```
In [141]: import numpy as np
          import pandas as pd
          import matplotlib.pyplot as plt
          import statsmodels.api as sm
```

Question 1 Time Series

We're proposing an ARIMA(0,1,1) model for y_t. Reasons are below:

-Without any differencing, the series look highly correlated and the ACF plot also shows high correlation for all first 10 lags.

-After first order differencing, the series look stationary. The ACF plot has a clear cutoff at lag 1 while the PACF plot decaying, indicating a MA(1) model for this differenced process.

-After second order differencing, the series look stationary. The ACF plot has a clear cutoff at lag 2 while the PACF plot decaying, indicating a MA(2) model for this 2nd order differenced process.

-In summary, the original series should be modeled using ARIMA(0,1,1)

As for the sign of the MA parameter, we believe it's positive. Reasons are below:

-ACF(1) for Δy_t is positive

-The sign of PACF for Δy_t is alternating

-ACF(2) for $\Delta^2 y_t$ is negative

-PACF for $\Delta^2 y_t$ is decaying exponentially for even lags but is very small for odd lags

Question 3 ARIMA

In [194]:
```
# Read in dataset
RawData = []
with open('m-COILWTICO.txt','r') as file:
    for line in file:
        if not line.startswith("#"):
            RawData.append(line.split())

DATA = []
col = RawData[0]
for i in range(1, len(RawData)):
    DATA.append([pd.to_datetime(RawData[i][0]), float(RawData[i][1])])

oil_prices = pd.DataFrame(DATA, columns=col)
oil_prices = oil_prices.set_index(['DATE'])
oil_prices.head()
```

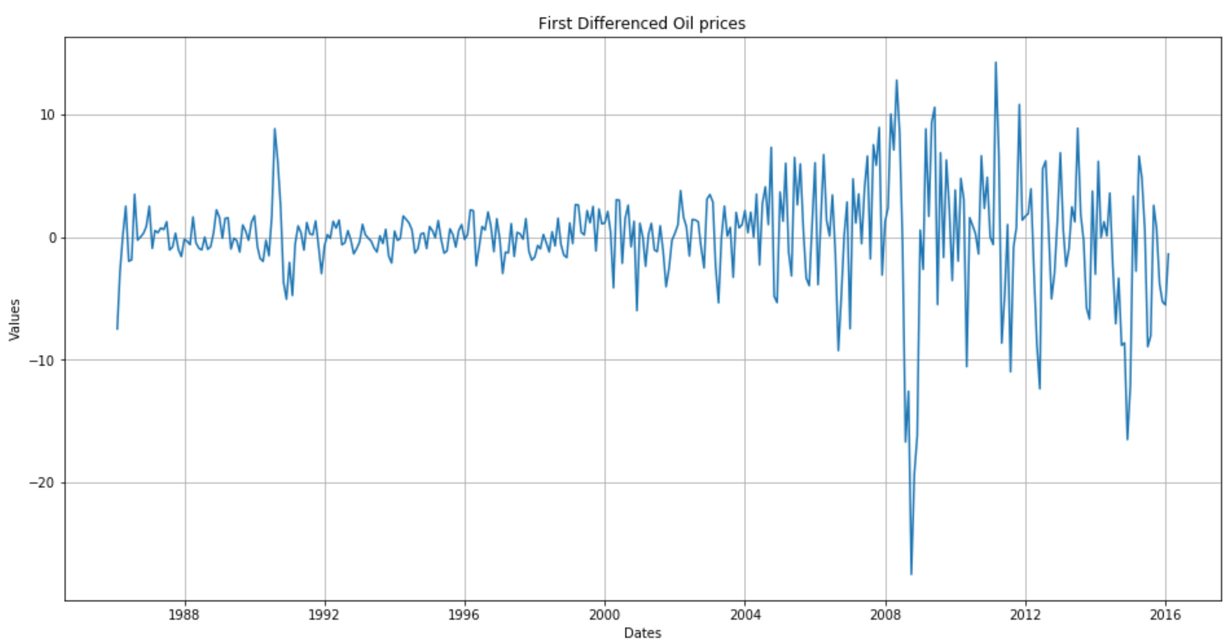
Out[194]:

	VALUE
DATE	
1986-01-01	22.93
1986-02-01	15.46
1986-03-01	12.61
1986-04-01	12.84
1986-05-01	15.38

3.1

In [195]:
```python
# Time plot of the oil prices
plt.figure(figsize=(16,8))
plt.grid(True)
plt.xlabel('Dates')
plt.ylabel('Values')
plt.title('Oil prices')
plt.plot(oil_prices)
plt.show()
```

In [196]:
```python
# Plot the first differenced oil prices
oil_prices_diff = oil_prices.diff().dropna()
plt.figure(figsize=(16,8))
plt.grid(True)
plt.xlabel('Dates')
plt.ylabel('Values')
plt.title('First Differenced Oil prices')
plt.plot(oil_prices_diff)
plt.show()
```



3.2

Yes. After the differencing, we removed the trend effect. Now it looks like that the series are centered around 0 and have equal variances over the time.

3.3

In [197]:
```
print("ADF test results for Oil prices: ")
print("Test stats: ", sm.tsa.stattools.adfuller(oil_prices)[0])
print("P-value: ", sm.tsa.stattools.adfuller(oil_prices)[1])
print("Critical values: ", sm.tsa.stattools.adfuller(oil_prices)[4])
```

```
ADF test results for Oil prices:
Test stats:  -1.6287276101798709
P-value:  0.46815680296168605
Critical values:  {'1%': -3.448905534655263, '5%': -2.8697161816205705, '10%': -
2.5711258103550882}
```

P-value is insignificant so we don't reject the null hypothesis that unit root exists. This time series is not stationary.

