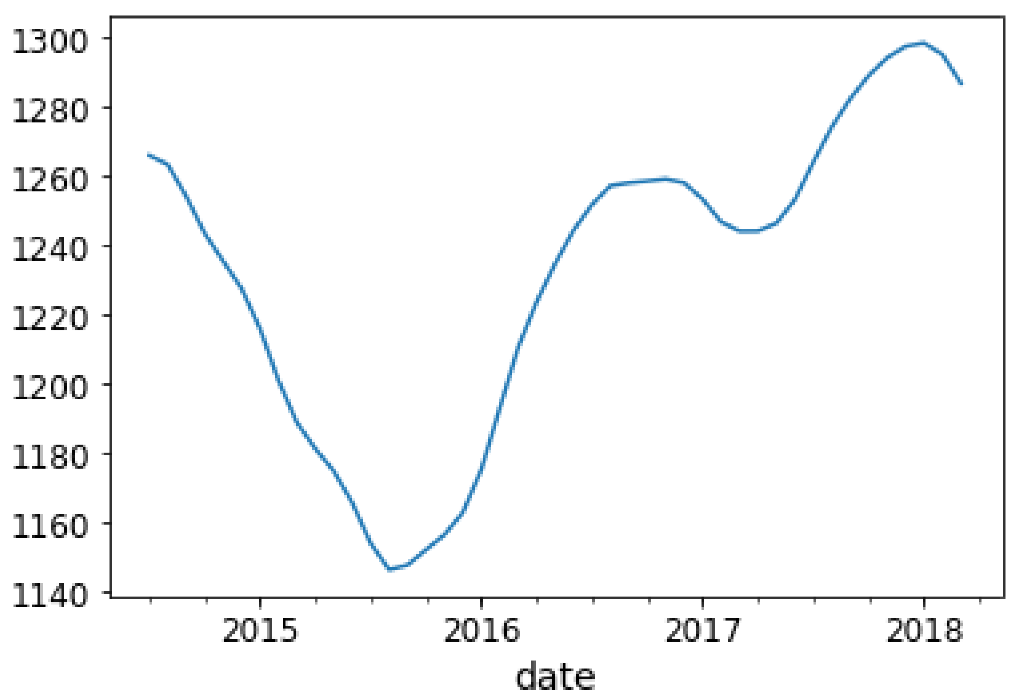
resid Plot



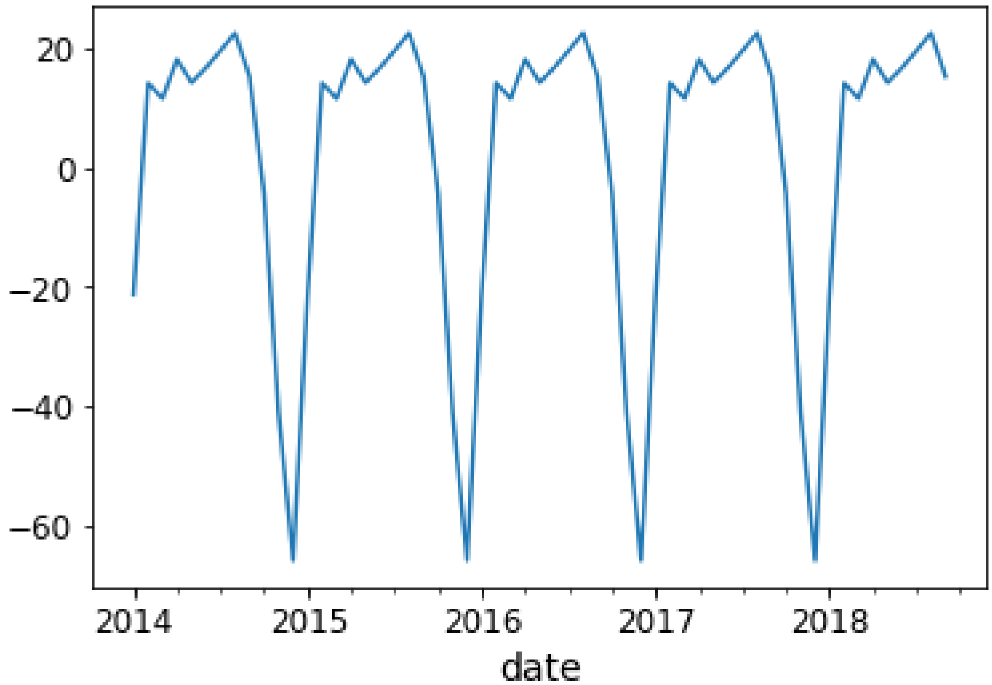
observed Plot



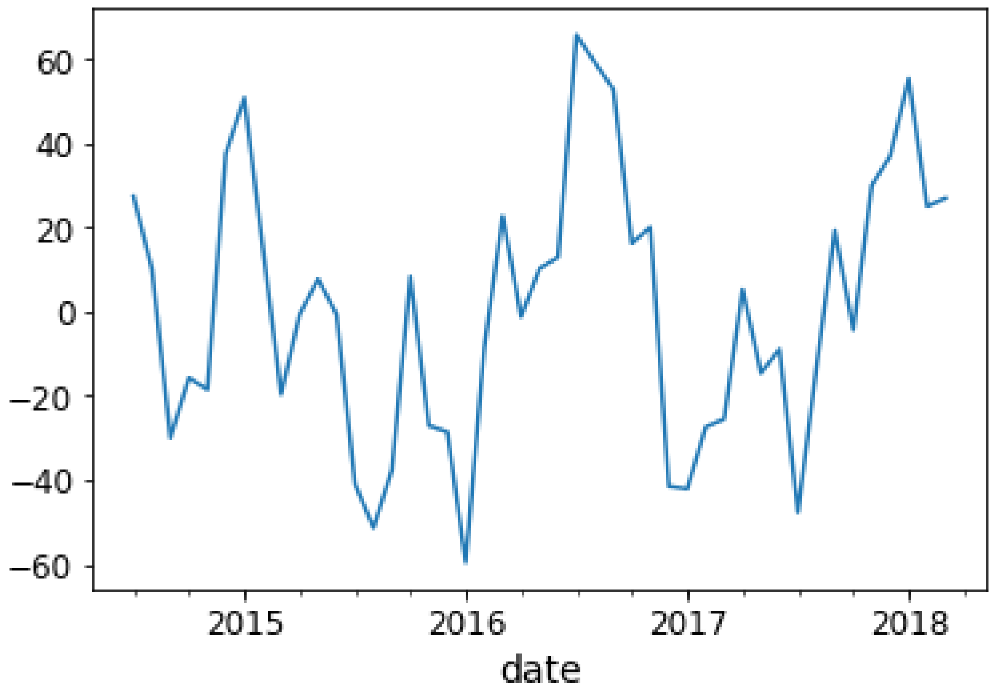
goldprice
Trend Plot



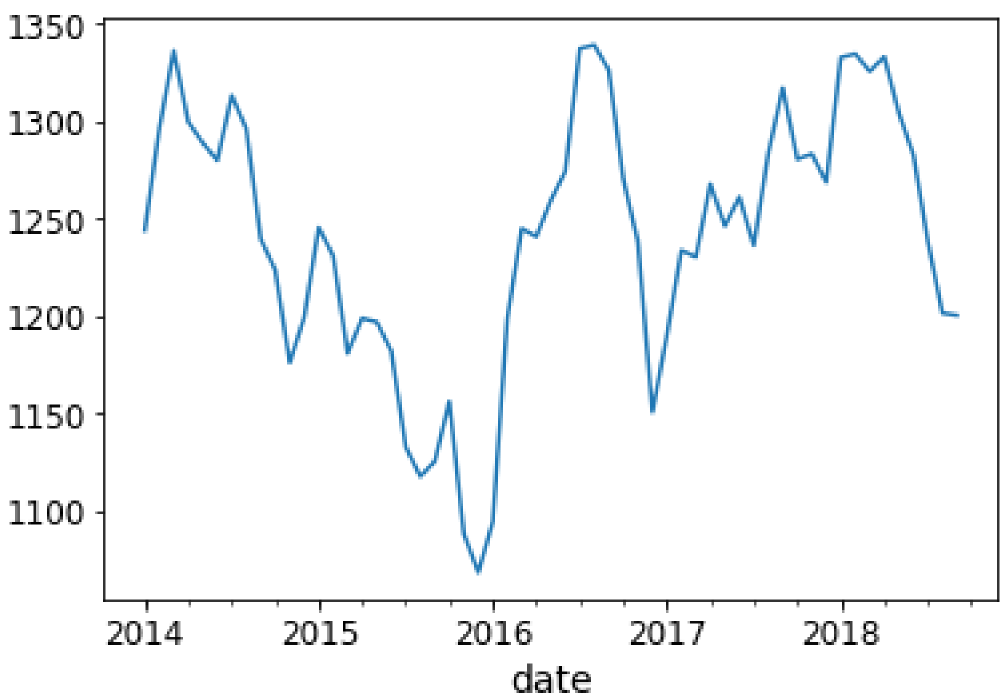
seasonal Plot



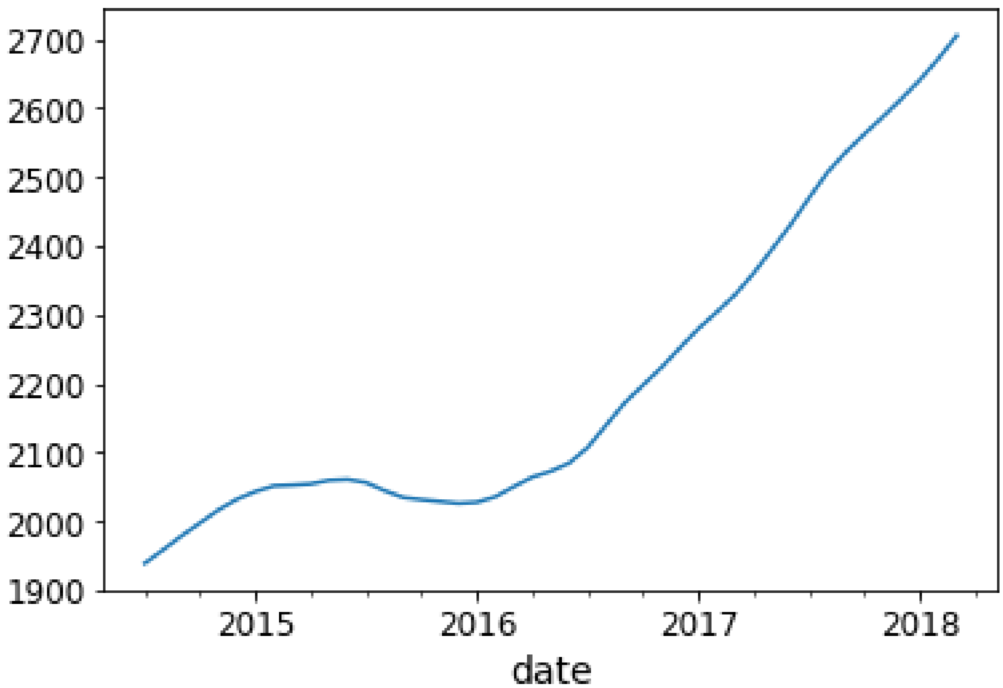
resid Plot



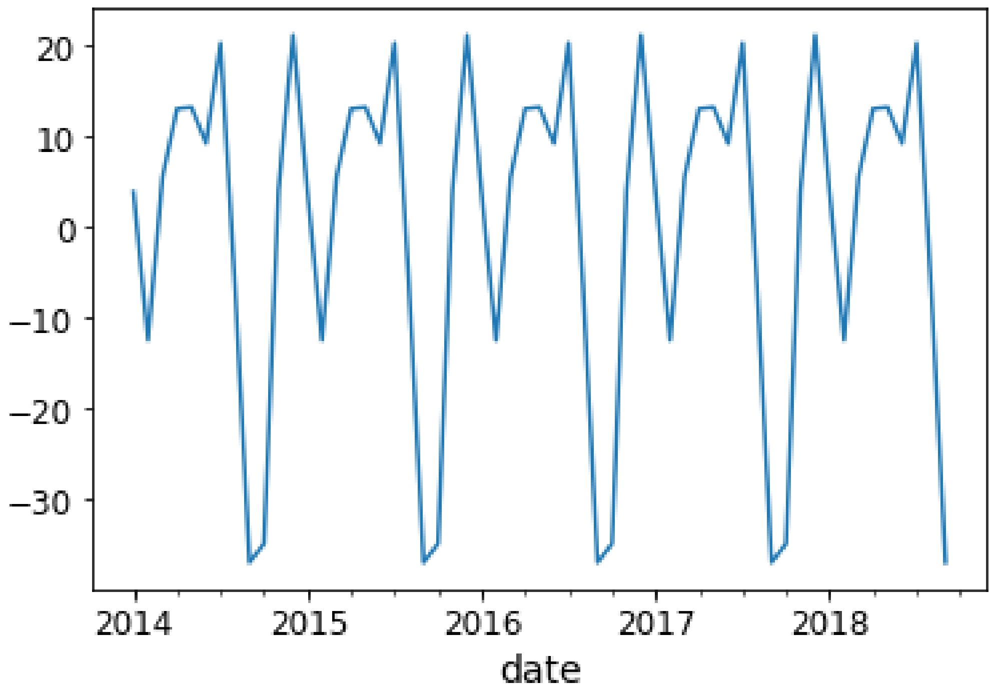
observed Plot



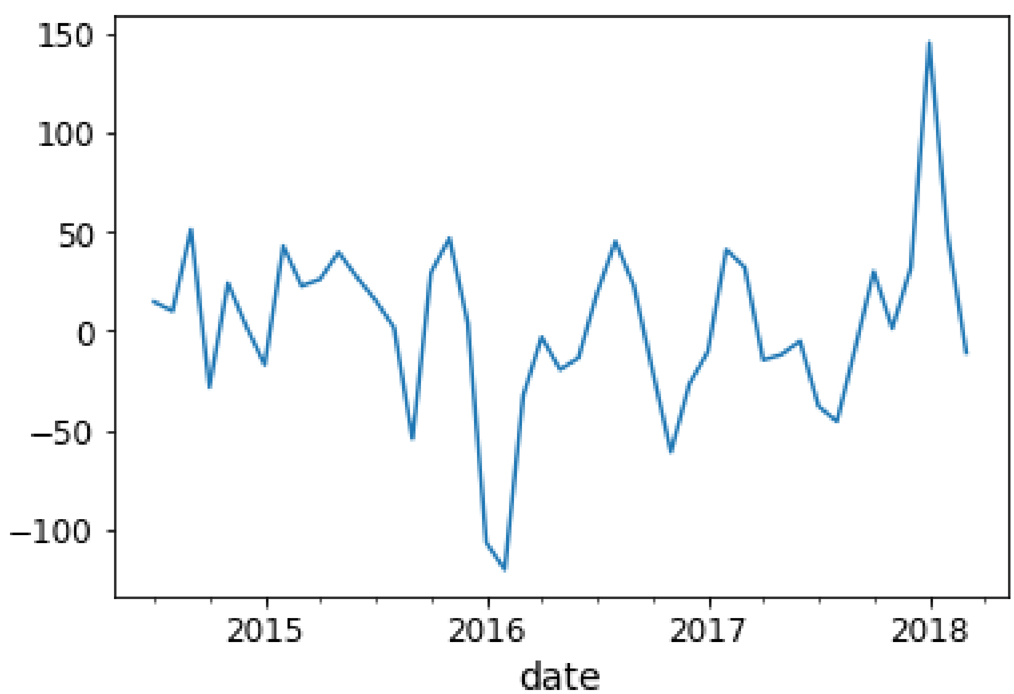
sp
Trend Plot



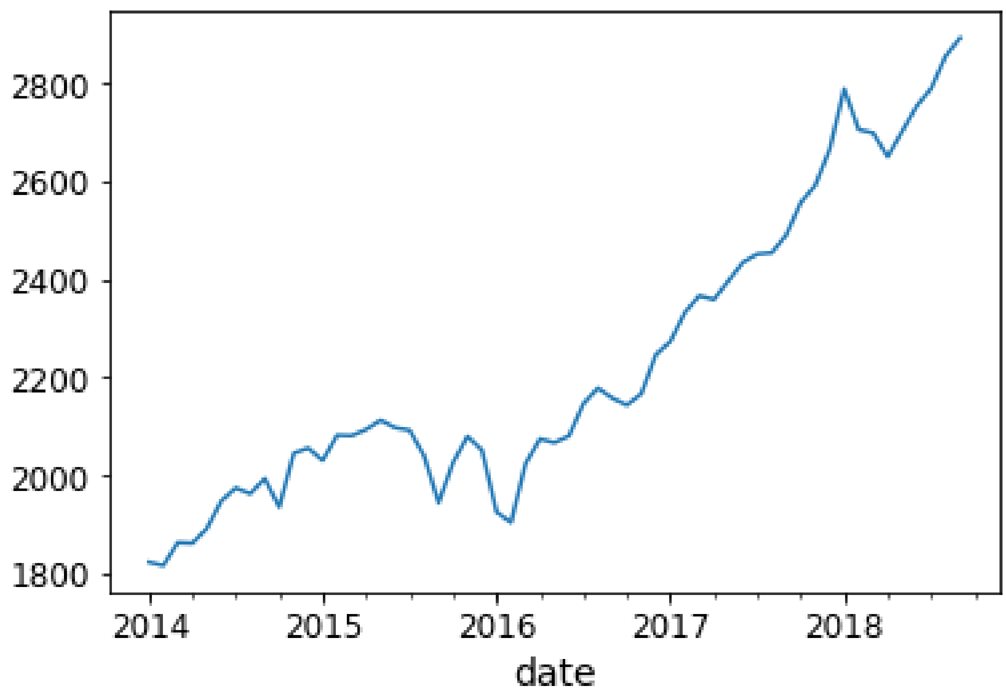
seasonal Plot


20
10
0
−10
−20
−30
2014
