Fit ARIMA models on the DailyRet time series

June 12, 2019

Name: Xuan Zhang

0.1 Problem

Univariate times series modeling of the S&P 500 over the period 2004-2006.

The data file "stock-treasury-2004_2006.csv", to be found in the "Data" folder, contains the following:
+ TREAS_3M: the yield of the 3-month treasury note in percent (i.e 2.1 means 2.1%) + Adjusted close price of ten major stocks: GM, F, UTX, CAT, MRK, PFE, IBM, MSFT, C, XOM + SP: The S&P 500 equity index level at the close of the trading day

Do the following:

Use the pandas read_csv function to read only the Date and SP columns in a data frame called "sp_df".
Rename the "SP" column into "ClosePx" in the same read_csv call.
Compute the close-to-close index returns as: $r_t = P_{t+1}/P_t - 1$ and add them as a new column "DailyRet".
It is recommended to express all daily returns in basis points (10,000 bps = 100% = 1)

Fit ARIMA models on the DailyRet time series, up to AR order p=2, MA order q=2, and differencing order d=1.
You can reuse the utility function auto_arima or provide your own.
Display the summary of the best selected model based on the AIC criterion.
Plot the original returns series and the predictions of the best selected model using the model's plot_predict method.

Run the Jarque-Bera normality test on the residuals of the best selected ARIMA model, and produce the qq plot of the residuals.
Repeat the Jarque-Bera test and the qq plot using the residuals of the white noise model ARMA(0, 0).
Compare the two and comment on whether they are really different.

0.2 Solution

<...>

```
In [1]: import pandas as pd
        import numpy as np
        from mlutils import run_adf_test, auto_arima
        import statsmodels.api as sm
```

```python
import matplotlib.pyplot as plt
plt.style.use('ggplot')

sp_df=pd.read_csv("stock-treasury-2004_2006.csv",
                  usecols=[0,12],parse_dates=['Date'],header=0,names=['Date','ClosePx']

sp_df.head()
```

Out[1]:

	Date	ClosePx
0	2004-01-02	1108.48
1	2004-01-05	1122.22
2	2004-01-06	1123.67
3	2004-01-07	1126.33
4	2004-01-08	1131.92

In [2]:
```python
sp_df = sp_df.assign(DailyRet=10000 * (sp_df.ClosePx / sp_df.ClosePx.shift() - 1))
sp_df.head()
```

Out[2]:

	Date	ClosePx	DailyRet
0	2004-01-02	1108.48	NaN
1	2004-01-05	1122.22	123.953522
2	2004-01-06	1123.67	12.920818
3	2004-01-07	1126.33	23.672431
4	2004-01-08	1131.92	49.630215

In [3]:
```python
best_arima, results = auto_arima(sp_df.DailyRet[1:].values, p_max = 2, q_max = 2, d_ma
```

```
D:\anaconda3\lib\site-packages\scipy\signal\signaltools.py:1341: FutureWarning: Using a non-tu
  out_full[ind] += zi
D:\anaconda3\lib\site-packages\scipy\signal\signaltools.py:1344: FutureWarning: Using a non-tu
  out = out_full[ind]
D:\anaconda3\lib\site-packages\scipy\signal\signaltools.py:1350: FutureWarning: Using a non-tu
  zf = out_full[ind]


ARIMA(0, 0, 0) AIC:7578.85  BIC:7587.87
ARIMA(0, 0, 1) AIC:7579.90  BIC:7593.43
ARIMA(0, 0, 2) AIC:7578.43  BIC:7596.47
ARIMA(0, 1, 0) AIC:8054.73  BIC:8063.75
ARIMA(0, 1, 1) AIC:7577.06  BIC:7590.58
The computed initial MA coefficients are not invertible
You should induce invertibility, choose a different model order, or you can
pass your own start_params.
ARIMA(1, 0, 0) AIC:7580.03  BIC:7593.56
The computed initial AR coefficients are not stationary
You should induce stationarity, choose a different model order, or you can
pass your own start_params.
The computed initial AR coefficients are not stationary
You should induce stationarity, choose a different model order, or you can
```

```
pass your own start_params.
ARIMA(1, 1, 0) AIC:7879.46  BIC:7892.99
ARIMA(1, 1, 1) AIC:7578.31  BIC:7596.34
```

```
D:\anaconda3\lib\site-packages\statsmodels\base\model.py:488: HessianInversionWarning: Invertin
  'available', HessianInversionWarning)
D:\anaconda3\lib\site-packages\statsmodels\base\model.py:508: ConvergenceWarning: Maximum Likel
  "Check mle_retvals", ConvergenceWarning)
```

```
ARIMA(1, 1, 2) AIC:7575.34  BIC:7597.88
ARIMA(2, 0, 0) AIC:7578.58  BIC:7596.62
ARIMA(2, 0, 1) AIC:7579.44  BIC:7601.99
```

```
D:\anaconda3\lib\site-packages\statsmodels\base\model.py:488: HessianInversionWarning: Invertin
  'available', HessianInversionWarning)
```

```
ARIMA(2, 0, 2) AIC:7574.90  BIC:7601.96
ARIMA(2, 1, 0) AIC:7774.89  BIC:7792.93
ARIMA(2, 1, 1) AIC:7576.99  BIC:7599.54
ARIMA(2, 1, 2) AIC:7577.78  BIC:7604.83
Best model params:(2, 0, 2) AIC:7574.90  BIC:7601.96
```