In [198]:
```
print("ADF test results for First Differenced Oil prices: ")
print("Test stats: ", sm.tsa.stattools.adfuller(oil_prices_diff)[0])
print("P-value: ", sm.tsa.stattools.adfuller(oil_prices_diff)[1])
print("Critical values: ", sm.tsa.stattools.adfuller(oil_prices_diff)[4])
```

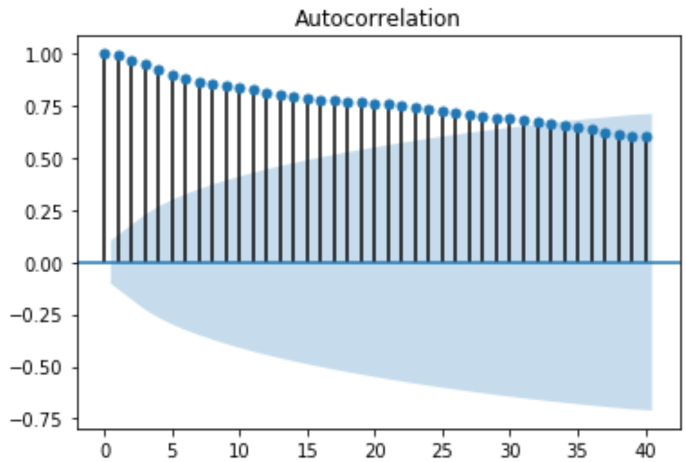
```
ADF test results for First Differenced Oil prices:
Test stats:  -9.306964876774327
P-value:  1.0742996246722798e-15
Critical values:  {'1%': -3.448905534655263, '5%': -2.8697161816205705, '10%': -
2.5711258103550882}
```

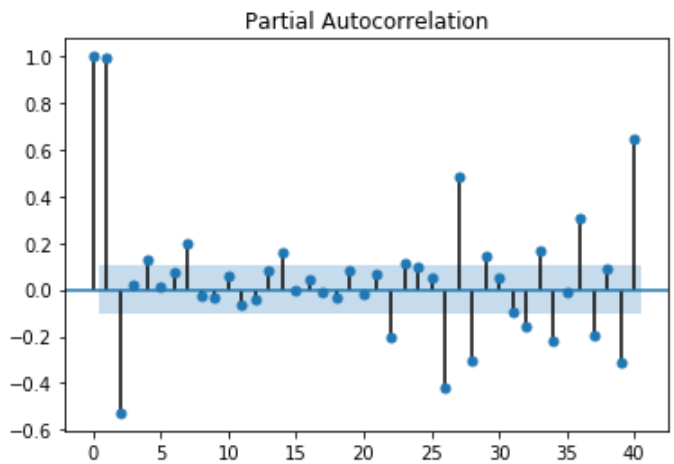
P-value is significant so we reject the null hypothesis and there shouldn't be any unit root. This first differenced time series is stationary.

3.4

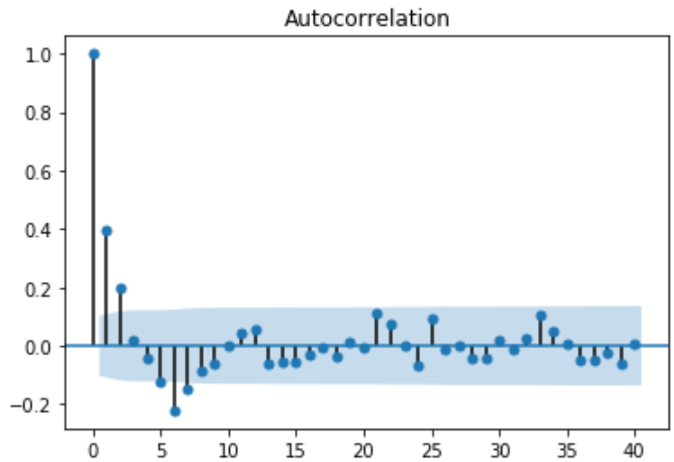
In [199]:
```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

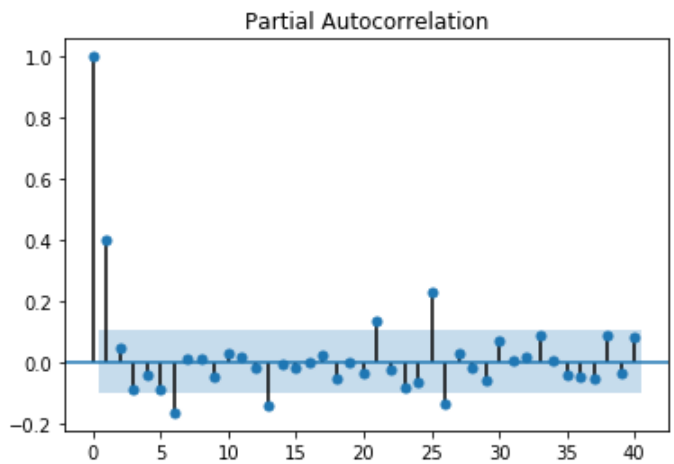
In [200]:
```
# for oil prices
acf_plot = plot_acf(oil_prices, lags=40, alpha=0.05)
pacf_plot = plot_pacf(oil_prices, lags=40, alpha=0.05)
```





In [145]:
```
# for differenced oil prices
acf_plot = plot_acf(oil_prices_diff, lags=40, alpha=0.05)
pacf_plot = plot_pacf(oil_prices_diff, lags=40, alpha=0.05)
```





Comments:

-Acf and PACF of oil prices are not decaying, indicating that the series is highly correlated and need further differnecing.

-PACF of the differenced oil prices has a clear cutoff(may have long run dependency). ACF of the differenced oil prices is decaying. We could possibly fit an ARIMA model for this series(although the residuals may have GARCH behavior)

3.5

In [201]:
```
sm.stats.diagnostic.acorr_ljungbox(oil_prices_diff, lags=12, boxpierce=False, retu
rn_df=True)
```

Out[201]:

	lb_stat	lb_pvalue
1	57.465099	3.440301e-14
2	71.799243	2.564439e-16
3	71.938271	1.641115e-15
4	72.675068	6.179524e-15
5	78.042684	2.153518e-15
6	96.139609	1.600767e-18
7	104.269344	1.413704e-19
8	106.980728	1.587383e-19
9	108.373236	3.151081e-19
10	108.374056	1.134711e-18
11	109.006754	2.892063e-18
12	110.110570	5.690184e-18

Conclusion: All pvalues are very close to 0. So we reject the null hypothesis and take the alternative one: the data are not independently distributed.

3.6

In [202]:
```
# PACF indicates a AR(6) process
model = statsmodels.tsa.arima_model.ARIMA(oil_prices_diff, order=(6,0,0))
model_fit = model.fit(disp=0)
print(model_fit.summary())
```

```
/Users/chamusyuan/anaconda3/lib/python3.7/site-packages/statsmodels/tsa/base/tsa_
model.py:162: ValueWarning: No frequency information was provided, so inferred fr
equency MS will be used.
  % freq, ValueWarning)