2015
2016
2017
2018
date

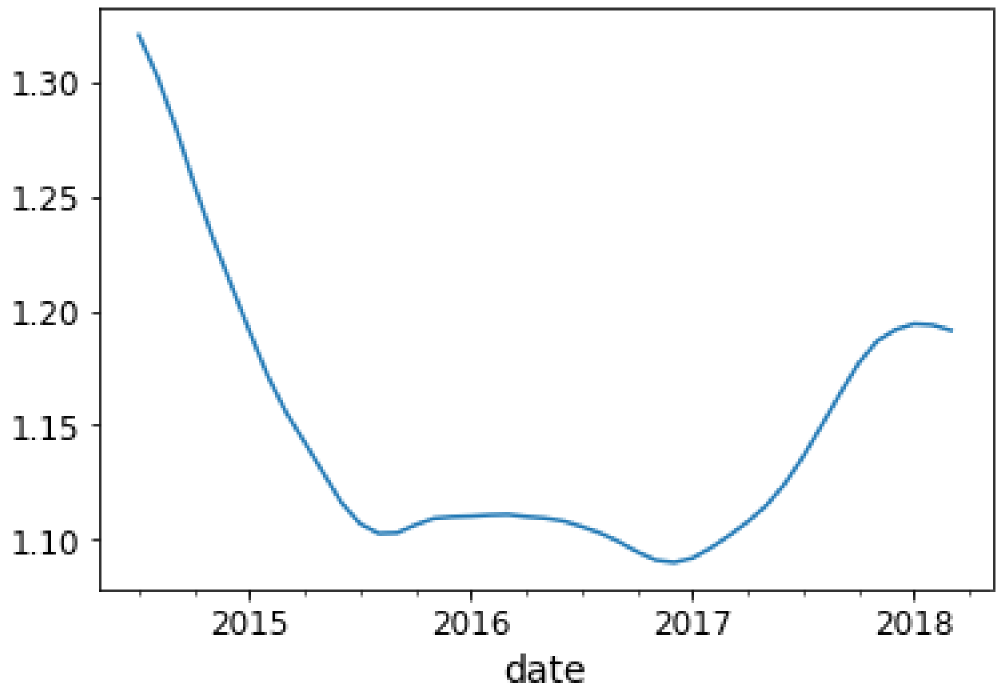
resid Plot


150
100
50
0
−50
−100
2015
2016
2017
2018
date

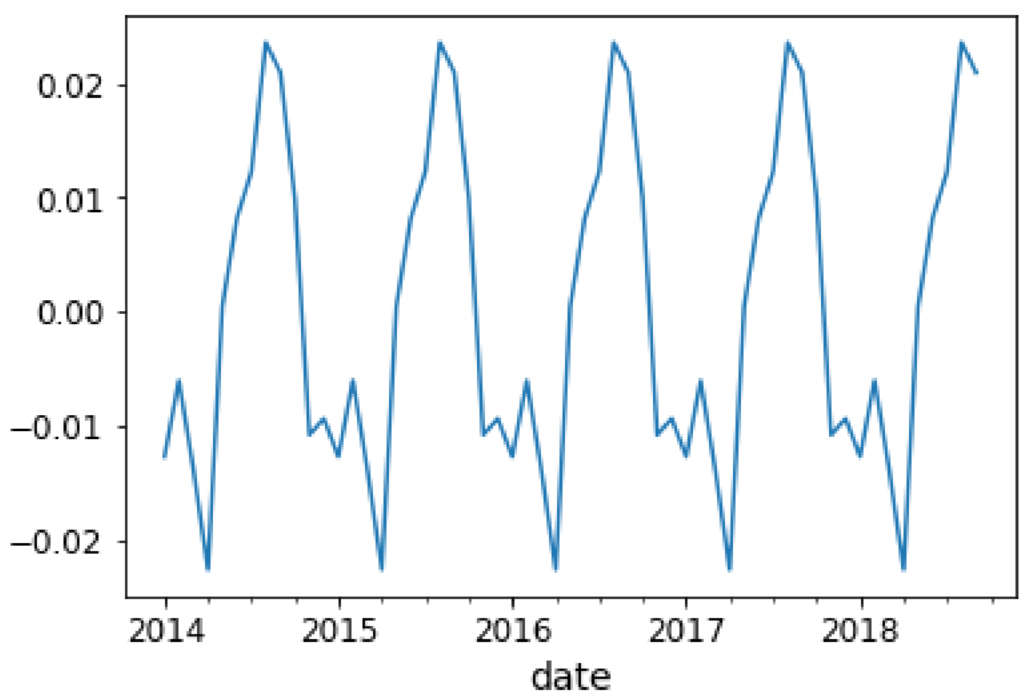
observed Plot



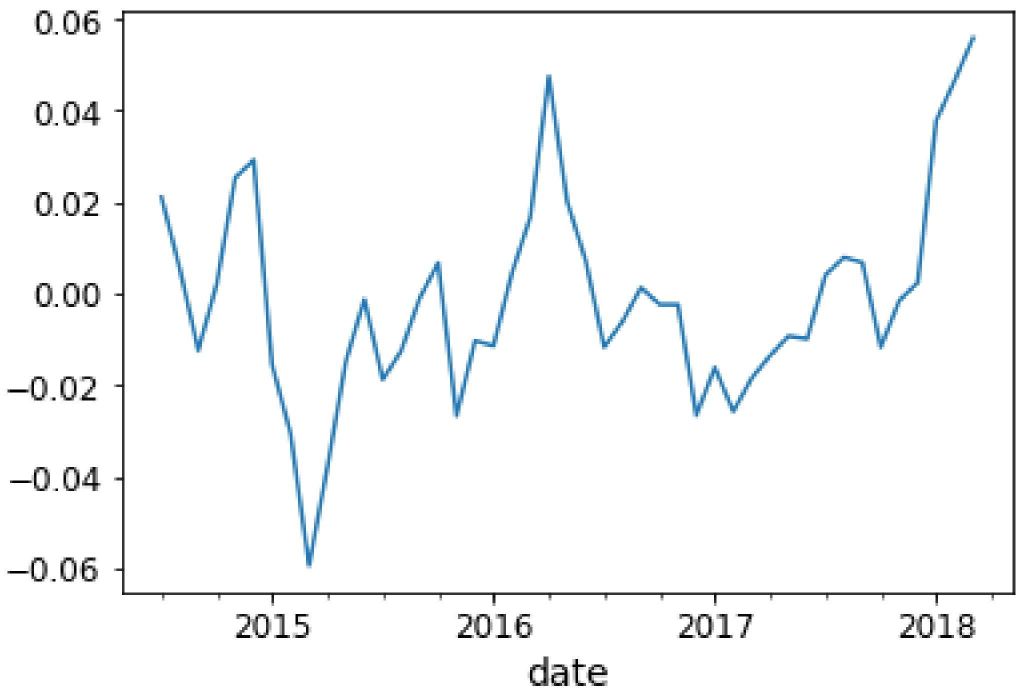
forex
Trend Plot



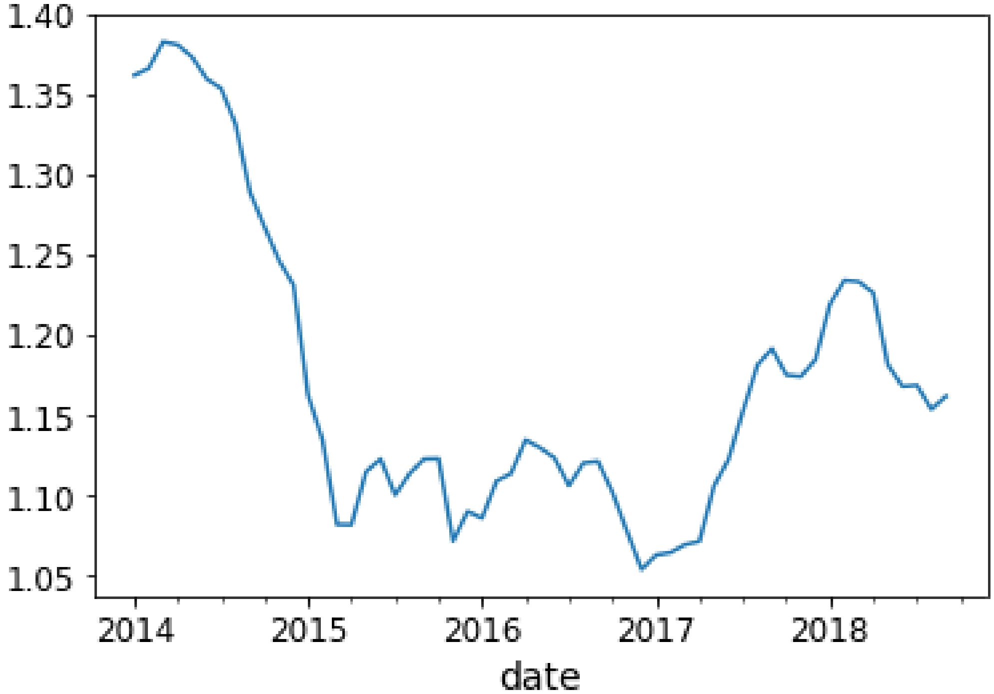
seasonal Plot



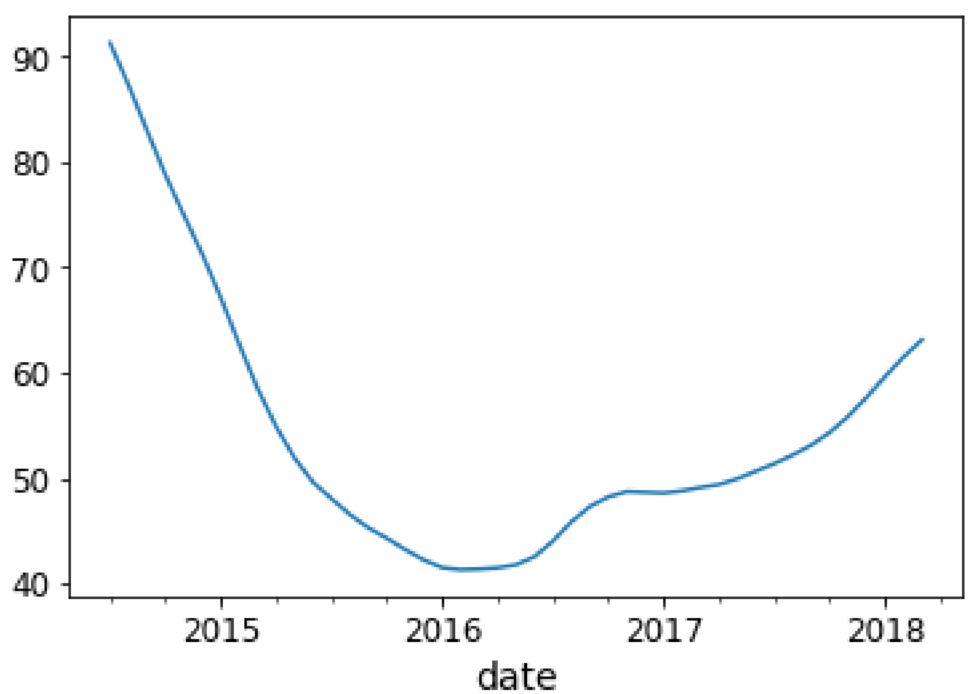
resid Plot



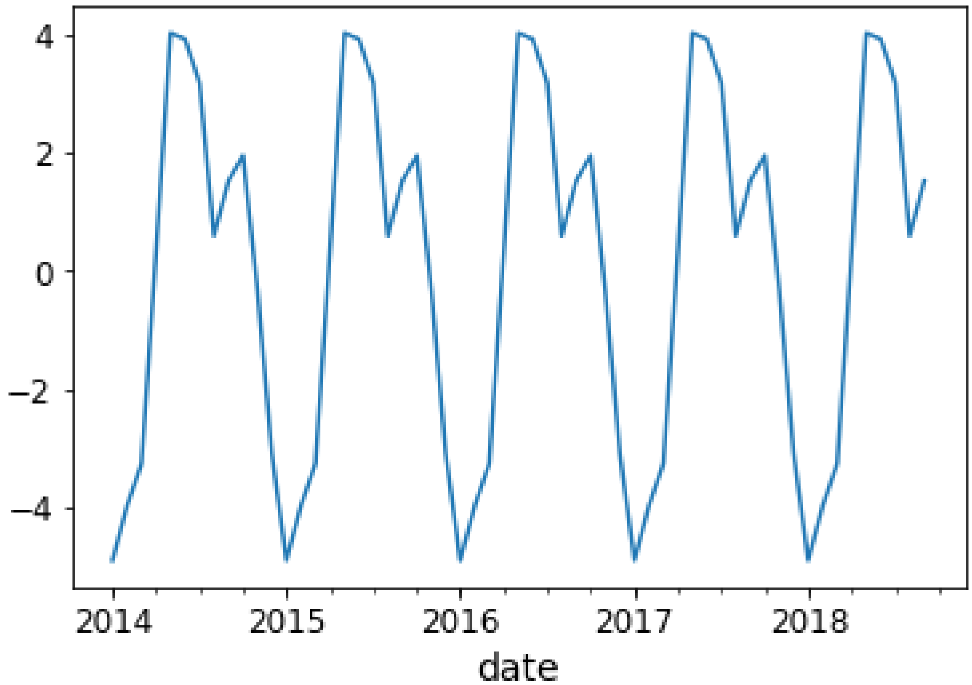
observed Plot



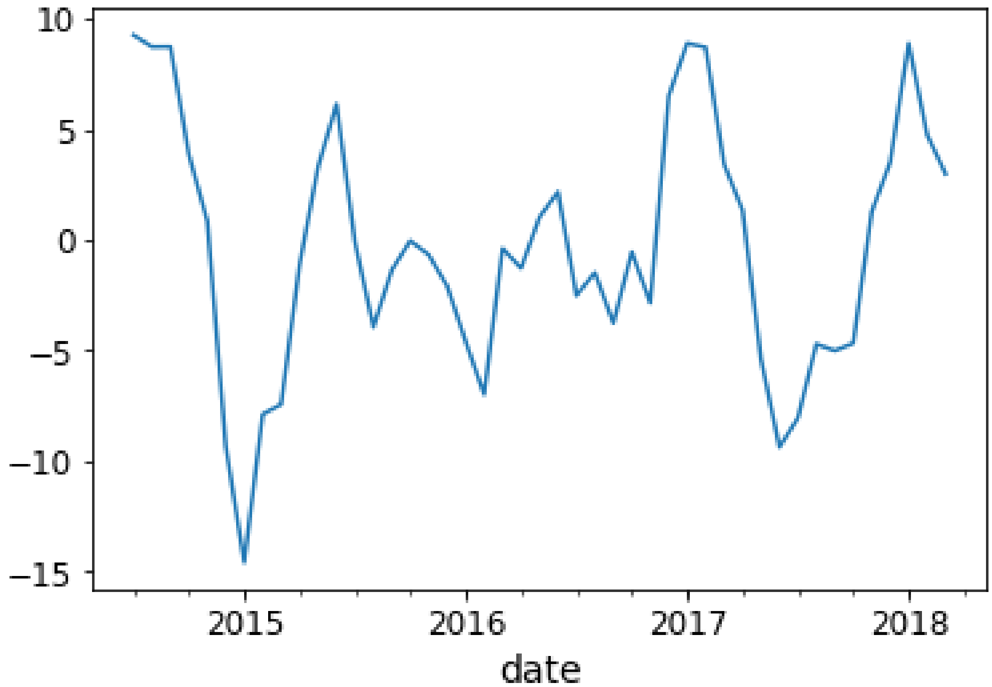
oil
Trend Plot