In [4]:
```
best_arima['model'].summary()
```

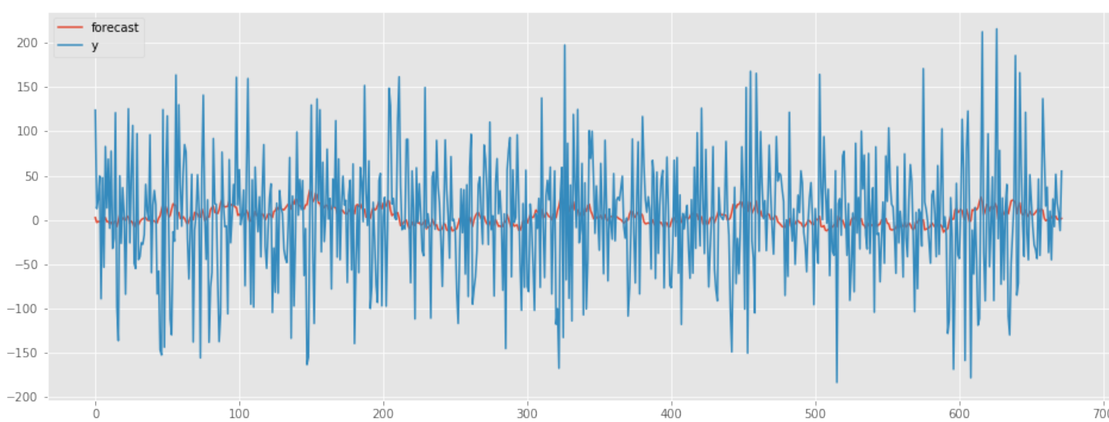
```
D:\anaconda3\lib\site-packages\statsmodels\tsa\arima_model.py:1455: RuntimeWarning: invalid val
  return np.sqrt(np.diag(-inv(hess)))
D:\anaconda3\lib\site-packages\scipy\stats\_distn_infrastructure.py:879: RuntimeWarning: invali
  return (self.a < x) & (x < self.b)
D:\anaconda3\lib\site-packages\scipy\stats\_distn_infrastructure.py:879: RuntimeWarning: invali
  return (self.a < x) & (x < self.b)
D:\anaconda3\lib\site-packages\scipy\stats\_distn_infrastructure.py:1821: RuntimeWarning: inval
  cond2 = cond0 & (x <= self.a)
```

Out[4]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                              ARMA Model Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  672
Model:                     ARMA(2, 2)   Log Likelihood               -3781.451
Method:                       css-mle   S.D. of innovations             67.108
Date:                Sun, 14 Apr 2019   AIC                           7574.902
Time:                        17:29:58   BIC                           7601.964
Sample:                             0   HQIC                          7585.383
```