                              ARMA Model Results                              
==============================================================================
Dep. Variable:                  VALUE   No. Observations:                  361
Model:                     ARMA(6, 0)   Log Likelihood               -1000.700
Method:                       css-mle   S.D. of innovations              3.867
Date:                Wed, 03 Feb 2021   AIC                           2017.401
Time:                        00:25:26   BIC                           2048.512
Sample:                    02-01-1986   HQIC                          2029.770
                         - 02-01-2016                                         
===============================================================================
                  coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------
const           0.0261      0.248      0.105      0.916      -0.461       0.513
ar.L1.VALUE     0.3621      0.052      6.960      0.000       0.260       0.464
ar.L2.VALUE     0.0779      0.055      1.407      0.159      -0.031       0.186
ar.L3.VALUE    -0.0761      0.056     -1.369      0.171      -0.185       0.033
ar.L4.VALUE     0.0059      0.056      0.106      0.915      -0.103       0.115
ar.L5.VALUE    -0.0291      0.055     -0.526      0.599      -0.138       0.079
ar.L6.VALUE    -0.1626      0.052     -3.127      0.002      -0.265      -0.061
                                    Roots                                    
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.0758           -0.5649j            1.2151           -0.0770
AR.2            1.0758           +0.5649j            1.2151            0.0770
AR.3            0.1138           -1.3993j            1.4039           -0.2371
AR.4            0.1138           +1.3993j            1.4039            0.2371
AR.5           -1.2791           -0.6906j            1.4536           -0.4212
AR.6           -1.2791           +0.6906j            1.4536            0.4212
-----------------------------------------------------------------------------
```

Fitted model:

$x_t = 0.3621x_{t-1} - 0.1626x_{t-6}$

3.7

In [203]:
```
mod = sm.tsa.statespace.SARIMAX(oil_prices_diff, trend='c', order=(1,0,6))
res = mod.fit(disp=False)
print(res.summary())
```

```
/Users/chamusyuan/anaconda3/lib/python3.7/site-packages/statsmodels/tsa/base/tsa_
model.py:162: ValueWarning: No frequency information was provided, so inferred fr
equency MS will be used.
  % freq, ValueWarning)
/Users/chamusyuan/anaconda3/lib/python3.7/site-packages/statsmodels/tsa/base/tsa_
model.py:162: ValueWarning: No frequency information was provided, so inferred fr
equency MS will be used.
  % freq, ValueWarning)

                               SARIMAX Results
==============================================================================
Dep. Variable:                  VALUE   No. Observations:                  361
Model:               SARIMAX(1, 0, 6)   Log Likelihood               -1000.138
Date:                Wed, 03 Feb 2021   AIC                           2018.276
Time:                        00:25:44   BIC                           2053.276
Sample:                    02-01-1986   HQIC                          2032.192
                         - 02-01-2016
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0186      0.065      0.287      0.774      -0.108       0.146
ar.L1          0.6892      0.107      6.437      0.000       0.479       0.899
ma.L1         -0.3331      0.112     -2.962      0.003      -0.553      -0.113
ma.L2         -0.0384      0.057     -0.670      0.503      -0.151       0.074
ma.L3         -0.1290      0.046     -2.820      0.005      -0.219      -0.039
ma.L4         -0.0149      0.047     -0.316      0.752      -0.107       0.077
ma.L5         -0.0312      0.038     -0.825      0.409      -0.105       0.043
ma.L6         -0.1717      0.044     -3.891      0.000      -0.258      -0.085
sigma2        14.9058      0.758     19.665      0.000      13.420      16.391
================================================================================
==
Ljung-Box (Q):                       46.75   Jarque-Bera (JB):                195.
30
Prob(Q):                              0.21   Prob(JB):                          0.
00
Heteroskedasticity (H):              13.29   Skew:                             -0.
49
Prob(H) (two-sided):                  0.00   Kurtosis:                          6.
47
================================================================================
==

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-st
ep).
```

Fitted model:

$$x_t = 0.0186 + 0.6892x_{t-1} + w_t - 0.3331w_{t-1} - 0.0384w_{t-2} - 0.1290w_{t-3} - 0.0149w_{t-4} - 0.0312w_{t-5} - 0.1717w_{t-6}$$

3.8

In [204]:
```
predict = res.get_prediction(start=361, end=364)
predict_val = predict.predicted_mean
predict_ci = predict.conf_int()
```

In [205]:
```
predict_val
```

Out[205]:
```
2016-03-01   -0.444887
2016-04-01    0.800528
2016-05-01    1.803406
2016-06-01    2.322258
Freq: MS, dtype: float64
```

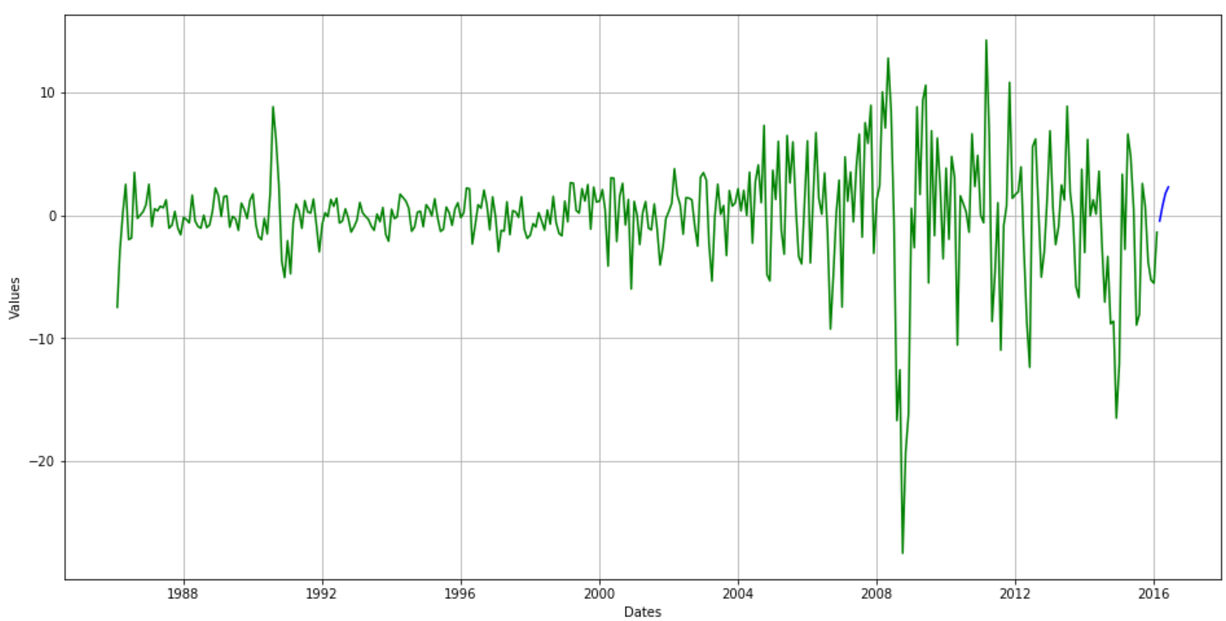
In [206]:
```
predict_ci
```

Out[206]:

	lower VALUE	upper VALUE
2016-03-01	-8.011916	7.122143
2016-04-01	-7.232006	8.833061
2016-05-01	-6.380489	9.987301
2016-06-01	-5.862297	10.506814

In [208]:
```
plt.figure(figsize=(16,8))
plt.grid(True)
plt.xlabel('Dates')
plt.ylabel('Values')
plt.plot(oil_prices_diff.index, oil_prices_diff, 'green')
plt.plot(predict_val.index, predict_val, 'blue')
```

Out[208]: [<matplotlib.lines.Line2D at 0x7fbdfb815320>]



Question 4 Garch

In [154]:
```python
# Read in dataset
RawData = []
with open('d-amzn3dx0914.txt','r') as file:
    for line in file:
        RawData.append(line.split())