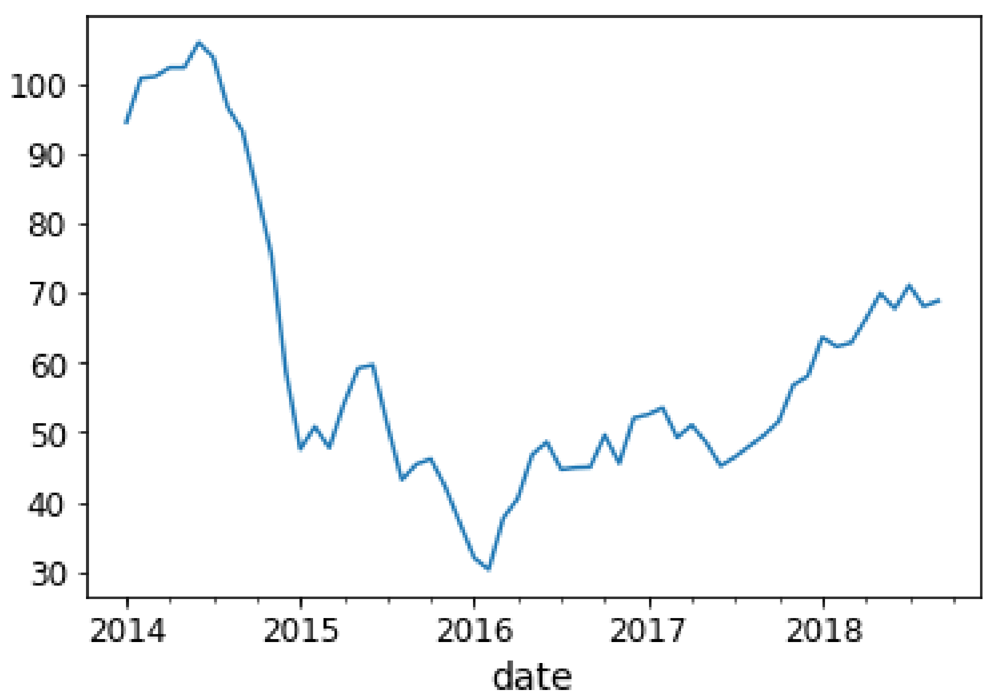
seasonal Plot



resid Plot



observed Plot



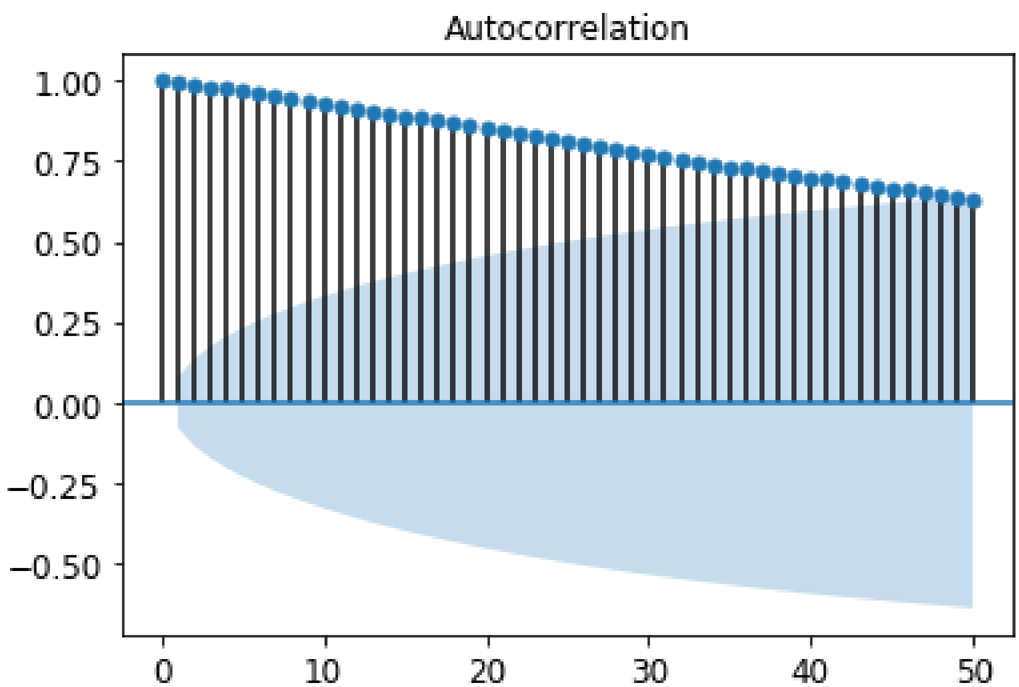
0.0.6 q8

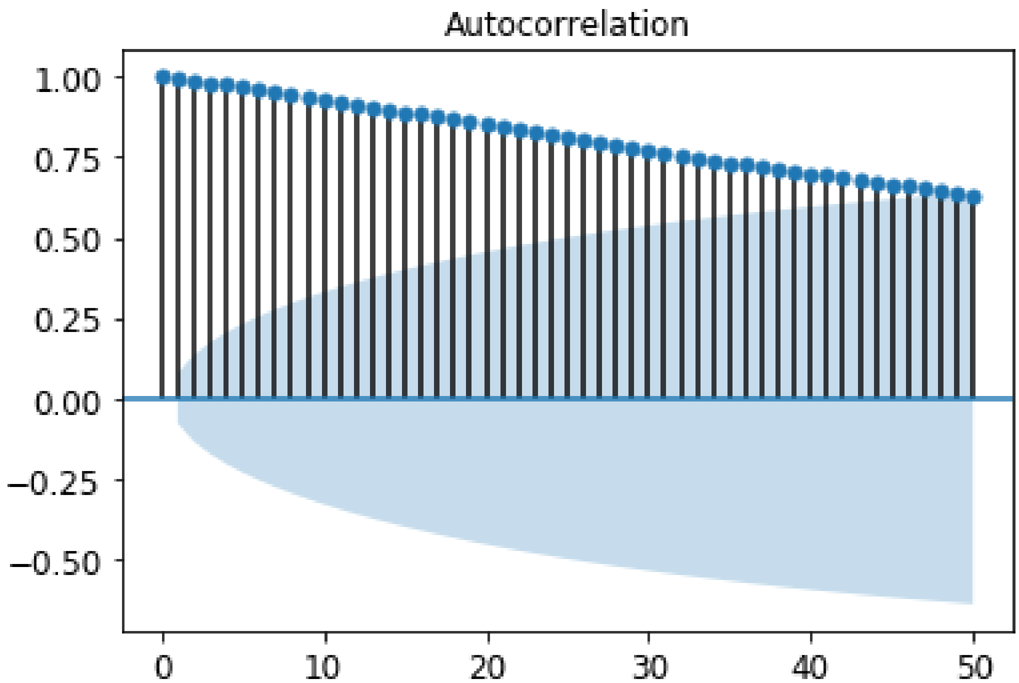
```
In [30]: final_date_2017 = final_date['2017':]
```

0.0.7 q9

```
In [31]: from statsmodels.graphics.tsaplots import plot_acf
         bprice_1d = final_date_2017['bprice'].diff().dropna()
         final_date_2017 = final_date['2017':]
         plot_acf(final_date_2017['bprice'],lags=50)
         # suggests p 7:10
```