```
================================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const          2.7419      0.315      8.699      0.000       2.124       3.360
ar.L1.y        1.4719        nan        nan        nan         nan         nan
ar.L2.y       -0.4893        nan        nan        nan         nan         nan
ma.L1.y       -1.5372        nan        nan        nan         nan         nan
ma.L2.y        0.5372        nan        nan        nan         nan         nan
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.0366           +0.0000j            1.0366            0.0000
AR.2            1.9717           +0.0000j            1.9717            0.0000
MA.1            1.0000           +0.0000j            1.0000            0.0000
MA.2            1.8616           +0.0000j            1.8616            0.0000
-----------------------------------------------------------------------------
"""
```

In [5]:
```python
armodel = best_arima['model']

fig2 = plt.figure(figsize=(16,6))
axx = fig2.add_subplot(111)
armodel.plot_predict(ax = axx);
```



In [6]:
```python
arresid = armodel.resid
jbres = sm.stats.jarque_bera(arresid)

print("Jarque-Bera Normality Test on AR Residuals")
print("statistic: {}".format(jbres[0]))
print("p-value: {}".format(jbres[1]))
```

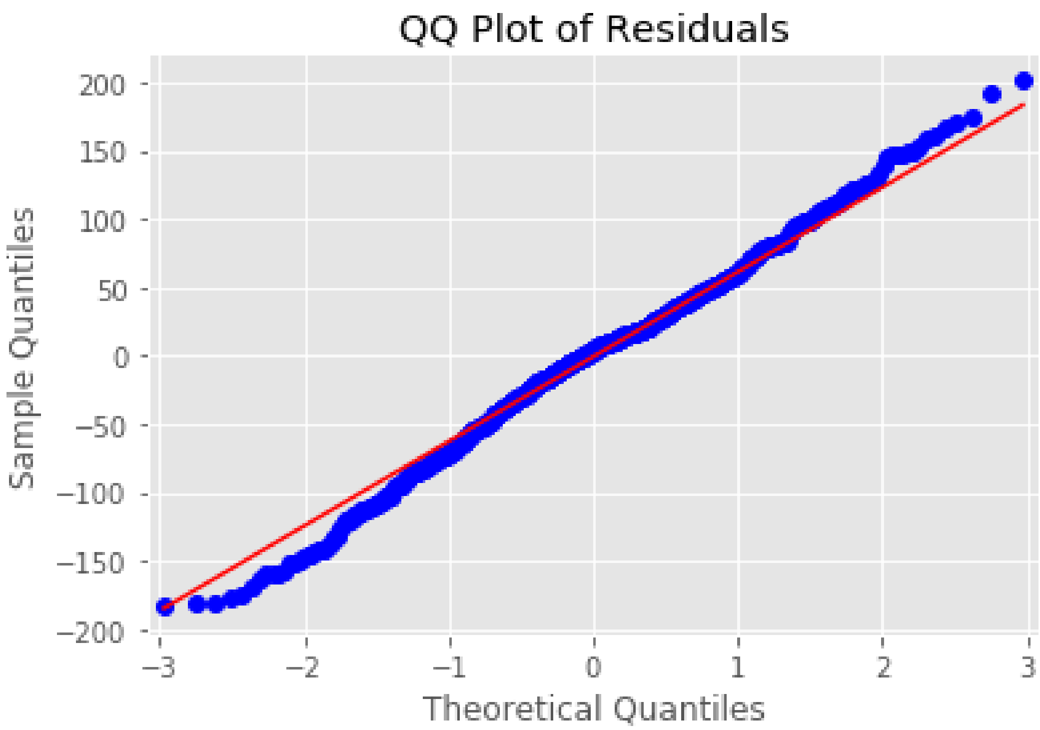
```
        print('skew: {}'.format(jbres[2]))
        print('kurtosis: {}'.format(jbres[3]))
```

```
Jarque-Bera Normality Test on AR Residuals
statistic: 1.5098564497740838
p-value: 0.4700443510061998
skew: -0.10713348436248081
kurtosis: 3.0895159804068806
```

```
In [7]: axxx = plt.figure().add_subplot(111)
        sm.graphics.qqplot(arresid, line='q', ax = axxx);
        axxx.set(title = 'QQ Plot of Residuals')
```