DATA = []
for i in range(1, len(RawData)):
    DATA.append([pd.to_datetime(RawData[i][1]), 100*float(RawData[i][2])])

amzn = pd.DataFrame(DATA, columns=['DATE', 'AMZN'])
amzn = amzn.set_index(['DATE'])
amzn.head()
```

Out[154]:

DATE	AMZN
2009-01-02	6.0062
2009-01-05	-0.5519
2009-01-06	6.1043
2009-01-07	-2.0223
2009-01-08	1.7082

4.1

In [215]:
```python
from scipy import stats

ret = amzn['AMZN']
print('Sample mean in percentage:', ret.mean())
result = stats.ttest_1samp(ret, 0)
print('Test stats:', result.statistic)
print('Pvalue:', result.pvalue)
```

```
Sample mean in percentage: 0.14555754966887402
Test stats: 2.4453838553781244
Pvalue: 0.01458343172407275
```

Performed a t-test for the mean. The p value is less than 0.05, so we reject the null hypothesis that the mean of r_t is 0. Also in economic sense, since AMZN is a company growing steadily, the log return of its stock should be positive.

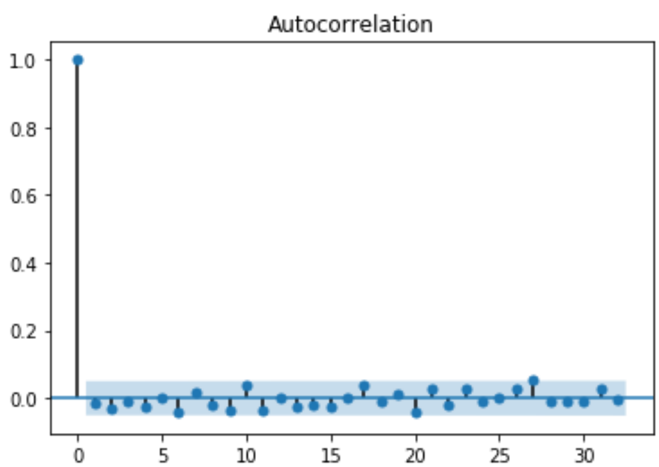
4.2

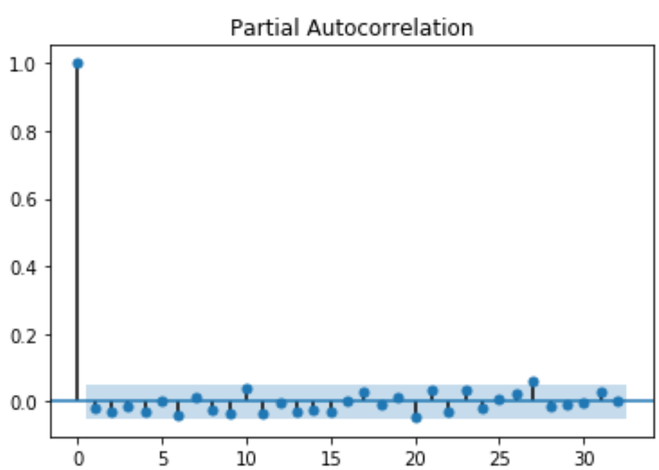
In [217]:
```
amzn_acf = plot_acf(ret)
amzn_pacf = plot_pacf(ret)

sm.stats.diagnostic.acorr_ljungbox(ret, lags=12, boxpierce=False, return_df=True)
```

Out[217]:

	lb_stat	lb_pvalue
1	0.410377	0.521778
2	1.750405	0.416778
3	1.914770	0.590284
4	2.980625	0.561073
5	2.994831	0.700783
6	5.384310	0.495548
7	5.808027	0.562336
8	6.478638	0.593778
9	8.222523	0.511886
10	10.778525	0.375031
11	12.481263	0.328576
12	12.486833	0.407415





All ljungbox test pvalues are larger than 0.05. So we don't reject the null hypothesis and claim that there isn't serial correlation in r_t. Also ACF and PACF plots indicates little autocorrelation among the series.

4.3

In [218]:
```python
import arch

am = arch.arch_model(ret, vol='Garch', p=1, q=1)
am_fit = am.fit()
print(am_fit.summary())
```