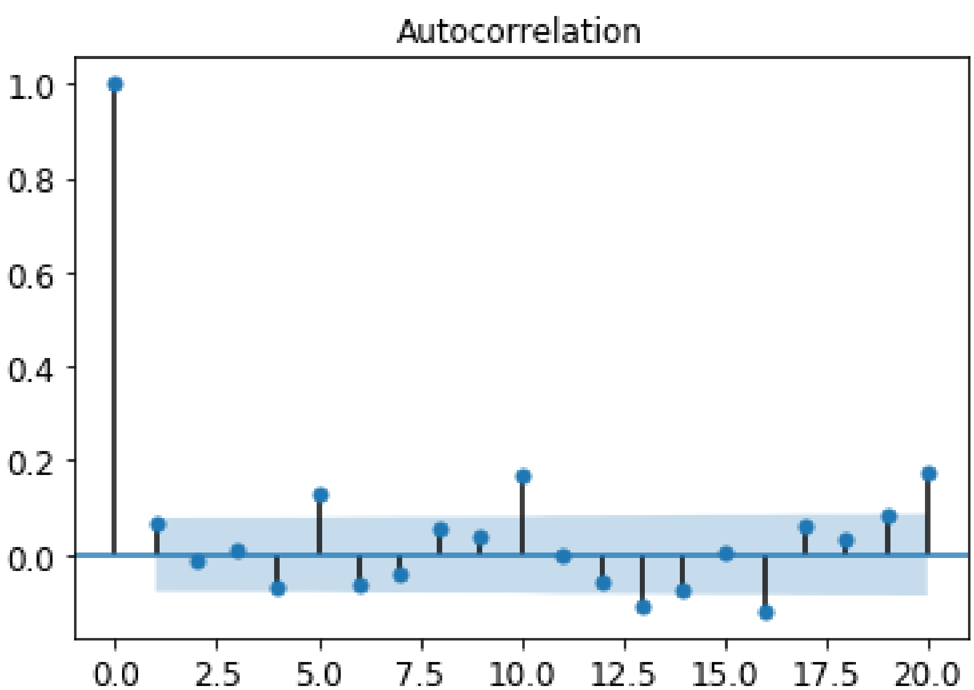
```
Out[31]:
```

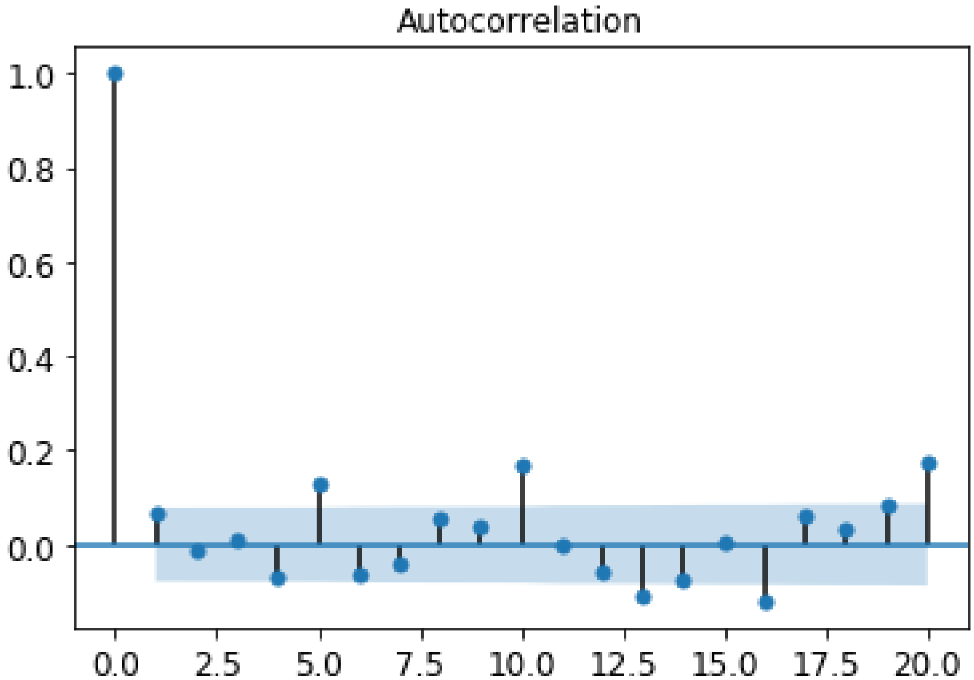




In [32]: `plot_acf(bprice_1d,lags=20)`

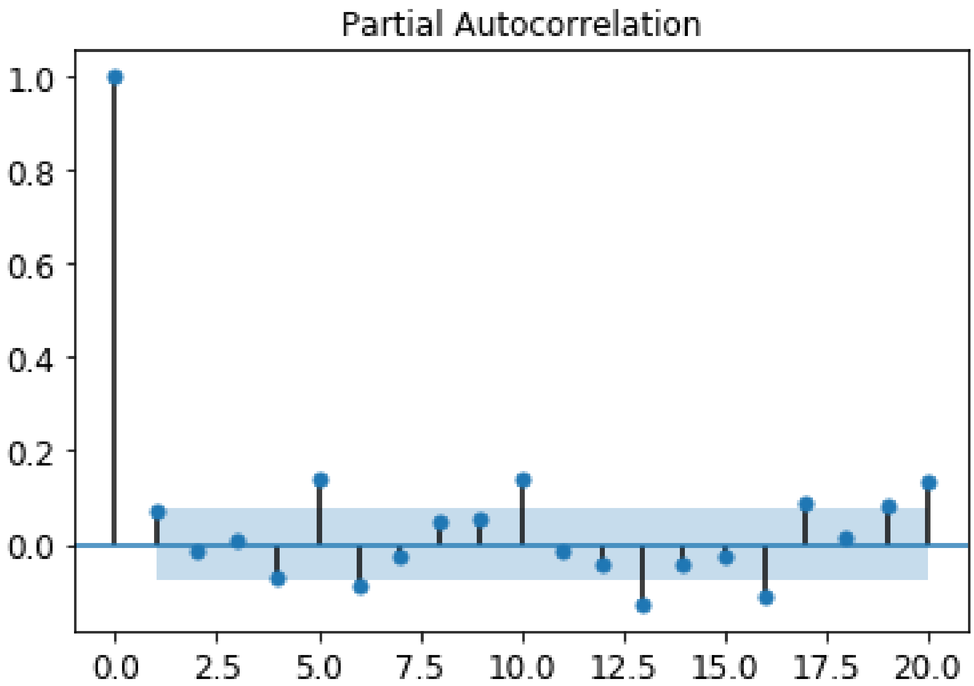
Out[32]:

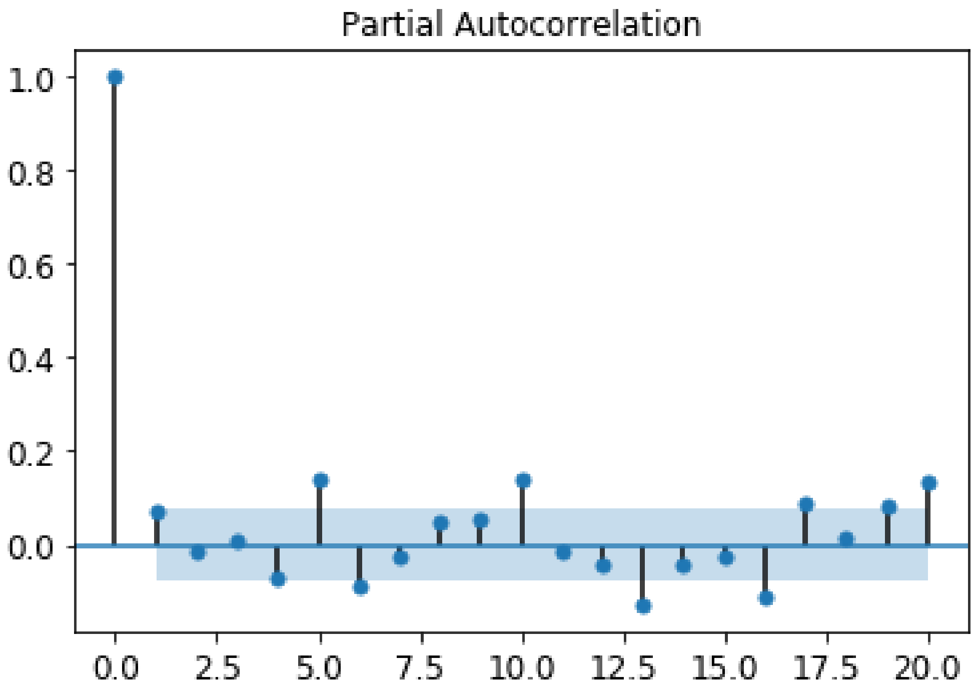




In [33]:
```
from statsmodels.graphics.tsaplots import plot_pacf
plot_pacf(bprice_1d,lags=20)
# q=2
```

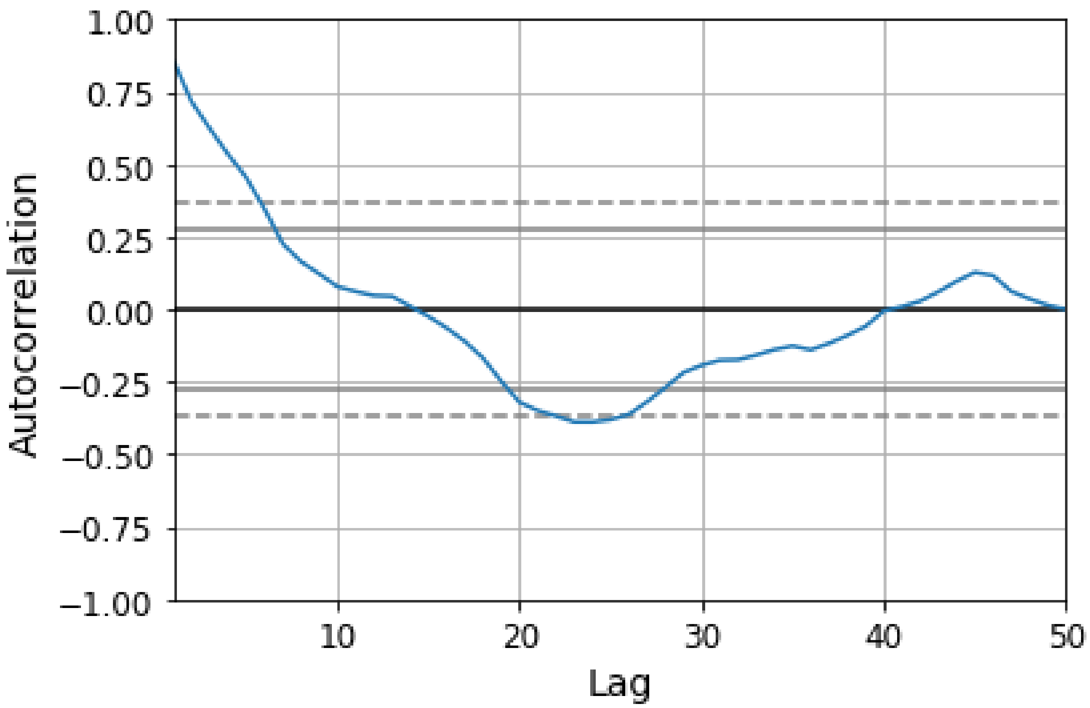
Out[33]:






```
In [34]: from pandas.tools.plotting import autocorrelation_plot
         autocorrelation_plot(final_date_2017['bprice'][0:50])
```

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x1bf4efdda90>



In [35]: bprice = final_date_2017['bprice']

0.0.8 q10

good model for ARIMA. from acf and pacf, p is around 6-10 and q is 2.

In [36]:
```
# https://www.statsmodels.org/stable/tsa.html
# https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima_model.ARIMA.predic
# https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima_process.ArmaProces
# https://www.statsmodels.org/stable/tsa.html#autogressive-moving-average-processes-arm
# https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima_model.ARIMA.html#s
# https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima_model.ARIMA.fit.ht
# https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima_model.ARIMAResults
# https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima_model.ARIMAResults
bprice = final_date_2017['bprice']
from statsmodels.tsa.arima_model import ARIMA
model=ARIMA(bprice,order=(15,1,2))
r=model.fit()
r.aic
```

Out[36]: 9285.09521462024

```
In [37]: from statsmodels.tsa.arima_model import ARIMA
         maxa = 10
         outp = np.zeros(((maxa+1)**2,3))
         count = 0
         for i in range(6,maxa+1):
             print((round(i*(maxa+1)/(maxa+1)**2*100,2)),'%...')
             for j in range(1,4):
                 try:
                     mod = ARIMA(bprice,order=(i,1,j))
                     results = mod.fit()
                     outp[count,:]= np.array([[i],[j],[results.aic]]).T
                     count+=1
                 except:
                     continue
         outp = pd.DataFrame(outp)
         outp.columns = ['p','q','AIC']
         outp = outp.loc[(outp!=0).any(axis=1)].sort_values('AIC')
         print("best p and q are:",outp.iloc[0,:])
```

```
54.55 %...
63.64 %...
72.73 %...
81.82 %...
90.91 %...
best p and q are: p         9.000000
q         2.000000
AIC    9256.501919
Name: 10, dtype: float64
```

0.0.9 best model

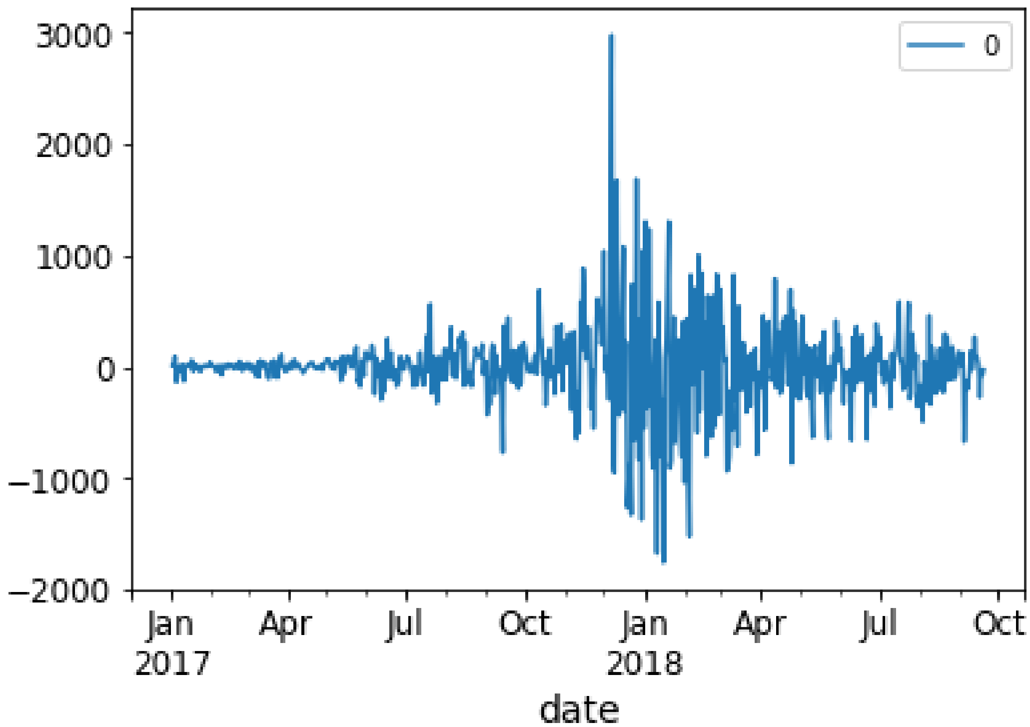
```
In [38]: mod = ARIMA(bprice,order=(9,1,2))
         results = mod.fit()
         print(results.summary())
```

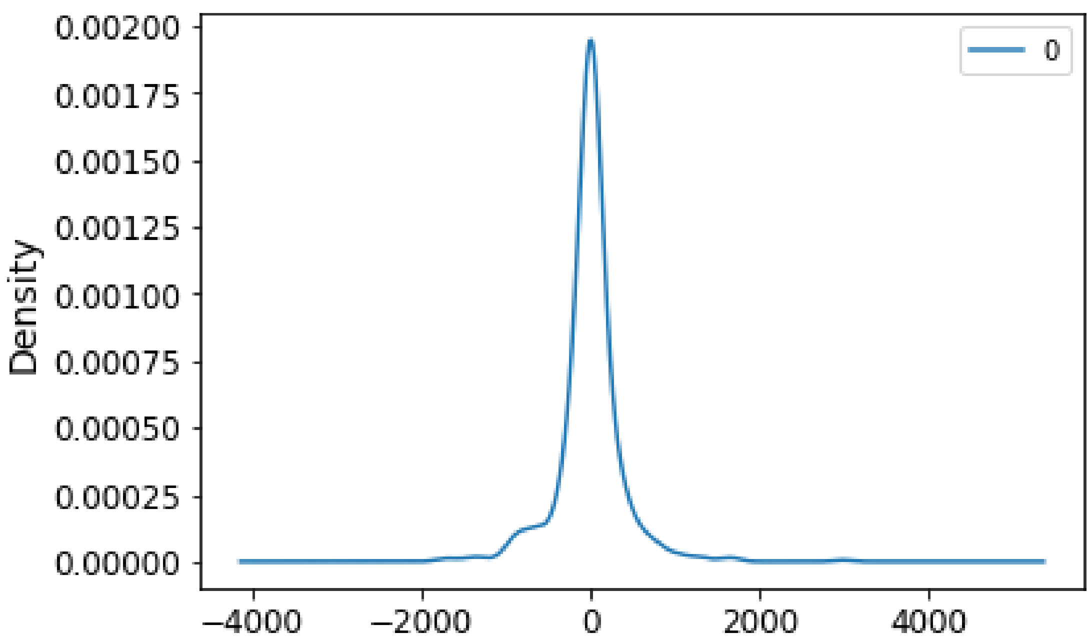
```
                             ARIMA Model Results
==============================================================================
Dep. Variable:               D.bprice   No. Observations:                  627
Model:                 ARIMA(9, 1, 2)   Log Likelihood               -4615.251
Method:                       css-mle   S.D. of innovations            379.979
Date:                Wed, 03 Apr 2019   AIC                           9256.502
Time:                        16:18:07   BIC                           9314.234
Sample:                    01-02-2017   HQIC                          9278.931
                         - 09-20-2018
==================================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
```

```
const                8.5911     17.508      0.491      0.624     -25.724      42.906
ar.L1.D.bprice       1.7481      0.042     41.257      0.000       1.665       1.831
ar.L2.D.bprice      -1.0639      0.082    -12.992      0.000      -1.224      -0.903
ar.L3.D.bprice       0.1155      0.090      1.279      0.201      -0.062       0.293
ar.L4.D.bprice      -0.1088      0.089     -1.223      0.222      -0.283       0.066
ar.L5.D.bprice       0.2816      0.088      3.191      0.001       0.109       0.455
ar.L6.D.bprice      -0.3886      0.089     -4.374      0.000      -0.563      -0.215
ar.L7.D.bprice       0.2349      0.090      2.608      0.009       0.058       0.411
ar.L8.D.bprice       0.0646      0.081      0.800      0.424      -0.094       0.223
ar.L9.D.bprice      -0.1009      0.040     -2.491      0.013      -0.180      -0.021
ma.L1.D.bprice      -1.7181      0.017   -102.766      0.000      -1.751      -1.685
ma.L2.D.bprice       0.9690      0.017     58.322      0.000       0.936       1.002
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            0.8770           -0.5827j            1.0529           -0.0933
AR.2            0.8770           +0.5827j            1.0529            0.0933
AR.3            1.1967           -0.4046j            1.2632           -0.0519
AR.4            1.1967           +0.4046j            1.2632            0.0519
AR.5            0.2474           -1.2482j            1.2725           -0.2189
AR.6            0.2474           +1.2482j            1.2725            0.2189
AR.7           -0.9674           -0.8594j            1.2940           -0.3844
AR.8           -0.9674           +0.8594j            1.2940            0.3844
AR.9           -2.0668           -0.0000j            2.0668           -0.5000
MA.1            0.8865           -0.4961j            1.0158           -0.0812
MA.2            0.8865           +0.4961j            1.0158            0.0812
-----------------------------------------------------------------------------
```

residual plot

```
In [39]: resid = pd.DataFrame(results.resid)
         resid.plot()
         plt.show()
         resid.plot(kind='kde')
         plt.show()
```