```
D:\anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a non-tuple seq
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

```
Out[7]: [Text(0.5, 1.0, 'QQ Plot of Residuals')]
```




```
In [8]: best_arima1, results1 = auto_arima(sp_df.DailyRet[1:].values, p_max = 0, q_max = 0, d_
        armodel1 = best_arima1['model']
        arresid1 = armodel1.resid
```

```
ARIMA(0, 0, 0) AIC:7578.85  BIC:7587.87
Best model params:(0, 0, 0) AIC:7578.85  BIC:7587.87
```

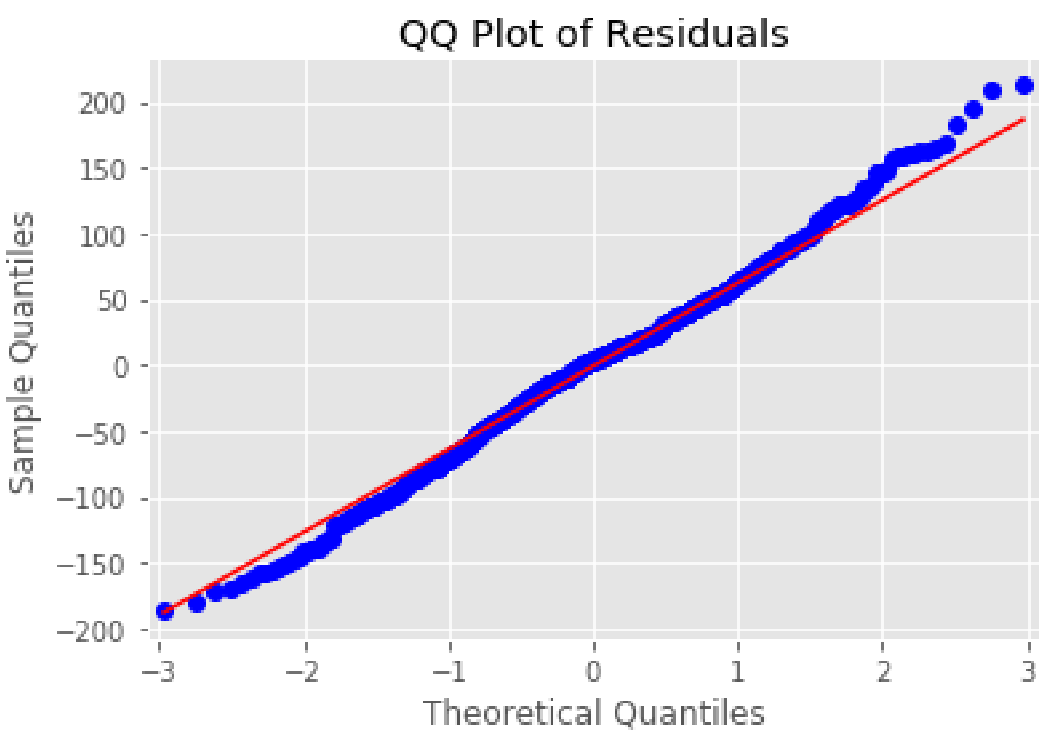
```
In [9]: jbres1 = sm.stats.jarque_bera(arresid1)

        print("Jarque-Bera Normality Test on AR Residuals")
        print("statistic: {}".format(jbres1[0]))
        print("p-value: {}".format(jbres1[1]))
        print('skew: {}'.format(jbres1[2]))
        print('kurtosis: {}'.format(jbres1[3]))
```

```
Jarque-Bera Normality Test on AR Residuals
statistic: 0.6653429629453448
p-value: 0.717005705127056
skew: 0.02162822835081098
kurtosis: 3.1479565058733145
```

```
In [10]: axxx = plt.figure().add_subplot(111)
         sm.graphics.qqplot(arresid1, line='q', ax = axxx);
         axxx.set(title = 'QQ Plot of Residuals')
```

```
Out[10]: [Text(0.5, 1.0, 'QQ Plot of Residuals')]
```



0.2.1 According to Jarque-Bera Normality Test, the ARIMA(0, 0, 0) is normally distributed, while the ARIMA(2, 0, 2) is not. However, their qq plots are very similar. Therefore, we cannot reject the null hypothesis. Both of them don't have large deviation from normal distribution.