```
Iteration:      1,   Func. Count:      6,   Neg. LLF: 3311134346.2968283
Iteration:      2,   Func. Count:     14,   Neg. LLF: 8336.16977026914
Iteration:      3,   Func. Count:     23,   Neg. LLF: 5537.6163832879665
Iteration:      4,   Func. Count:     29,   Neg. LLF: 5233.853676693495
Iteration:      5,   Func. Count:     35,   Neg. LLF: 3365.208786790381
Iteration:      6,   Func. Count:     40,   Neg. LLF: 4216.655253487106
Iteration:      7,   Func. Count:     46,   Neg. LLF: 15100.272744751033
Iteration:      8,   Func. Count:     55,   Neg. LLF: 3352.592123976801
Iteration:      9,   Func. Count:     60,   Neg. LLF: 3352.2697686409656
Iteration:     10,   Func. Count:     65,   Neg. LLF: 3352.2029036055537
Iteration:     11,   Func. Count:     70,   Neg. LLF: 3352.2004235479612
Iteration:     12,   Func. Count:     75,   Neg. LLF: 3352.2002812714763
Iteration:     13,   Func. Count:     80,   Neg. LLF: 3352.2002763041974
Iteration:     14,   Func. Count:     85,   Neg. LLF: 3352.2002757157843
Optimization terminated successfully    (Exit mode 0)
            Current function value: 3352.2002757157843
            Iterations: 14
            Function evaluations: 85
            Gradient evaluations: 14
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:                   AMZN   R-squared:                      -0.000
Mean Model:             Constant Mean   Adj. R-squared:                 -0.000
Vol Model:                      GARCH   Log-Likelihood:               -3352.20
Distribution:                  Normal   AIC:                           6712.40
Method:            Maximum Likelihood   BIC:                           6733.68
                                        No. Observations:                 1510
Date:                Wed, Feb 03 2021   Df Residuals:                     1506
Time:                        00:41:16   Df Model:                            4
                                 Mean Model
===========================================================================
                 coef    std err          t      P>|t|     95.0% Conf. Int.
---------------------------------------------------------------------------
mu             0.1156  5.562e-02      2.079  3.766e-02 [6.593e-03,  0.225]
                              Volatility Model
============================================================================
                 coef    std err          t      P>|t|        95.0% Conf. Int.
----------------------------------------------------------------------------
omega          0.0324  1.943e-02      1.665  9.596e-02 [-5.736e-03,7.044e-02]
alpha[1]       0.0000  4.172e-03      0.000      1.000 [-8.178e-03,8.178e-03]
beta[1]        0.9921  7.352e-03    134.939      0.000     [  0.978,  1.007]
============================================================================

Covariance estimator: robust
```

ARMA-GARCH Model

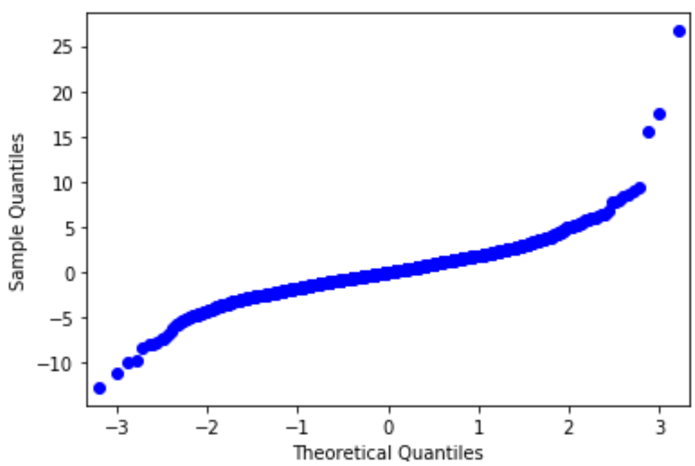
$r_t = 0.1156 + \epsilon_t$

$\epsilon_t = \sigma_t \cdot e_t$

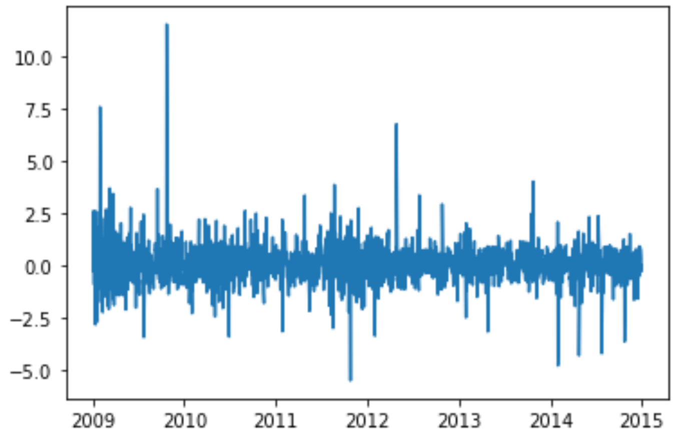
${\sigma_t}^2 = 0.0324 + 0.9921{\sigma_{t-1}}^2$

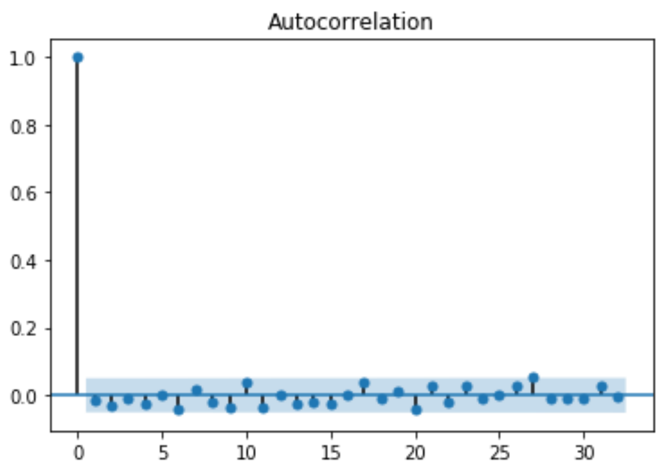
$e_t \sim N(0, 1)$

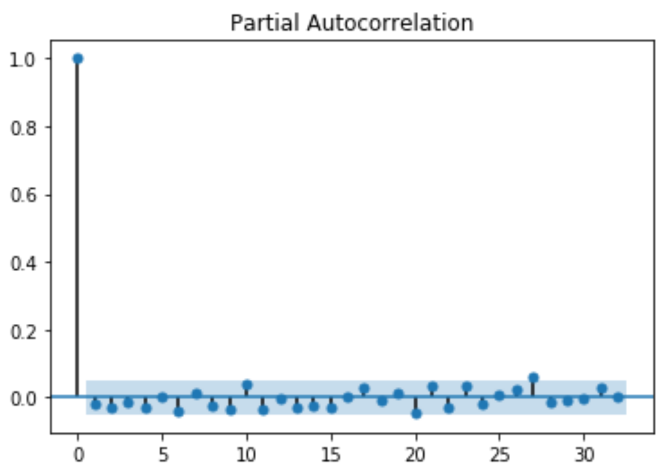
In [221]:
```
res = am_fit.resid
plt.plot(res)
qq_plot = statsmodels.api.qqplot(res)
```



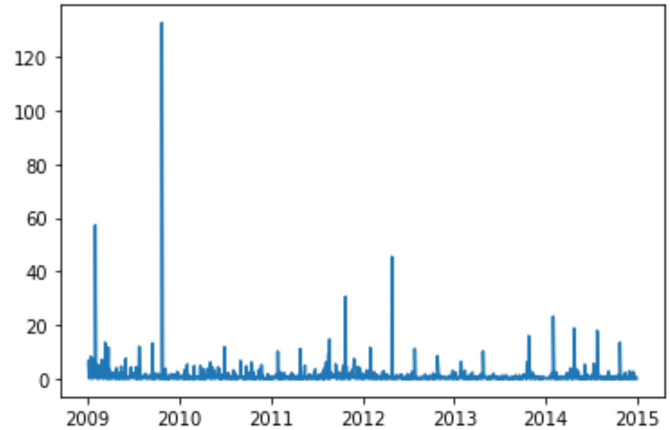
In [227]:
```
standardized_res = (res-res.mean())/res.std()
plt.plot(standardized_res)
res_std_acf = plot_acf(standardized_res)
res_std_pacf = plot_pacf(standardized_res)
```

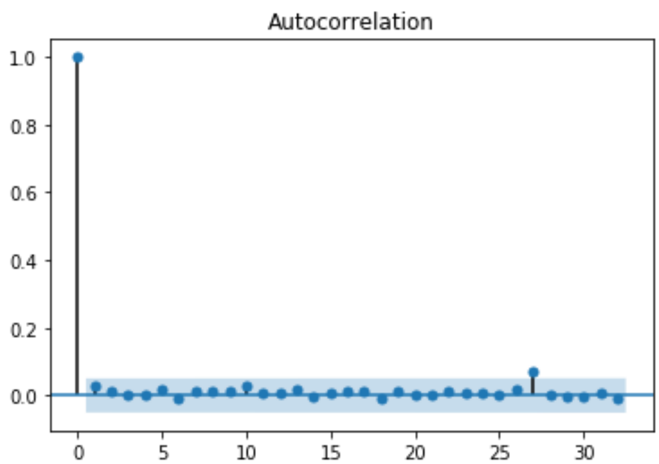


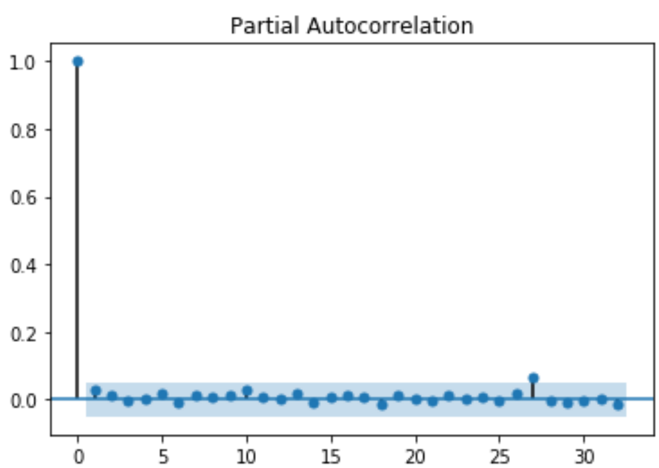




In [228]:
```
sqr_standardized_res = standardized_res**2
plt.plot(sqr_standardized_res)
sqr_res_std_acf = plot_acf(sqr_standardized_res)
sqr_res_std_pacf = plot_pacf(sqr_standardized_res)
```







Comment: this model is adequate. Residuals look stationary from ACF/PACF, but have a fat-tail problem from qq plots.

4.4

In [229]:
```
am = arch.arch_model(ret, vol='Garch', p=1, q=1, dist='StudentsT')
am_fit = am.fit()
print(am_fit.summary())
```