3000
2000
1000
0
−1000
−2000
0
Jan
2017
Apr
Jul
Oct
Jan
2018
Apr
Jul
Oct
date


0.00200
0.00175
0.00150
0.00125
0.00100
0.00075
0.00050
0.00025
0.00000
Density
0
−4000
−2000
0
2000
4000

it is really close to normal distribution

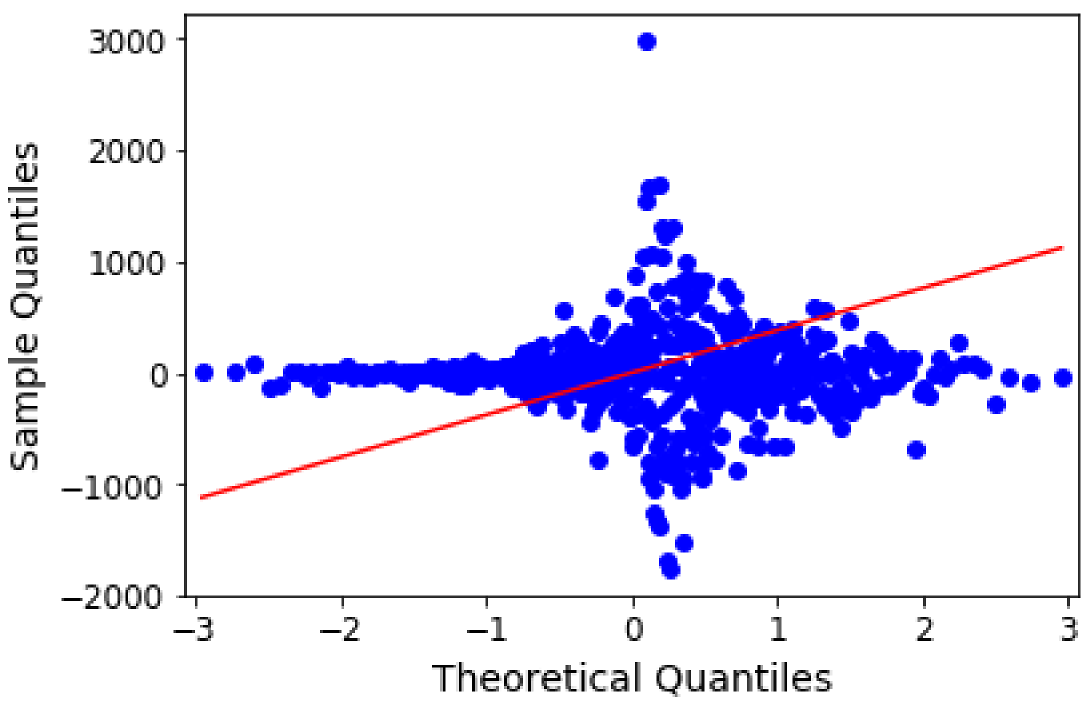
In [40]:
```
resid.describe()
```

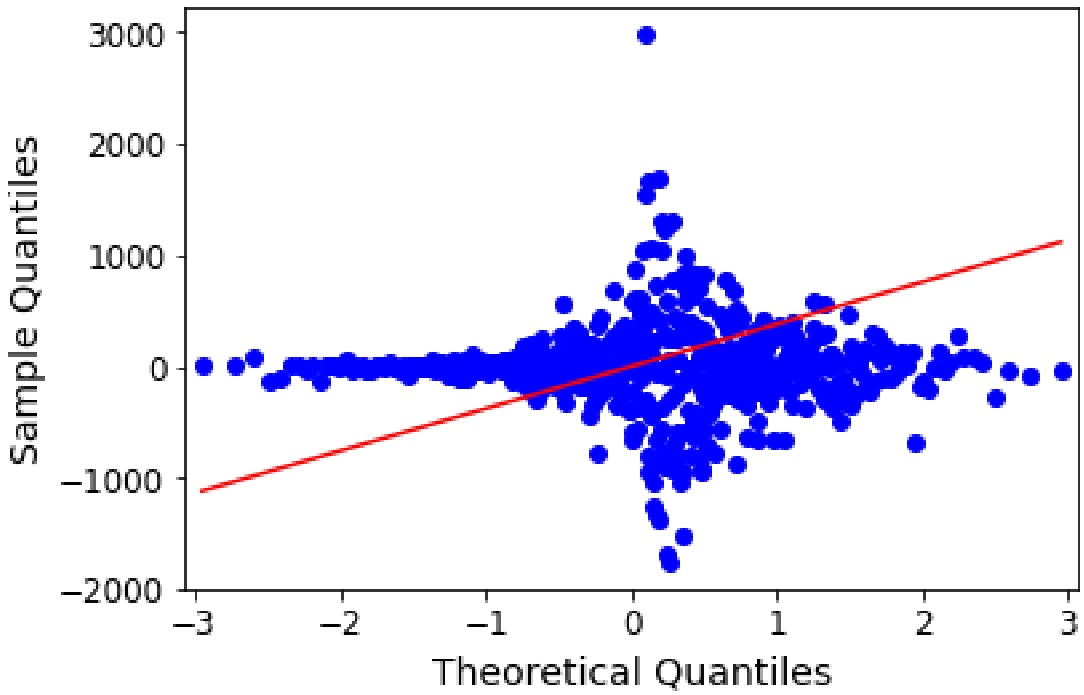
Out[40]:

	0
count	627.000000
mean	-0.032713
std	380.295937
min	-1761.413007
25%	-110.755254
50%	4.738762
75%	107.503017
max	2981.417723

In [41]:
```
sm.ProbPlot(resid).qqplot(line='s')
```

Out[41]:



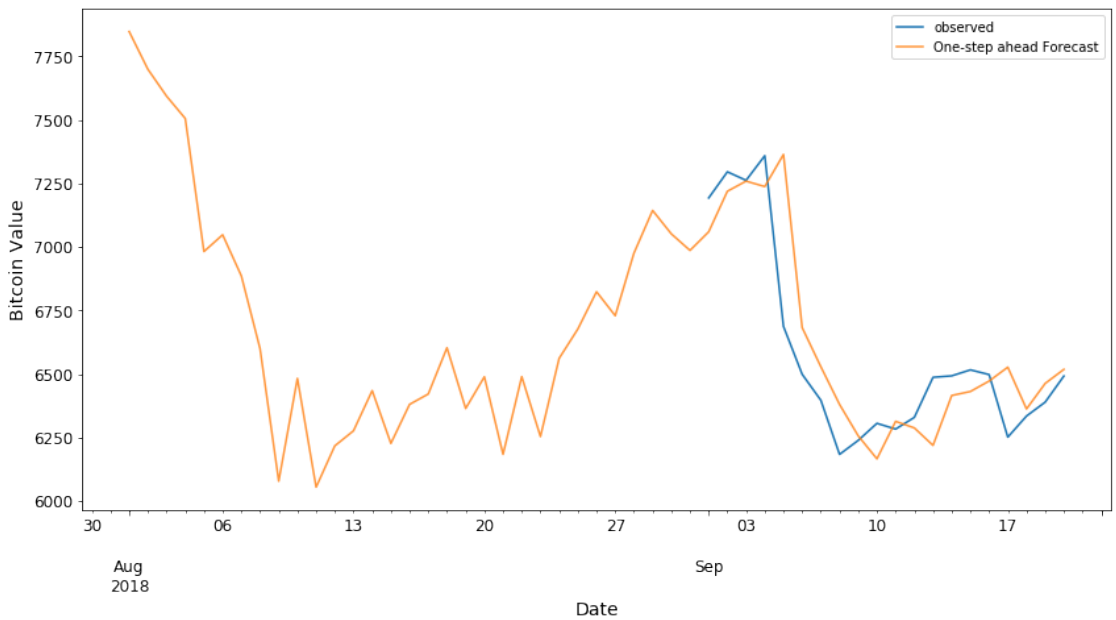


0.0.10 the mean of residuals is really close to zero. This model with p,d,q = 9,1,2 best describes the above process

0.0.11 q11

0.0.12 prediction - one step ahead forecast validation

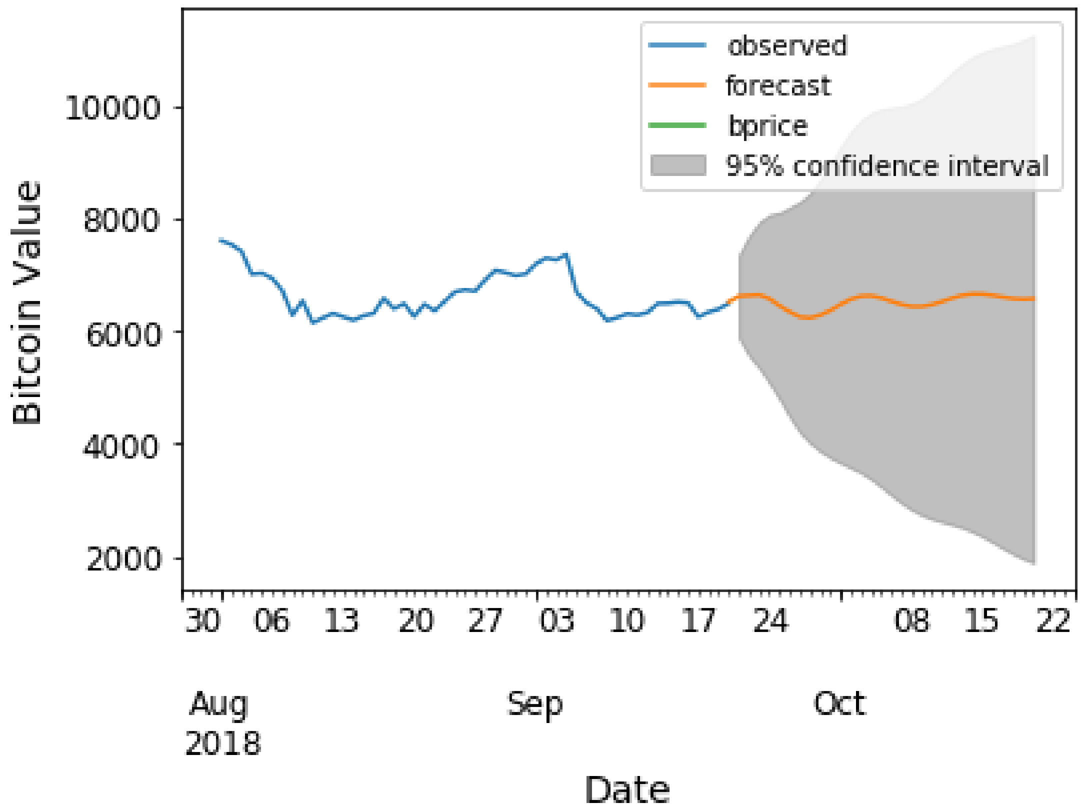
In [42]:
```
pred = results.predict(start=pd.to_datetime('2018-08-01'), dynamic=False,typ='levels')
pred_ci = pred
ax = y['2018-09':].plot(label='observed')
pred.plot(ax=ax, label='One-step ahead Forecast', alpha=.8, figsize=(14, 7))
#ax.fill_between(pred_ci.index, pred_ci.iloc[:], color='k', alpha=.2)
ax.set_xlabel('Date')
ax.set_ylabel('Bitcoin Value')
plt.legend()
plt.show()
```

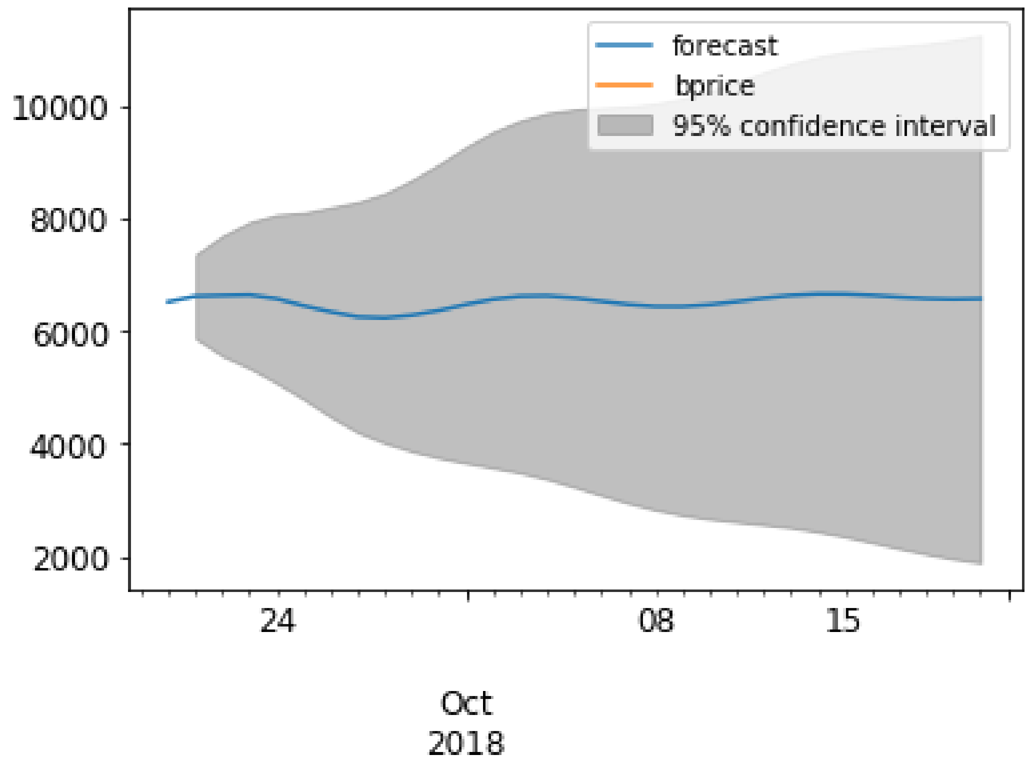


0.0.13 forecast

```
In [43]: from datetime import datetime
         from datetime import timedelta
         start_date = pd.to_datetime('2018-09-20')
         end_date = start_date+timedelta(days=int(input('Forecast Days: ')))
         fcast = results.predict(start=start_date,end = end_date, dynamic=False,typ='levels')
         fcast_ci = fcast
         ax = y['2018-08':].plot(label='observed')
         results.plot_predict(start_date,end_date)
         fig = results.plot_predict(start_date, end_date, dynamic=False, ax=ax)
         ax.set_xlabel('Date')
         ax.set_ylabel('Bitcoin Value')
         plt.legend()
         plt.show()
```

```
Forecast Days: 30
```


observed
forecast
bprice
95% confidence interval
Bitcoin Value
10000
8000
6000
4000
2000
30
06
13
20
27
03
10
17
24
08
15
22
Aug
2018
Sep
Oct
Date