```
Iteration:      1,   Func. Count:      7,   Neg. LLF: 31604.212731429856
Iteration:      2,   Func. Count:     16,   Neg. LLF: 77794.45888614171
Iteration:      3,   Func. Count:     25,   Neg. LLF: 4604.45450338835
Iteration:      4,   Func. Count:     34,   Neg. LLF: 3223.105100448301
Iteration:      5,   Func. Count:     41,   Neg. LLF: 3370.1167425622866
Iteration:      6,   Func. Count:     48,   Neg. LLF: 3237.823876224604
Iteration:      7,   Func. Count:     55,   Neg. LLF: 3229.066898542713
Iteration:      8,   Func. Count:     62,   Neg. LLF: 3194.2651409936775
Iteration:      9,   Func. Count:     69,   Neg. LLF: 3188.7080259264235
Iteration:     10,   Func. Count:     76,   Neg. LLF: 3188.2301559514735
Iteration:     11,   Func. Count:     82,   Neg. LLF: 3188.2281453927335
Iteration:     12,   Func. Count:     88,   Neg. LLF: 3188.2282454125893
Iteration:     13,   Func. Count:     95,   Neg. LLF: 3188.2277420391356
Iteration:     14,   Func. Count:    101,   Neg. LLF: 3188.2277308096536
Iteration:     15,   Func. Count:    106,   Neg. LLF: 3188.2277308096473
Optimization terminated successfully    (Exit mode 0)
            Current function value: 3188.2277308096536
            Iterations: 15
            Function evaluations: 106
            Gradient evaluations: 15
                        Constant Mean - GARCH Model Results
==================================================================================
===
Dep. Variable:                         AMZN   R-squared:                       -0.
000
Mean Model:                   Constant Mean   Adj. R-squared:                  -0.
000
Vol Model:                            GARCH   Log-Likelihood:                 -318
8.23
Distribution:      Standardized Student's t   AIC:                             638
6.46
Method:                  Maximum Likelihood   BIC:                             641
3.05
                                              No. Observations:                  1
510
Date:                      Wed, Feb 03 2021   Df Residuals:                      1
505
Time:                              00:51:48   Df Model:
5
                                Mean Model
===========================================================================
                 coef    std err          t      P>|t|    95.0% Conf. Int.
---------------------------------------------------------------------------
mu             0.0945  4.662e-02      2.026  4.272e-02 [3.097e-03,  0.186]
                              Volatility Model
=============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
-----------------------------------------------------------------------------
omega          0.0518  1.473e-02      3.514  4.419e-04 [2.289e-02,8.062e-02]
alpha[1]   6.8702e-03  3.017e-03      2.277  2.276e-02 [9.577e-04,1.278e-02]
beta[1]        0.9796  5.155e-03    190.022      0.000     [  0.969,  0.990]
                              Distribution
=========================================================================
                 coef    std err          t      P>|t|  95.0% Conf. Int.
-------------------------------------------------------------------------
nu             4.3514      0.496      8.781  1.624e-18 [  3.380,  5.323]
=========================================================================

Covariance estimator: robust
```

4.5

ARMA-GARCH Model

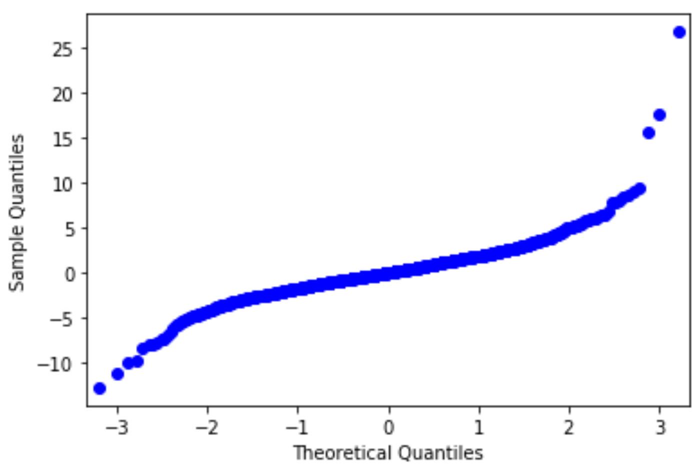
$r_t = 0.0945 + \epsilon_t$

$\epsilon_t = \sigma_t \cdot e_t$

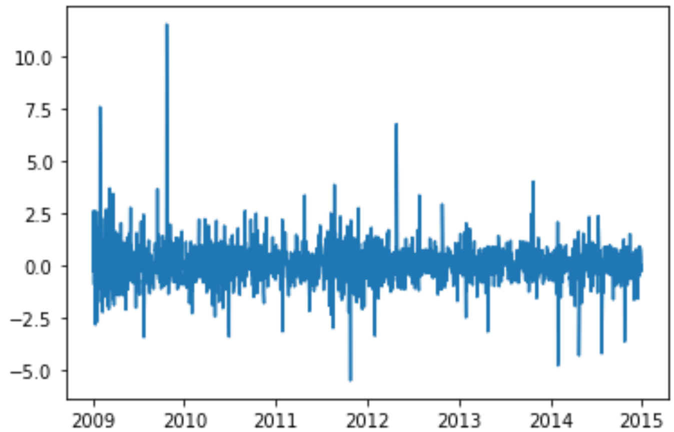
${\sigma_t}^2 = 0.0518 + 0.9796{\sigma_{t-1}}^2$

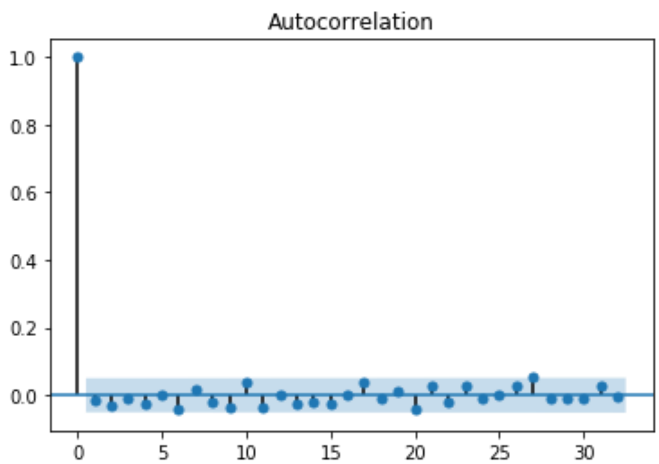
$e_t \sim T(0;4)$

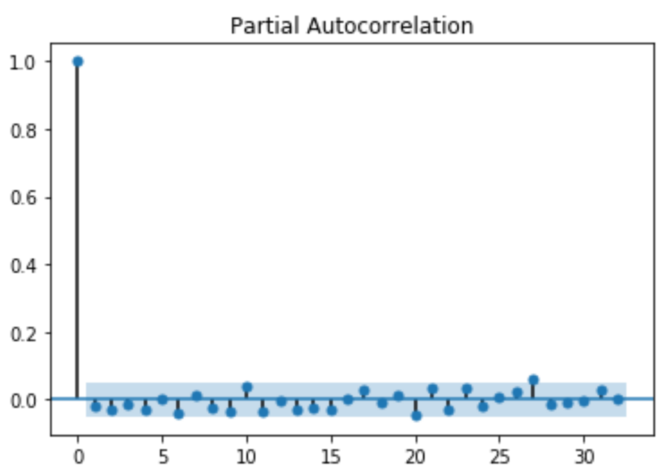
In [230]:
```
res = am_fit.resid
plt.plot(res)
qq_plot = statsmodels.api.qqplot(res)
```



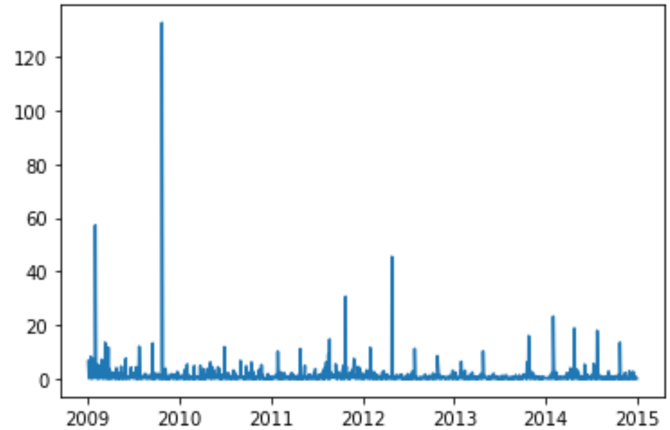
In [231]:
```
standardized_res = (res-res.mean())/res.std()
plt.plot(standardized_res)
res_std_acf = plot_acf(standardized_res)
res_std_pacf = plot_pacf(standardized_res)
```

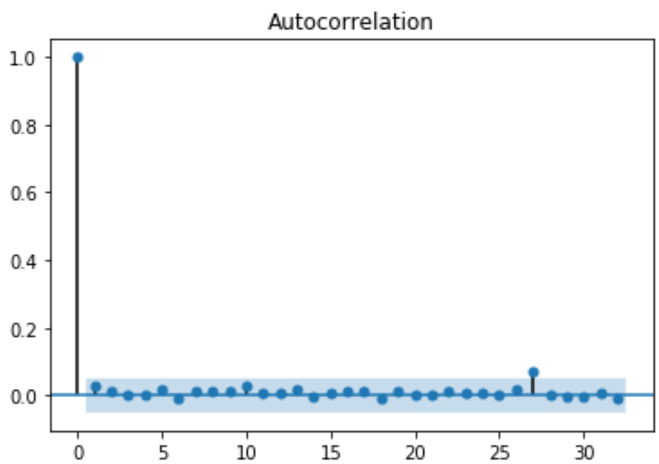


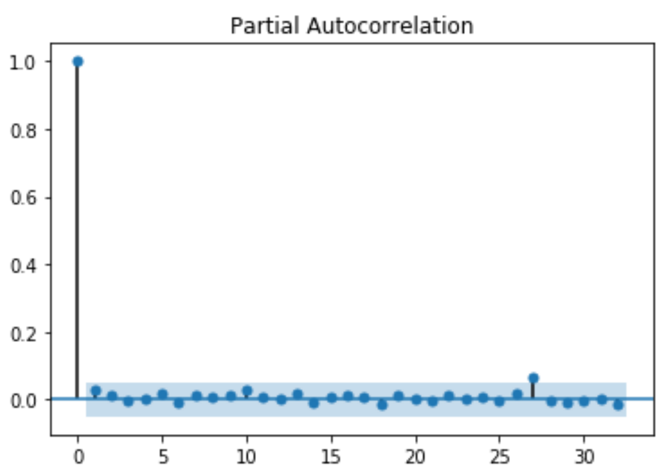




In [232]:
```
sqr_standardized_res = standardized_res**2
plt.plot(sqr_standardized_res)
sqr_res_std_acf = plot_acf(sqr_standardized_res)
sqr_res_std_pacf = plot_pacf(sqr_standardized_res)
```







Comment: this model is adequate. Residuals look stationary from ACF/PACF, but still have a fat-tail problem from qq plots.

4.6

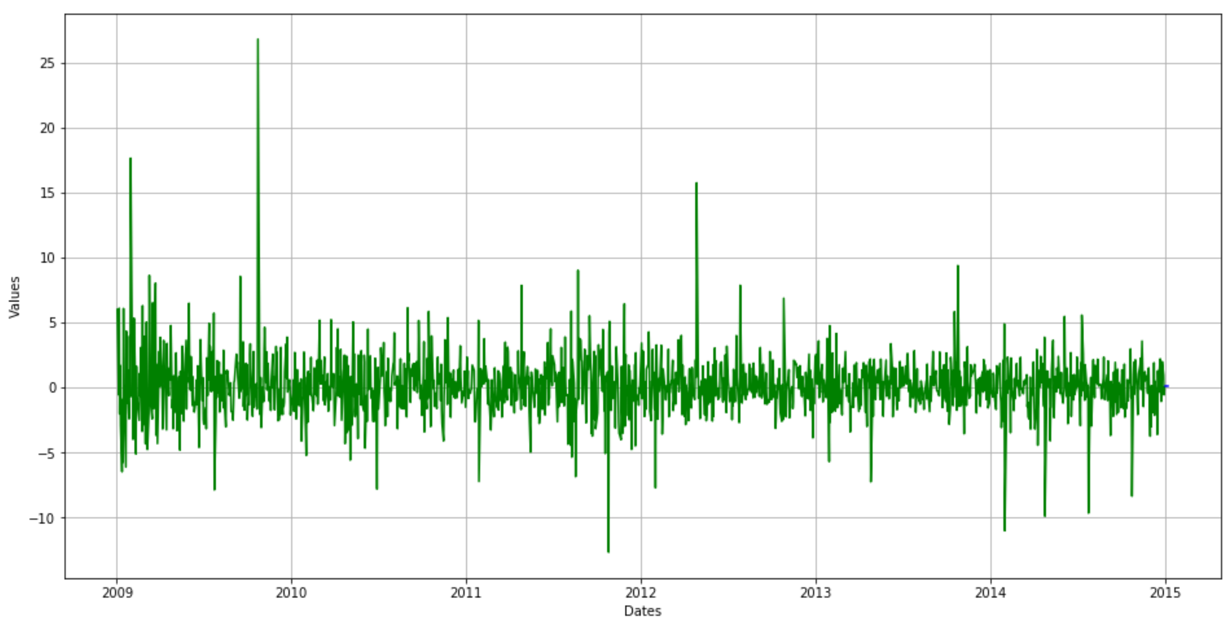
In [233]:
```
forecast = am_fit.forecast(horizon=5, align='origin')
print(forecast.mean.dropna())
print(forecast.variance.dropna())
```