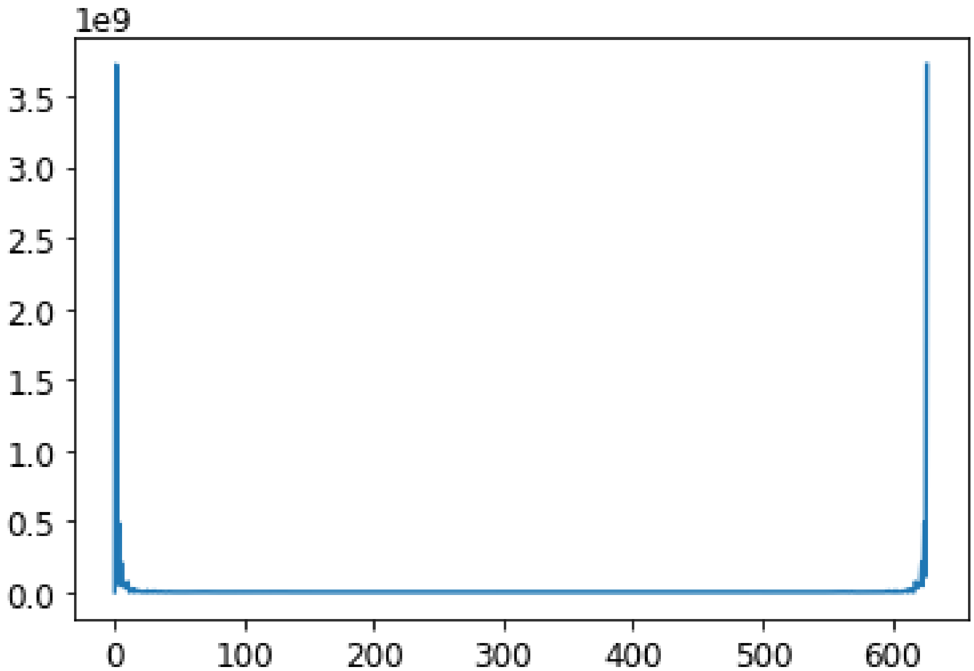
0.0.14 RMSE

```
In [44]: bitcoin_arima = results.forecast(steps=31)[0][1:]
         bitcoin_future=pd.read_csv('bitcoin_future.csv')
         np.sqrt((bitcoin_future['Closing'] - bitcoin_arima)**2).sum()

Out[44]: 5649.74656540898

In [45]: bitcoin_arima

Out[45]: array([6629.62938495, 6640.63603822, 6573.83524144, 6448.02817136,
                6339.56252902, 6250.59450055, 6234.06418159, 6280.40398395,
                6363.61910268, 6471.56747623, 6562.88598576, 6617.16717201,
                6624.44247367, 6586.81275405, 6528.33956559, 6470.92662542,
                6435.35539081, 6434.44747583, 6466.6431838 , 6522.06303965,
                6583.039639  , 6631.75585619, 6656.91175643, 6655.19103876,
                6632.65229123, 6601.27906356, 6574.55549505, 6563.47653152,
                6572.76946385, 6600.2255994 ])
```

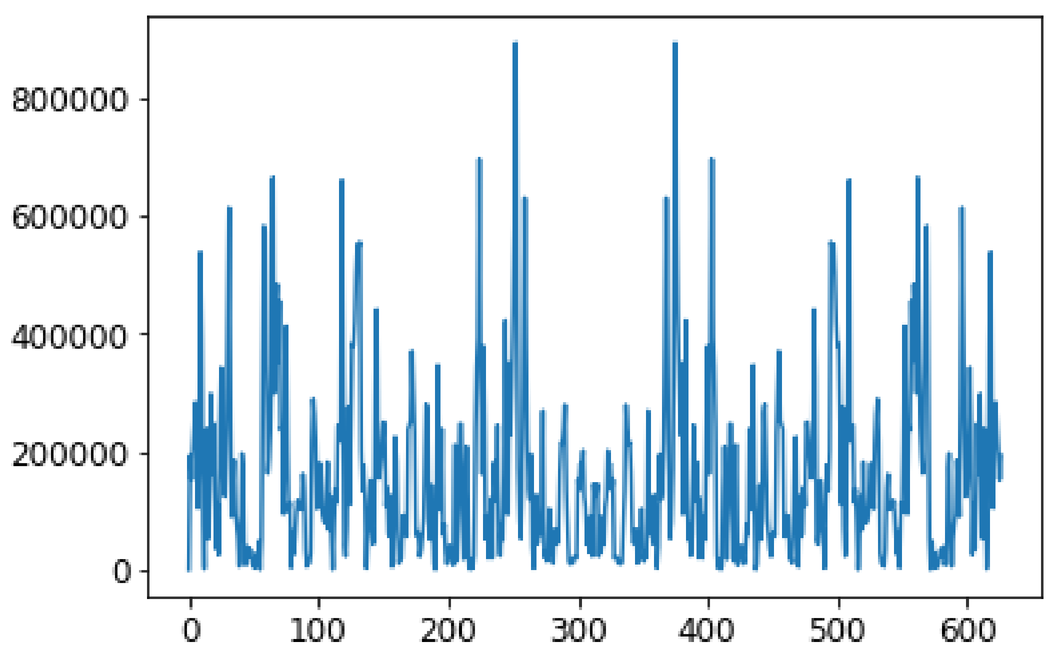
0.0.15 q12 - Periodogram

```
In [46]: # https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.periodogram.html
         from statsmodels.tsa.stattools import periodogram
```

```
plt.plot(periodogram(bprice))
plt.show()
plt.plot(periodogram(bprice_1d))
```




```
Out[46]: [<matplotlib.lines.Line2D at 0x1bf4fbb0518>]
```



0.0.16 There is no seasonality

Q13

In [47]:
```
final_1d = final_date_2017.diff().dropna()
final_1d
from statsmodels.tsa.vector_ar import var_model
var= var_model.VAR(final_1d)
np.argmin(var.select_order().ics['aic'])
var_results = var.fit(maxlags=1)
var_results.summary()
```

Out[47]:
```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Wed, 03, Apr, 2019
Time:                     16:20:21
--------------------------------------------------------------------
No. of Equations:         5.00000    BIC:                    9.06014
Nobs:                     626.000    HQIC:                   8.93006
Log likelihood:          -7180.51    FPE:                    6956.27
AIC:                      8.84740    Det(Omega_mle):         6632.28
--------------------------------------------------------------------
Results for equation bprice
================================================================================
                 coefficient       std. error           t-stat            prob
--------------------------------------------------------------------------------
const               8.905596        16.158048            0.551           0.582
L1.bprice           0.069110         0.040040            1.726           0.084
L1.goldprice        1.336761         2.854434            0.468           0.640
L1.sp               0.422667         1.263261            0.335           0.738
L1.forex        -1546.995382      4334.377682           -0.357           0.721
L1.oil            -40.307750        21.758846           -1.852           0.064
================================================================================

Results for equation goldprice
================================================================================
                 coefficient       std. error           t-stat            prob
--------------------------------------------------------------------------------
const              -0.002602         0.228070           -0.011           0.991
L1.bprice          -0.001013         0.000565           -1.792           0.073
L1.goldprice        0.063268         0.040290            1.570           0.116
L1.sp              -0.009138         0.017831           -0.512           0.608
L1.forex          382.948062        61.179613            6.259           0.000
L1.oil              1.066983         0.307125            3.474           0.001
```

Results for equation sp

	coefficient	std. error	t-stat	prob
const	0.969575	0.520856	1.862	0.063
L1.bprice	0.001679	0.001291	1.301	0.193
L1.goldprice	0.174788	0.092013	1.900	0.057
L1.sp	0.071246	0.040721	1.750	0.080
L1.forex	159.523161	139.718964	1.142	0.254
L1.oil	-0.798069	0.701398	-1.138	0.255

Results for equation forex

	coefficient	std. error	t-stat	prob
const	0.000155	0.000157	0.984	0.325
L1.bprice	-0.000000	0.000000	-0.367	0.714
L1.goldprice	-0.000002	0.000028	-0.077	0.939
L1.sp	0.000016	0.000012	1.266	0.206
L1.forex	0.090106	0.042178	2.136	0.033
L1.oil	0.000241	0.000212	1.137	0.255

Results for equation oil

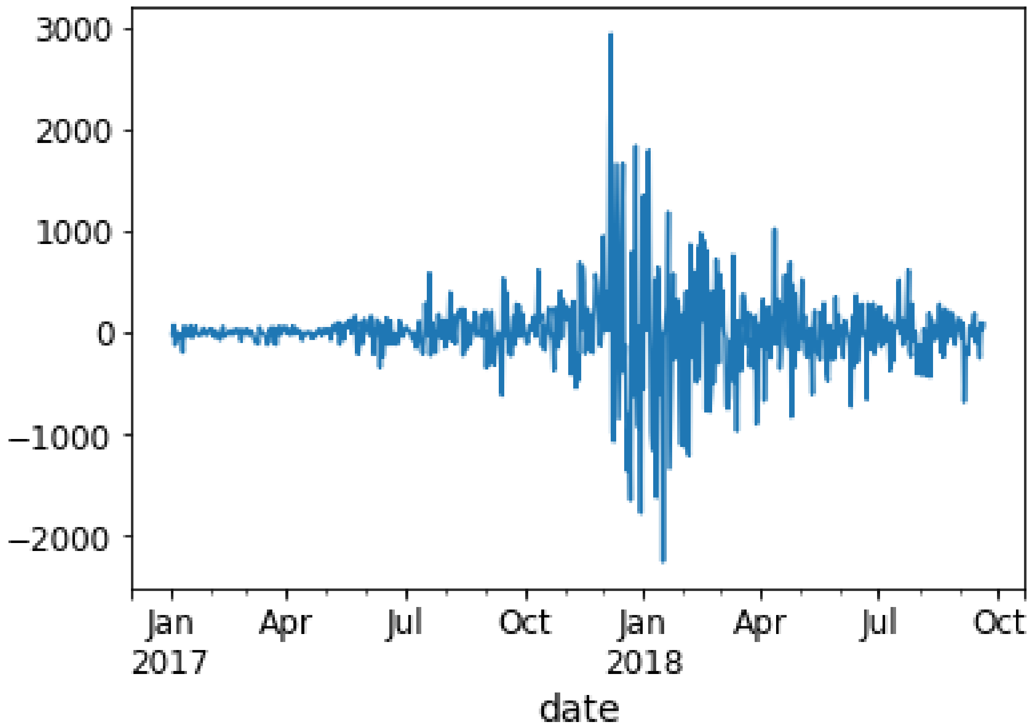
	coefficient	std. error	t-stat	prob
const	0.021184	0.030376	0.697	0.486
L1.bprice	0.000006	0.000075	0.081	0.935
L1.goldprice	0.005188	0.005366	0.967	0.334
L1.sp	0.002826	0.002375	1.190	0.234
L1.forex	1.270291	8.148196	0.156	0.876
L1.oil	0.037341	0.040904	0.913	0.361

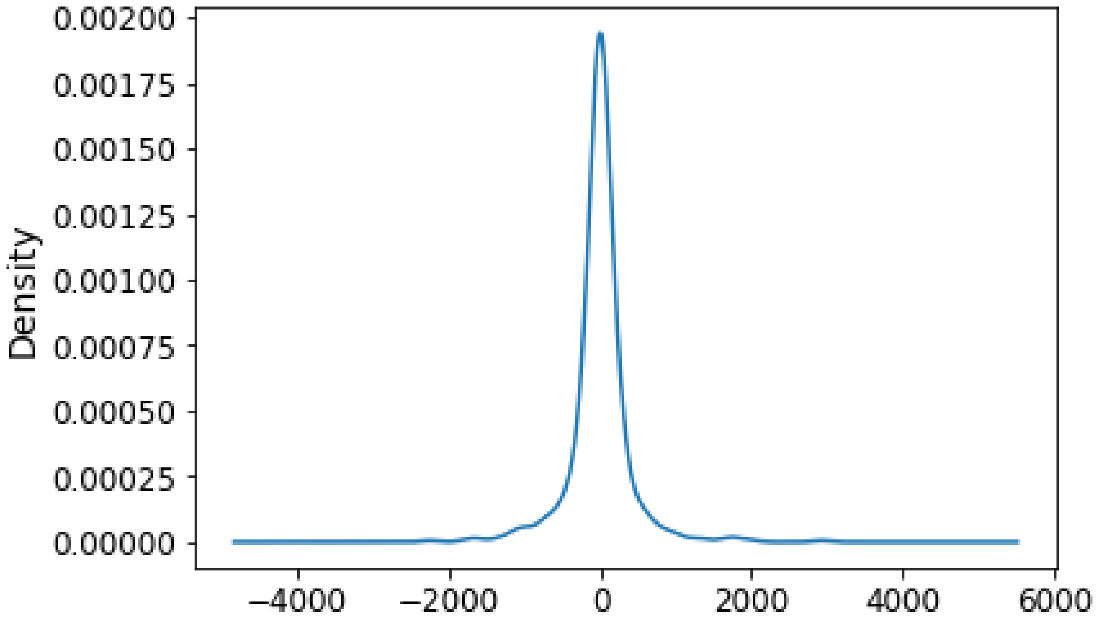
Correlation matrix of residuals

	bprice	goldprice	sp	forex	oil
bprice	1.000000	0.037788	0.051801	-0.015241	0.019423
goldprice	0.037788	1.000000	-0.042486	0.311933	0.053672
sp	0.051801	-0.042486	1.000000	-0.018298	0.189110
forex	-0.015241	0.311933	-0.018298	1.000000	0.033503
oil	0.019423	0.053672	0.189110	0.033503	1.000000

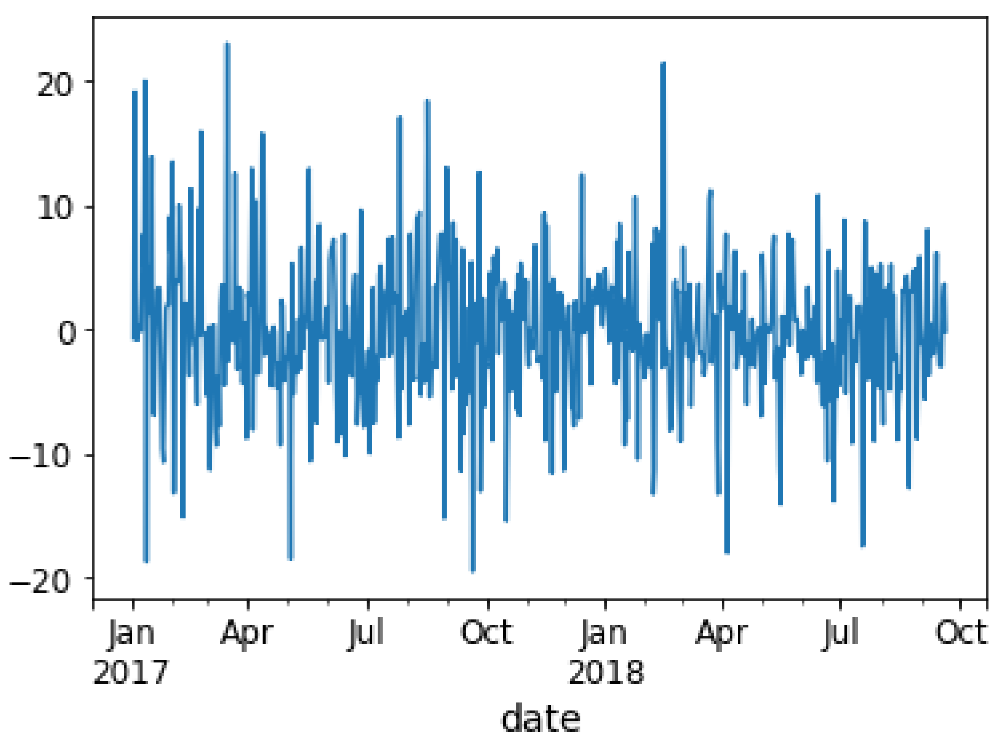
In [48]:
```
resid.columns
resid = pd.DataFrame(var_results.resid)
for i in resid.columns:
    print(i)
    resid[i].plot()
    plt.show()
    resid[i].plot(kind='kde')
    plt.show()
```

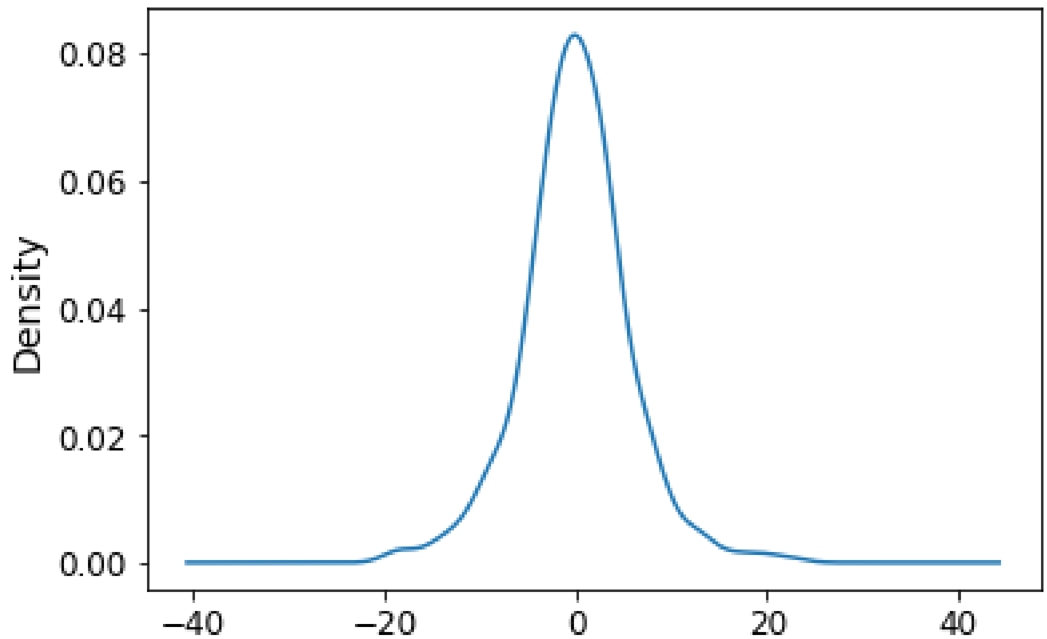
bprice



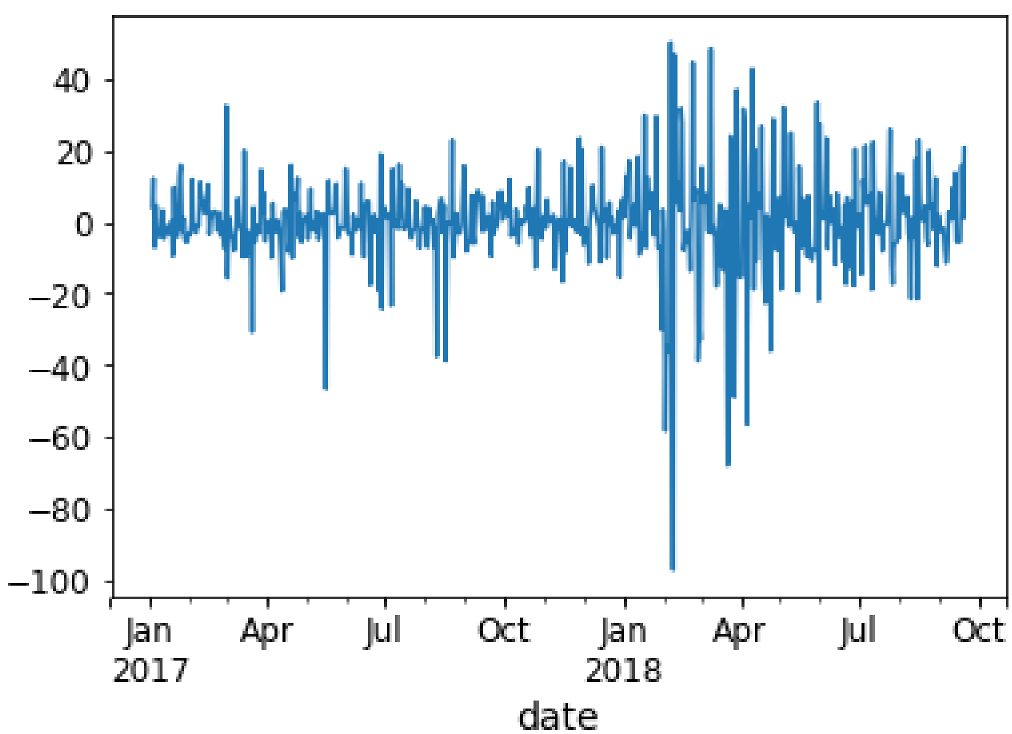

Density
0.00200
0.00175
0.00150
0.00125
0.00100
0.00075
0.00050
0.00025
0.00000
−4000
−2000
0
2000
4000
6000

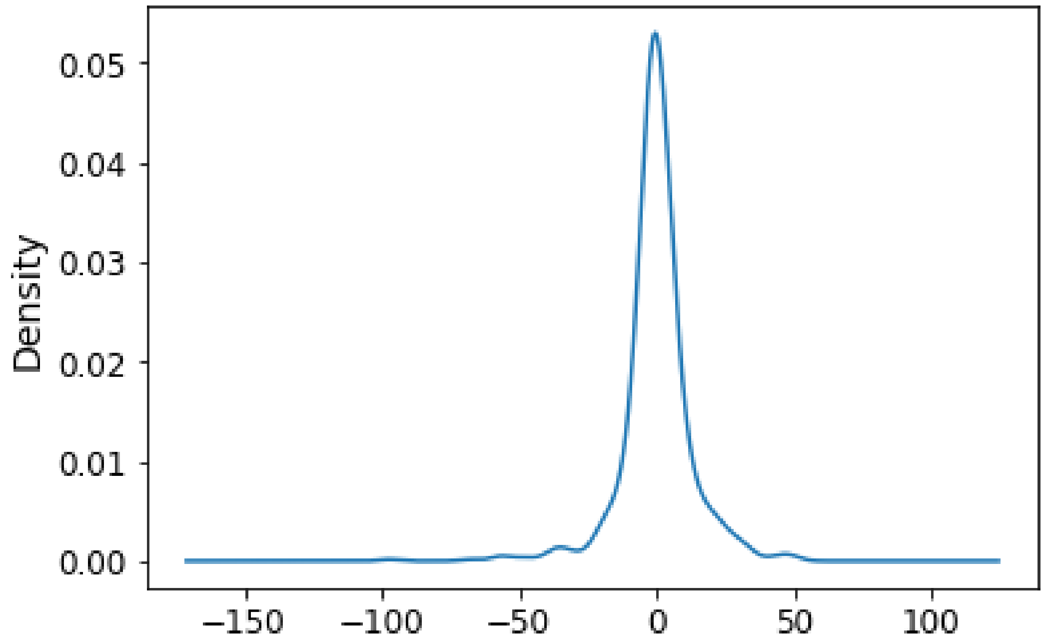
`goldprice`


20
10
0
−10
−20
Jan
2017
Apr
Jul
Oct
Jan
2018
Apr
Jul
Oct
date

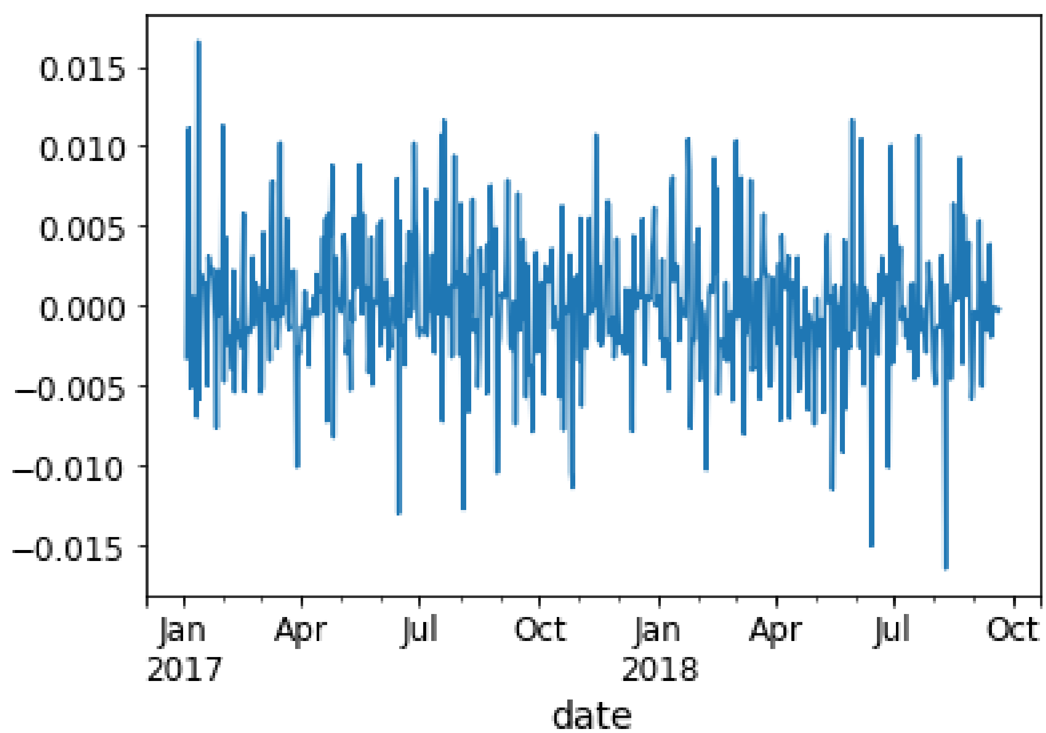

Density
0.08
0.06
0.04
0.02
0.00
−40
−20
0
20
40

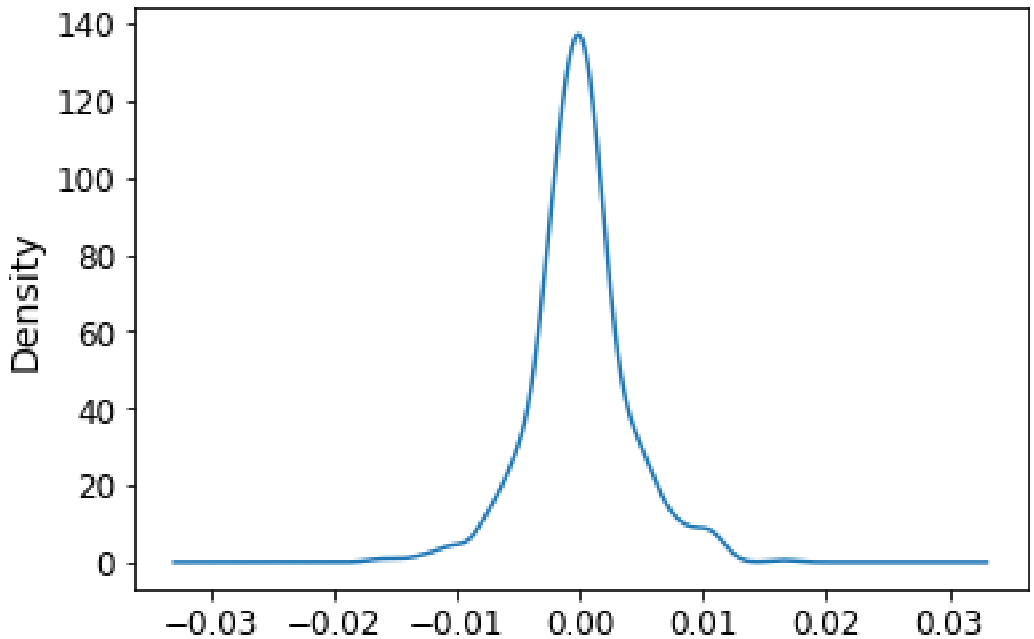
sp


40
20
0
−20
−40
−60
−80
−100
Jan
2017
Apr
Jul
Oct
Jan
2018
Apr
Jul
Oct
date

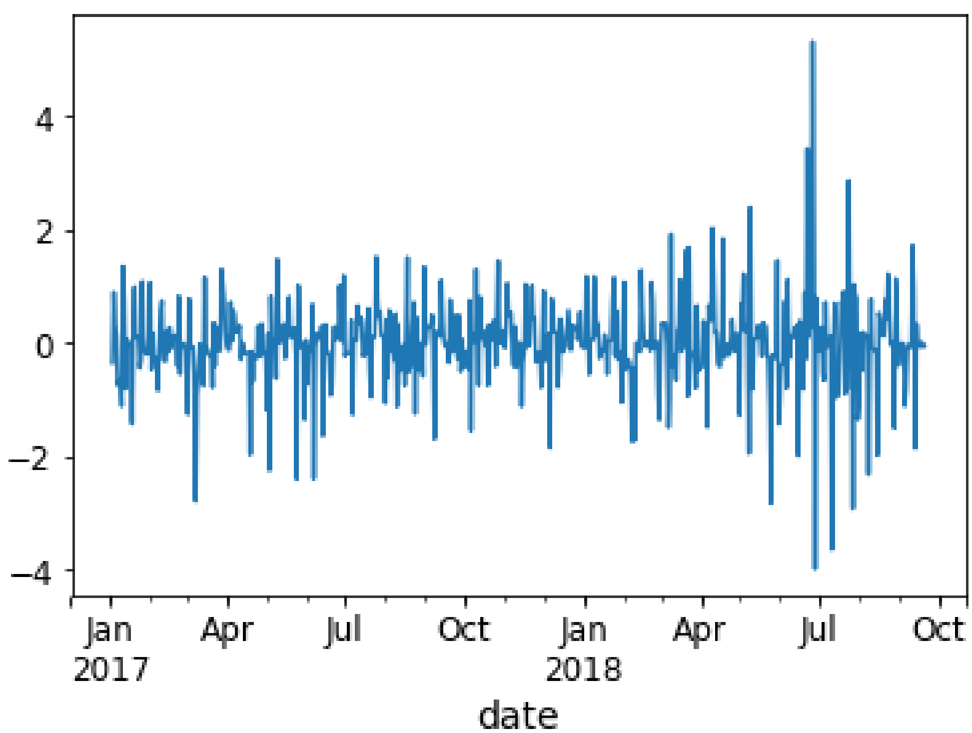

Density
0.05
0.04
0.03
0.02
0.01
0.00
−150
−100
−50
0
50
100

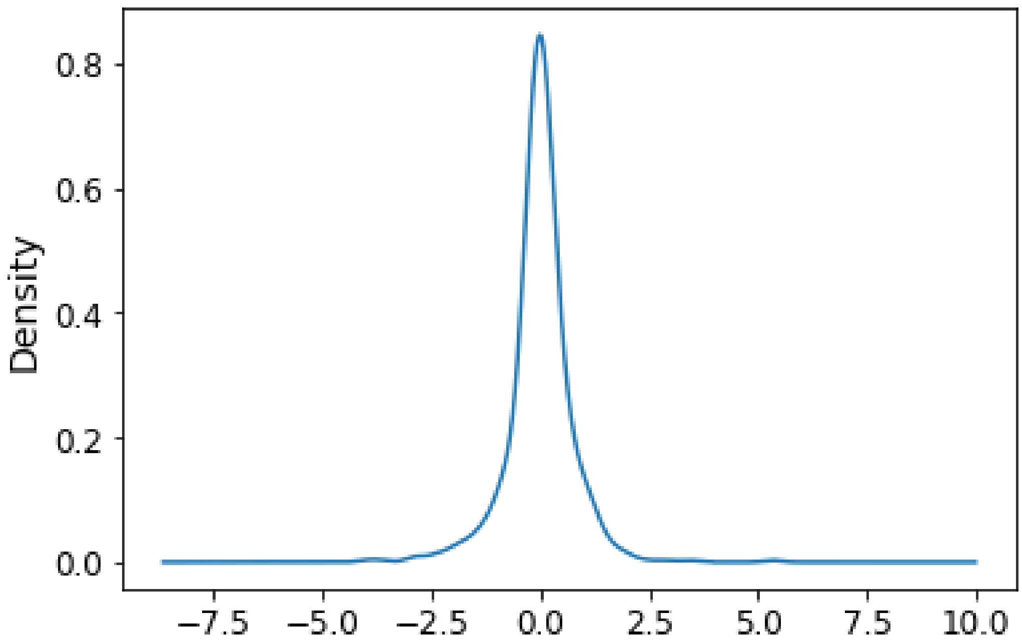
forex


0.015
0.010
0.005
0.000
−0.005
−0.010
−0.015
Jan
2017
Apr
Jul
Oct
Jan
2018
Apr
Jul
Oct
date


140
120
100
80
60
40
20
0
Density
−0.03
−0.02
−0.01
0.00
0.01
0.02
0.03

oil


4
2
0
−2
−4
Jan
2017
Apr
Jul
Oct
Jan
2018
Apr
Jul
Oct
date




```
In [49]: resid.describe()
```

```
Out[49]:              bprice     goldprice            sp         forex           oil
         count  6.260000e+02  6.260000e+02  6.260000e+02  6.260000e+02  6.260000e+02
         mean  -5.743365e-15 -1.059013e-15 -8.910161e-16 -3.288112e-19  3.258842e-18
         std    4.007564e+02  5.656664e+00  1.291841e+01  3.899774e-03  7.533819e-01
         min   -2.247581e+03 -1.946989e+01 -9.730499e+01 -1.643226e-02 -3.983851e+00
         25%   -1.065176e+02 -2.996391e+00 -4.000892e+00 -1.943141e-03 -2.641911e-01
         50%    1.922872e+00 -1.533415e-02 -3.771619e-02 -1.191374e-04 -9.623573e-04
         75%    1.160385e+02  3.125817e+00  4.639900e+00  1.628155e-03  2.995310e-01
         max    2.930638e+03  2.303008e+01  5.031274e+01  1.654610e-02  5.319883e+00
```

All the residuals are very close to zero, this suggests that the VAR model of lag 1 fits the above data very good.

0.0.17 q14

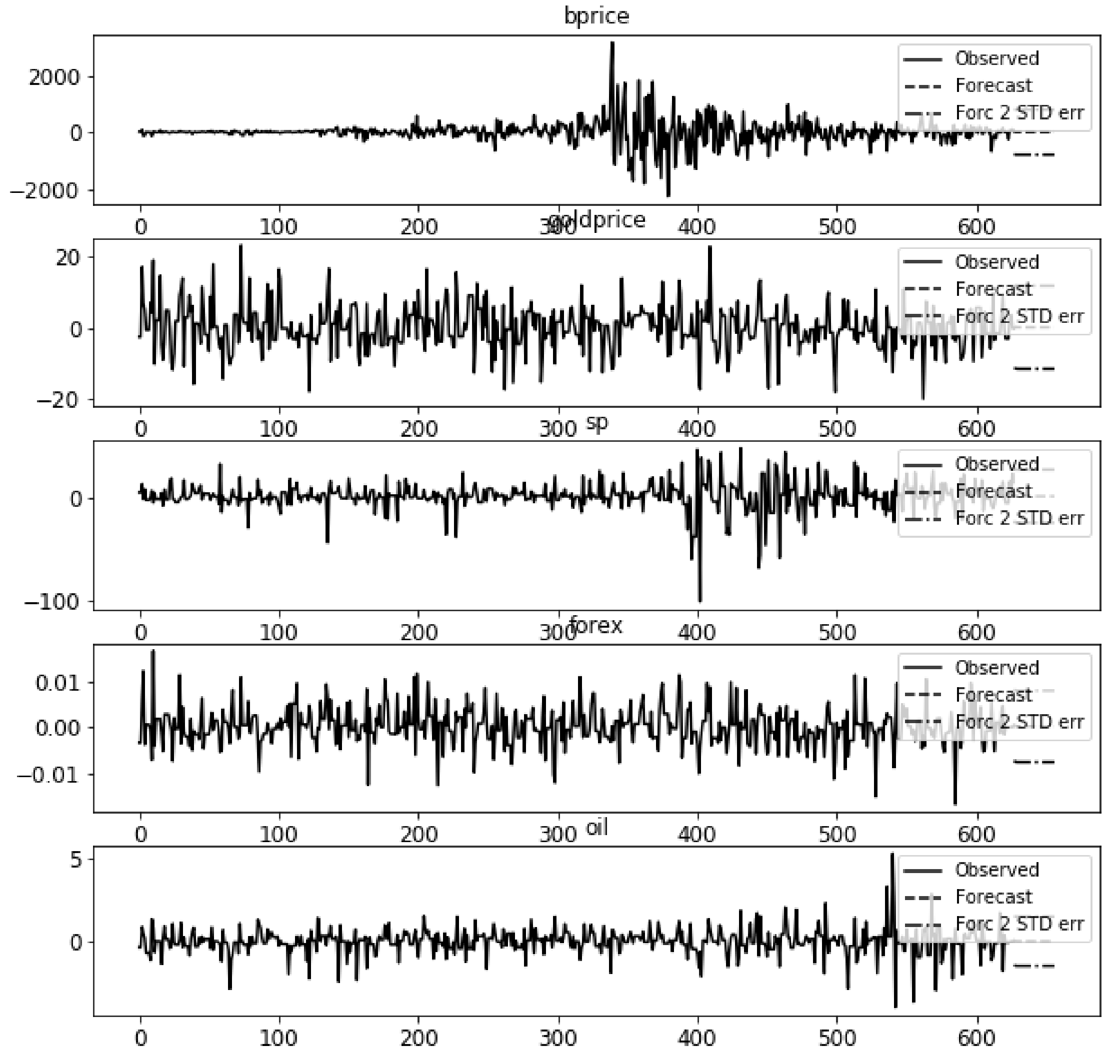
```
In [50]: final_1d.tail()
```

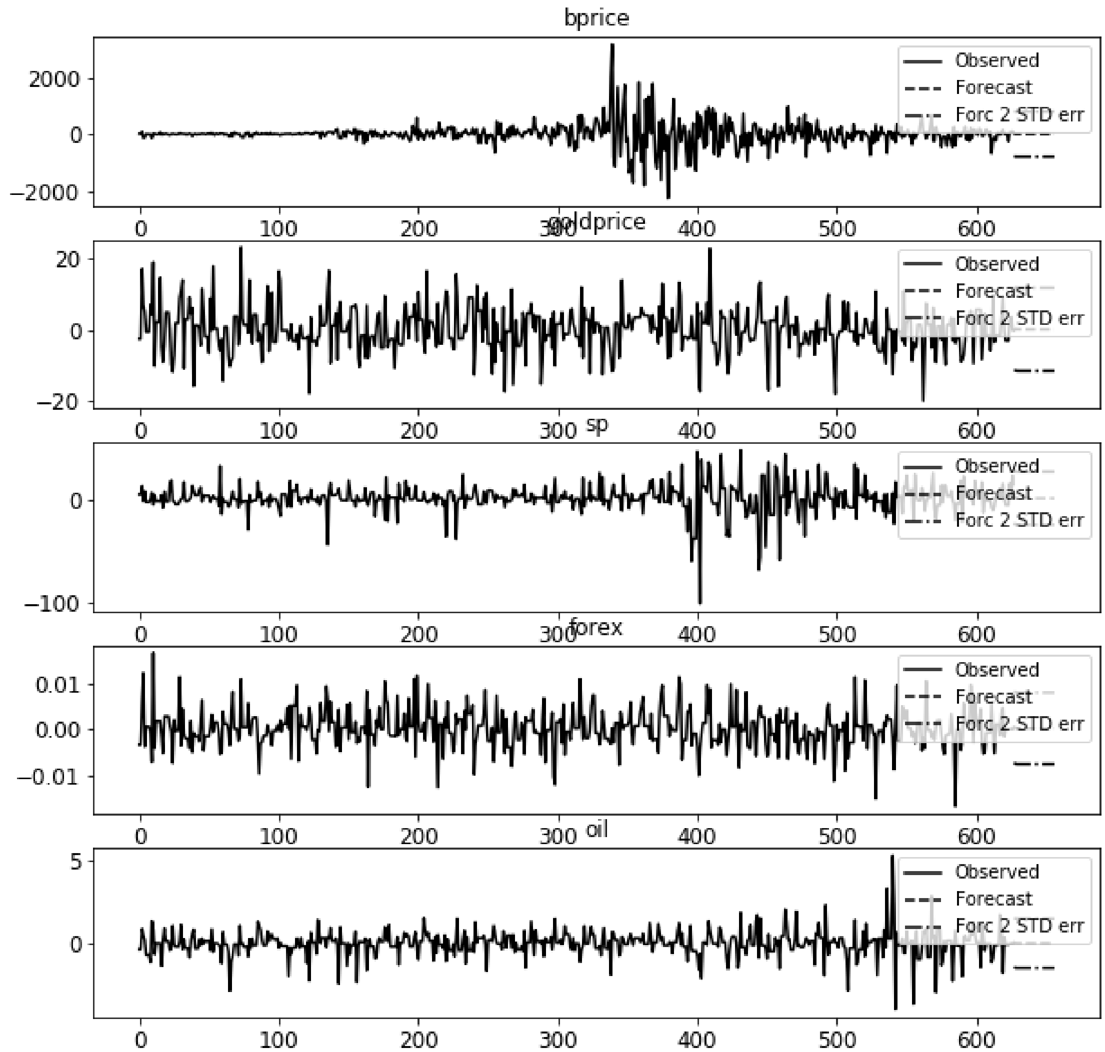
```
Out[50]:             bprice  goldprice         sp  forex   oil
         date
         2018-09-16  -18.53  -3.133333  -5.393333    0.0 -0.04
         2018-09-17 -246.21  -3.133333  -5.393333    0.0 -0.04
```

```
2018-09-18    83.04   2.600000  15.510000    0.0  0.00
2018-09-19    54.78   3.600000   3.640000    0.0  0.00
2018-09-20   102.66   0.000000  22.800000    0.0  0.00
```

In [51]:
```
# https://www.statsmodels.org/stable/tsa.html#vector-autogressive-processes-var
# https://www.statsmodels.org/stable/generated/statsmodels.tsa.vector_ar.var_model.VARR
# https://www.statsmodels.org/stable/generated/statsmodels.tsa.vector_ar.var_model.VARP
# https://www.statsmodels.org/stable/generated/statsmodels.tsa.vector_ar.var_model.VAR.
# https://www.statsmodels.org/stable/generated/statsmodels.tsa.vector_ar.var_model.VARP

var_results.plot_forecast(steps=30,alpha=0.05)
```

Out[51]:






```
In [52]: var_forecast = var_results.forecast(var_results.y,steps=30)
         bitcoin_var_op = var_forecast[:,0]
         bitcoin_forcast_var = bprice[-1]+np.cumsum(bitcoin_var_op)

In [53]: bitcoin_future=pd.read_csv('bitcoin_future.csv')
         np.sqrt((bitcoin_future['Closing']-bitcoin_forcast_var)**2).sum()

Out[53]: 4589.177796876813
```

rmse is around 4600

```
In [55]: bitcoin_forcast_var

Out[55]: array([6517.27721312, 6524.46088274, 6532.96592991, 6541.69425772,
                6550.4276275 , 6559.16321022, 6567.89987081, 6576.6367718 ,
```

```
6585.37371769, 6594.1106727 , 6602.84762968, 6611.58458706,
6620.32154454, 6629.05850203, 6637.79545952, 6646.53241702,
6655.26937451, 6664.00633201, 6672.7432895 , 6681.480247  ,
6690.21720449, 6698.95416199, 6707.69111948, 6716.42807698,
6725.16503447, 6733.90199197, 6742.63894946, 6751.37590696,
6760.11286445, 6768.84982195])
```