```
                 h.1       h.2       h.3       h.4       h.5
DATE
2014-12-31  0.094468  0.094468  0.094468  0.094468  0.094468
                 h.1       h.2      h.3       h.4       h.5
DATE
2014-12-31  3.594059  3.597115  3.60013  3.603104  3.606037
```

In [234]:
```
idx = ['2015-01-02', '2015-01-05', '2015-01-06', '2015-01-07', '2015-01-08']
idx = [pd.to_datetime(_) for _ in idx]
predict = pd.DataFrame(list(forecast.mean.iloc[-1,:]), index=idx, columns=['Value
s'])
```

In [235]:
```
plt.figure(figsize=(16,8))
plt.grid(True)
plt.xlabel('Dates')
plt.ylabel('Values')
plt.plot(ret.index, ret, 'green', label='Train data')
plt.plot(predict.index, predict, 'blue', label='Test data')
```

Out[235]: [<matplotlib.lines.Line2D at 0x7fbde0d00cc0>]



Comments: Returns are not predictable for short horizons. Judging from our predictions, the forcasted values are very close and have similar variance and it's actually quite random. Although we introduced student-t innovations, we still can't solve the fat-tail problem of the residuals, which means our Gaussian/T-distributed assumption about them could be challenged. Generally speaking, short terms returns are just combinations of market uncertainties at that moment, thus are not predictable.

In [ ]: