# EGARCH-SP

## 1 Importing packages

[ ]:

```

```

[1]:

```
#importing packages
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
from sklearn import linear_model
import matplotlib.pyplot as plt
from scipy import stats
import datetime
```

[ ]:

```

```

## 2 Reading Excel file saved in hard drive

[2]:

```
#reading the file
df = pd.read_excel("C:\\Users\\rluck\\OneDrive\\shares.xlsx")
df.head()
```

[2]:

```
        Date       Price
0 1998-01-02  975.039978
1 1998-01-05  977.070007
2 1998-01-06  966.580017
3 1998-01-07  964.000000
4 1998-01-08  956.049988
```

## 3 Calculating annual return

```
[3]: #computing the annual return from S&P500
     df['R'] = 100*np.log(df['Price']/df['Price'].shift(1))
     df.head()
```

```
[3]:          Date       Price         R
     0 1998-01-02  975.039978       NaN
     1 1998-01-05  977.070007  0.207983
     2 1998-01-06  966.580017 -1.079422
     3 1998-01-07  964.000000 -0.267279
     4 1998-01-08  956.049988 -0.828109
```

```
[4]: df.tail(10)
```

```
[4]:            Date        Price         R
     984 2001-12-03  1129.900024 -0.841649
     985 2001-12-04  1144.800049  1.310084
     986 2001-12-05  1170.349976  2.207284
     987 2001-12-06  1167.099976 -0.278981
     988 2001-12-07  1158.310059 -0.755992
     989 2001-12-10  1139.930054 -1.599519
     990 2001-12-11  1136.760010 -0.278478
     991 2001-12-12  1137.069946  0.027261
     992 2001-12-13  1119.380005 -1.567977
     993 2001-12-14  1123.089966  0.330882
```

## 4 Remove the first row Nan

```
[5]: #Selecting the sample from
     dta =df.iloc[1:900]
     dta.head()
```

```
[5]:          Date       Price         R
     1 1998-01-05  977.070007  0.207983
     2 1998-01-06  966.580017 -1.079422
     3 1998-01-07  964.000000 -0.267279
     4 1998-01-08  956.049988 -0.828109
     5 1998-01-09  927.690002 -3.011257
```

```
[6]: dta.tail()
```

```
[6]:            Date        Price         R
     895 2001-07-23  1191.030029 -1.650407
     896 2001-07-24  1171.650024 -1.640547
     897 2001-07-25  1190.489990  1.595195
```

```
898 2001-07-26  1202.930054  1.039531
899 2001-07-27  1205.819946  0.239950
```

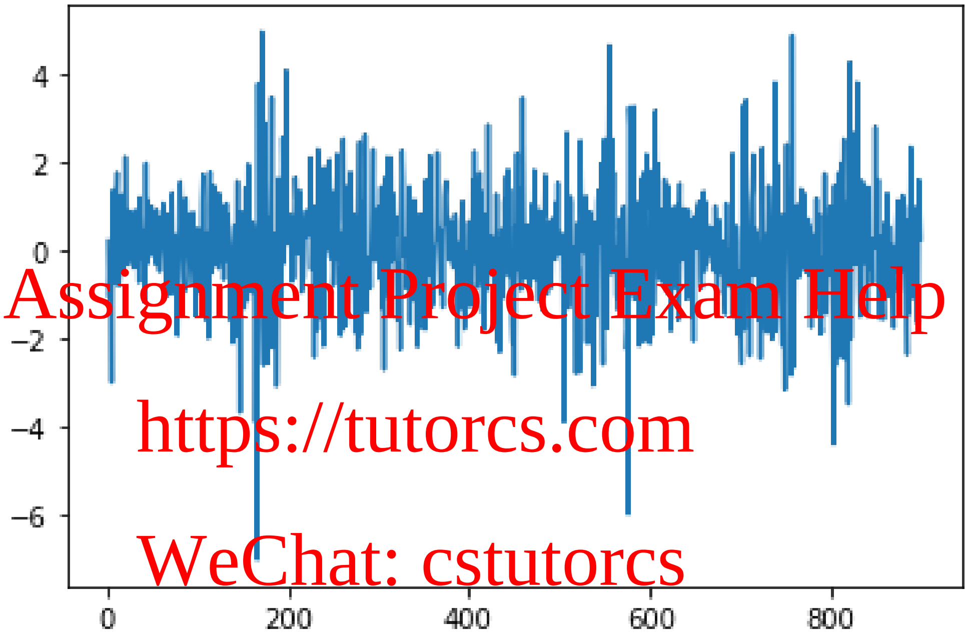
## 5 Plotting the time series: Stock Returns (R)

[7]:

```
#plotting the series
plt.plot(dta["R"])
```

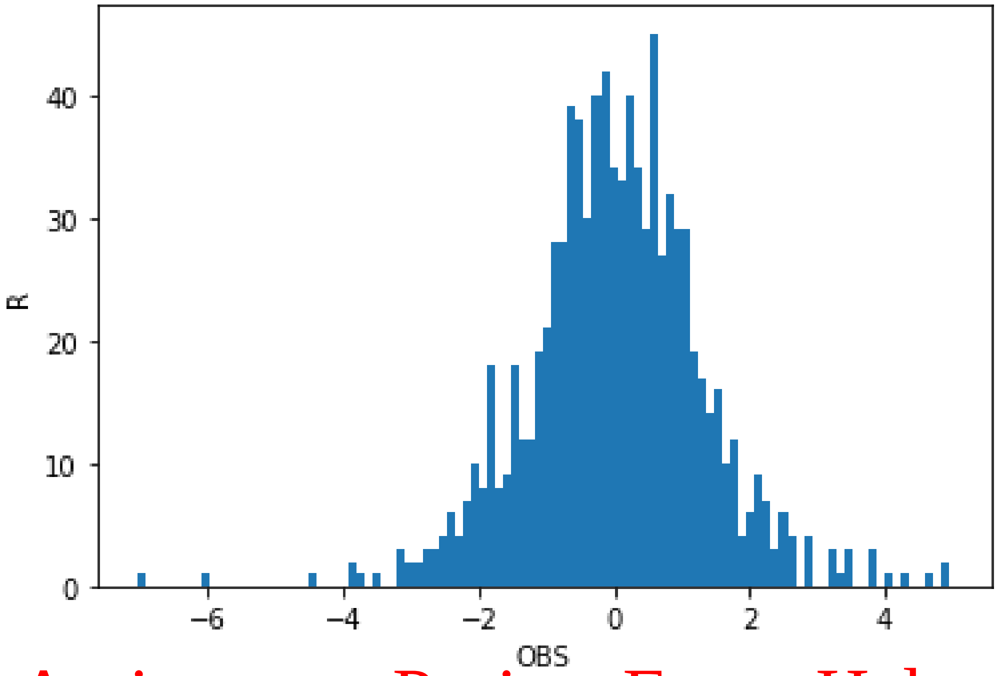
[7]:

```
[<matplotlib.lines.Line2D at 0x26ba1f5ef08>]
```



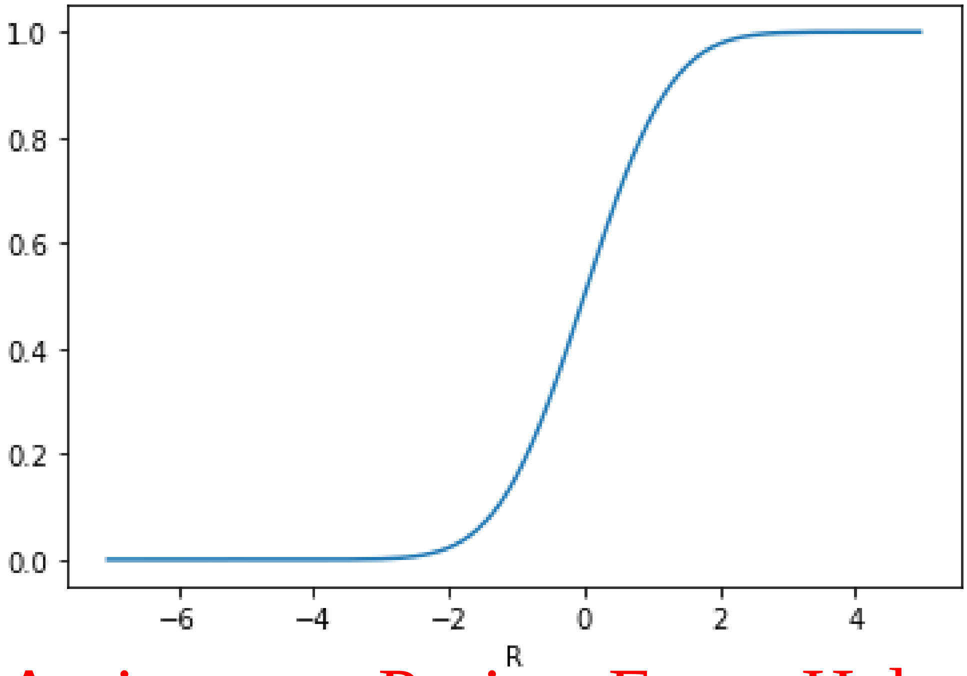
[8]:

```
import matplotlib.pyplot as plt
_ = plt.hist(dta['R'],bins=100)
_ = plt.xlabel('OBS')
_ = plt.ylabel('R')
plt.show()
```



## 6 Q4(a) CDF & 1% quantile

[9]:
```
import numpy as np
import scipy
import matplotlib.pyplot as plt
import seaborn as sns
dta=dta['R']
# generate samples from normal distribution (discrete data)
norm_cdf = scipy.stats.norm.cdf(dta) # calculate the cdf - also discrete

# plot the cdf
sns.lineplot(x=dta, y=norm_cdf)
plt.show()
```



[10]:
```
#lower 1% quantile
np.percentile(dta,1)
```

[10]:
```
-3.0849950142334084
```

## 7 Q4b-c:GARCH(1,1), GJR and EGARCH

[11]:
```
from arch import arch_model
```

## 8 GARCH(1,1)

[12]:
```
#GARCH(1,1)
model = arch_model(dta, mean='constant', vol='GARCH', p=1, q=1)
res_1 =model.fit()
res_1.summary
```

```
Iteration:      1,   Func. Count:      6,   Neg. LLF: 1478.37238353009
Iteration:      2,   Func. Count:     15,   Neg. LLF: 1477.9764253323951
Iteration:      3,   Func. Count:     26,   Neg. LLF: 1477.970683730851
Iteration:      4,   Func. Count:     34,   Neg. LLF: 1476.571267302416
Iteration:      5,   Func. Count:     43,   Neg. LLF: 1476.10266273572
Iteration:      6,   Func. Count:     49,   Neg. LLF: 1475.8308574534146
Iteration:      7,   Func. Count:     56,   Neg. LLF: 1475.682907378089
```

```
Iteration:      8,   Func. Count:     62,   Neg. LLF: 1475.6514266378354
Iteration:      9,   Func. Count:     68,   Neg. LLF: 1475.6494614480196
Iteration:     10,   Func. Count:     74,   Neg. LLF: 1475.6487752742607
Iteration:     11,   Func. Count:     80,   Neg. LLF: 1475.6484969528722
Iteration:     12,   Func. Count:     86,   Neg. LLF: 1475.6484950954127
Optimization terminated successfully.    (Exit mode 0)
            Current function value: 1475.6484950951449
            Iterations: 12
            Function evaluations: 86
            Gradient evaluations: 12
```

[12]:

```
<bound method ARCHModelResult.summary of                      Constant Mean -
GARCH Model Results
==============================================================================
Dep. Variable:                      R   R-squared:                      -0.000
Mean Model:             Constant Mean   Adj. R-squared:                 -0.000
Vol Model:                      GARCH   Log-Likelihood:               -1475.65
Distribution:                  Normal   AIC:                           2959.30
Method:            Maximum Likelihood   BIC:                           2978.50
                                        No. Observations:                  899
Date:                Thu, Aug 06 2020   Df Residuals:                      895
Time:                        20:19:12   Df Model:                            4
                                 Mean Model
============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
----------------------------------------------------------------------------
mu             0.0503  4.034e-02      1.247      0.212 [-2.876e-02,  0.129]
                              Volatility Model
============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
----------------------------------------------------------------------------
omega          0.0685  3.987e-02      1.718  8.574e-02 [-9.636e-03,  0.147]
alpha[1]       0.0875  3.351e-02      2.610  9.052e-03  [2.179e-02,  0.153]
beta[1]        0.8739  4.556e-02     19.183  5.140e-82    [  0.785,  0.963]
============================================================================

Covariance estimator: robust
ARCHModelResult, id: 0x26ba2d35988>
```

## 9 GJR

[13]:

```python
from arch.univariate import EGARCH
resi = arch_model(dta, mean ='constant',vol='GARCH', p=1,o=1, q=1)
resi = resi.fit(update_freq=5, disp='off')
resi
```

[13]:

```
                   Constant Mean - GJR-GARCH Model Results
==============================================================================
Dep. Variable:                      R   R-squared:                      -0.000
Mean Model:             Constant Mean   Adj. R-squared:                 -0.000
Vol Model:                  GJR-GARCH   Log-Likelihood:               -1447.88
Distribution:                  Normal   AIC:                           2905.76
Method:            Maximum Likelihood   BIC:                           2929.76
                                        No. Observations:                  899
Date:                Thu, Aug 06 2020   Df Residuals:                      894
Time:                        20:19:12   Df Model:                            5
                                  Mean Model
==============================================================================
                  coef    std err          t      P>|t|       95.0% Conf. Int.
------------------------------------------------------------------------------
mu         -6.0831e-04  4.394e-02 -1.384e-02      0.989 [-8.673e-02,8.551e-02]
                              Volatility Model
===========================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
---------------------------------------------------------------------------
omega          0.0809  5.934e-02      1.363      0.173 [-3.540e-02, 0.197]
alpha[1]       0.0000  7.091e-02      0.000      1.000   [ -0.139,  0.139]
gamma[1]       0.2094  7.451e-02      2.810  4.957e-03 [6.332e-02,  0.355]
beta[1]        0.8520      0.115      7.439  1.011e-13   [  0.628,  1.076]
===========================================================================

Covariance estimator: robust
ARCHModelResult, id: 0x26ba50f9888
```

## 10 EGARCH

[14]:

```python
from arch.univariate import EGARCH
model = arch_model(dta, mean ='constant',vol='EGARCH', p=1,o=1, q=1)
res =model.fit(update_freq=5)
res
```

```
Iteration:      5,   Func. Count:     50,   Neg. LLF: 1447.6823394342935
Iteration:     10,   Func. Count:     87,   Neg. LLF: 1444.6700658480654
Optimization terminated successfully.    (Exit mode 0)
            Current function value: 1444.6671831233068
            Iterations: 14
            Function evaluations: 115
            Gradient evaluations: 14
```

[14]:

```
                     Constant Mean - EGARCH Model Results
==============================================================================
Dep. Variable:                      R   R-squared:                      -0.001
```

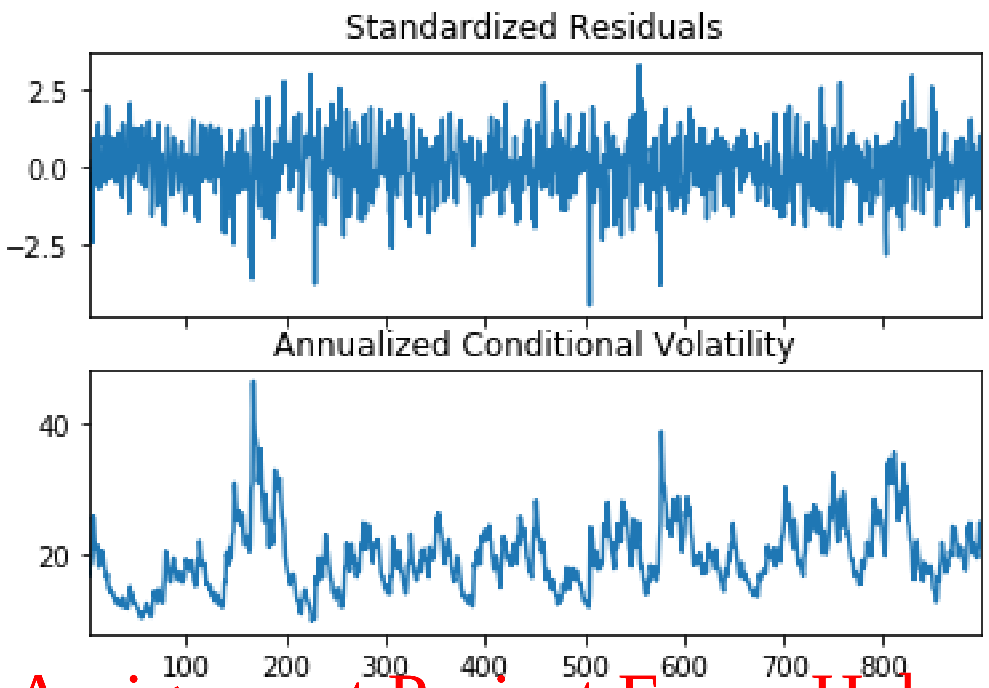
```
Mean Model:             Constant Mean   Adj. R-squared:                 -0.001
Vol Model:                     EGARCH   Log-Likelihood:               -1444.67
Distribution:                  Normal   AIC:                           2899.33
Method:            Maximum Likelihood   BIC:                           2923.34
                                        No. Observations:                  899
Date:                Thu, Aug 06 2020   Df Residuals:                      894
Time:                        20:19:12   Df Model:                            5
                                  Mean Model
==============================================================================
                 coef    std err          t      P>|t|        95.0% Conf. Int.
------------------------------------------------------------------------------
mu        -6.9738e-03  3.763e-02     -0.185      0.853 [-8.073e-02,6.678e-02]
                               Volatility Model
=============================================================================
                coef    std err          t      P>|t|        95.0% Conf. Int.
-----------------------------------------------------------------------------
omega         0.0243  1.557e-02      1.558      0.119 [-6.265e-03,5.477e-02]
alpha[1]      0.0862  3.307e-02      2.608  9.114e-03     [2.142e-02,  0.151]
gamma[1]     -0.1707  4.196e-02     -4.068  4.738e-05    [ -0.253,-8.846e-02]
beta[1]       0.9418  2.856e-02     33.082 6.375e-240      [  0.889,  1.001]
=============================================================================

Covariance estimator: robust
ARCHModelResult, id: 0x26ba50b94c8
```

[ ]:

## 11 4d Plotting residuals and conditional volatility

[15]:

```
#Standardised residual plots
fig =res.plot(annualize='D')
```



## 12 ACF and PACF of Standardised Residuals (dt) and Standardised Residuals Squared (dts)

[16]:

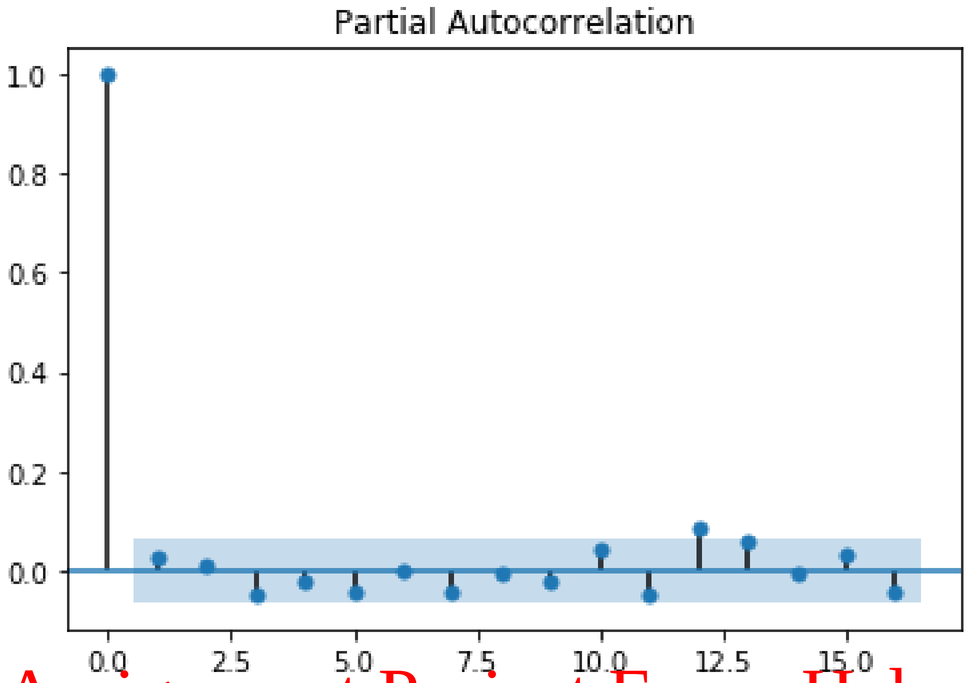
```
dt=res.resid/res.conditional_volatility
dts=dt**2
```

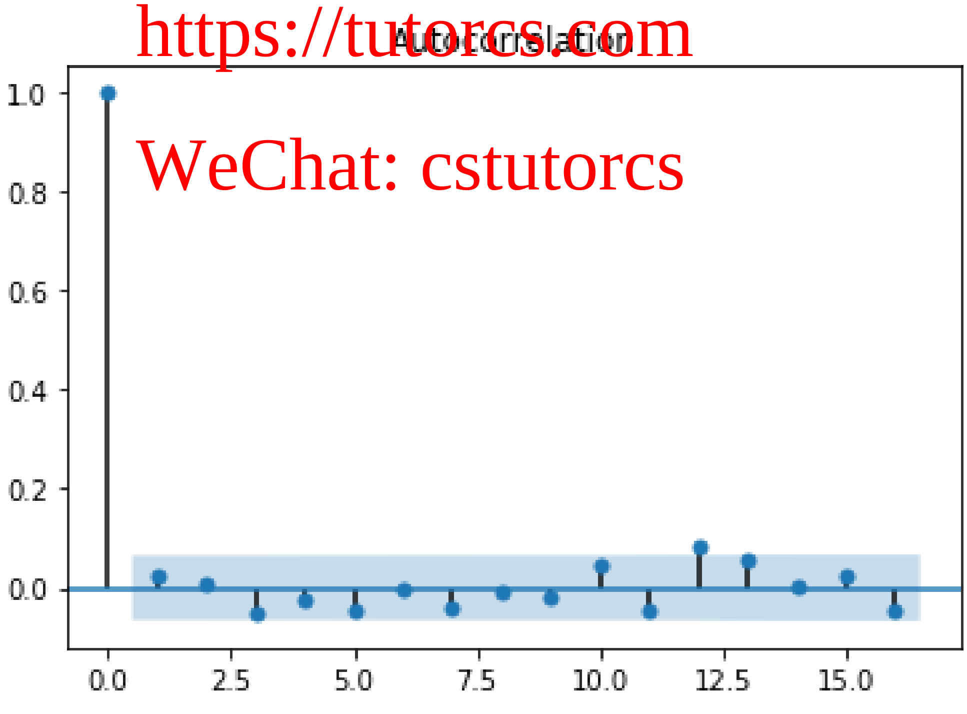
## 13 Standardised Residuals

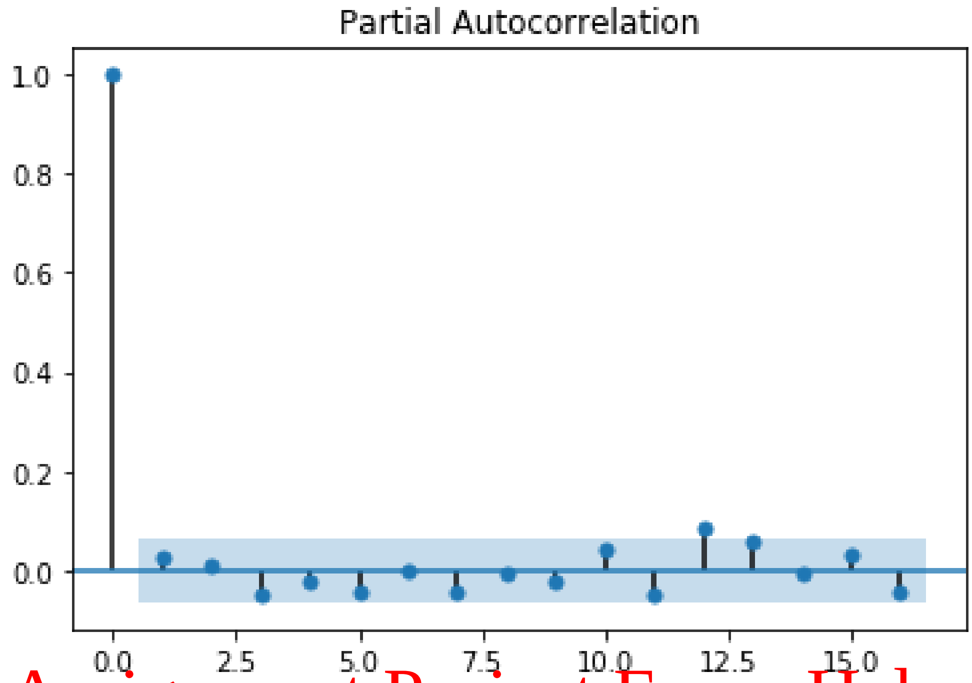
[17]:

```
sm.graphics.tsa.plot_acf(dt.values.squeeze(),lags=16)
sm.graphics.tsa.plot_pacf(dt.values.squeeze(),lags=16)
```

[17]:


Partial Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0


Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0



```
[18]: r,q,p=sm.tsa.acf(dt.values.squeeze(),qstat=True)
      data = np.c_[range(1,41),r[1:],q,p]
      table =pd.DataFrame(data,columns =['lag',"AC","Q","Prob(>Q)"])
      print(table.set_index('lag'))
```

```
            AC          Q  Prob(>Q)
lag
1.0   0.023511   0.498594  0.480118
2.0   0.009502   0.580118  0.748219
3.0  -0.048597   2.715134  0.437661
4.0  -0.023518   3.215702  0.522399
5.0  -0.045766   5.113457  0.402191
6.0  -0.000914   5.114214  0.529250
7.0  -0.039940   6.562797  0.475773
8.0  -0.005681   6.592140  0.581207
9.0  -0.018728   6.911354  0.646348
10.0  0.045250   8.776947  0.553395
11.0 -0.046517  10.750744  0.464375
12.0  0.086353  17.560259  0.129709
13.0  0.055651  20.391598  0.085867
14.0  0.003819  20.404947  0.117883
15.0  0.023698  20.919521  0.139420
16.0 -0.046797  22.928449  0.115661
17.0  0.027888  23.642689  0.129473
18.0 -0.049931  25.934858  0.101257
```

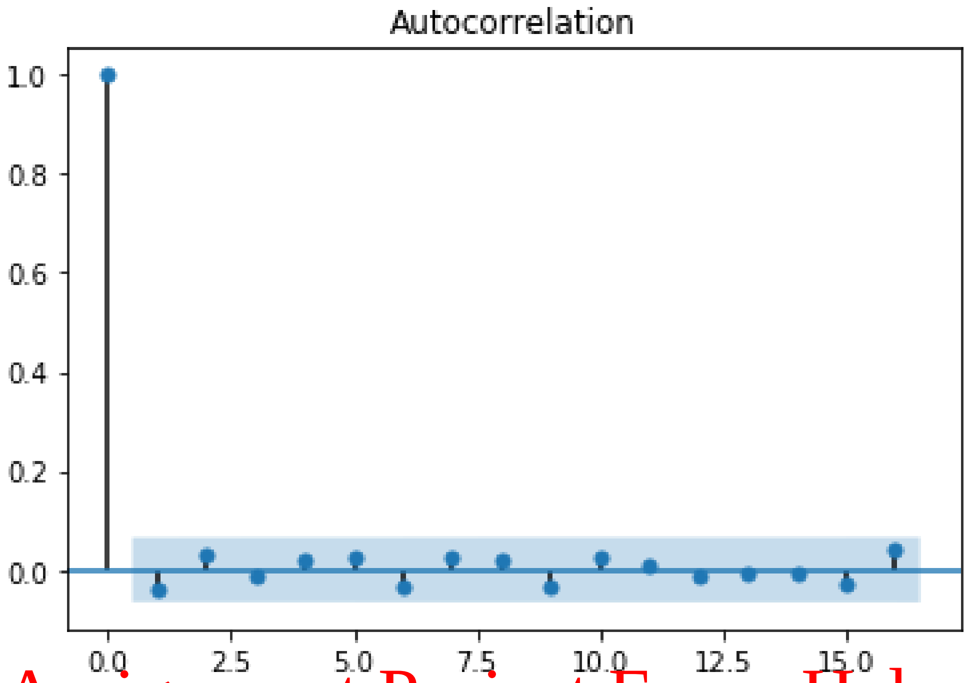
```
19.0  0.036480  27.159779  0.100986
20.0 -0.026914  27.827276  0.113552
21.0 -0.059934  31.141168  0.071340
22.0 -0.034555  32.244006  0.073330
23.0  0.011467  32.365596  0.092776
24.0  0.034136  33.444312  0.095040
25.0  0.013432  33.611528  0.116476
26.0  0.018026  33.913014  0.137228
27.0  0.067144  38.100821  0.076282
28.0 -0.017172  38.375053  0.091511
29.0  0.056514  41.348610  0.064183
30.0 -0.005811  41.380084  0.080782
31.0 -0.013936  41.561314  0.097461
32.0 -0.050735  43.966145  0.077387
33.0 -0.016447  44.219145  0.091800
34.0 -0.091372  52.037197  0.024621
35.0 -0.028167  52.781009  0.027331
36.0  0.010745  52.889367  0.034398
37.0 -0.001613  52.891812  0.043694
38.0 -0.011688  53.020320  0.053535
39.0  0.069750  57.602506  0.027760
40.0 -0.088893  65.053707  0.007387

C:\Users\rluck\anaconda3\lib\site-packages\statsmodels\tsa\stattools.py:572:
FutureWarning: fft=True will become the default in a future version of
statsmodels. To suppress this warning, explicitly set fft=False.
  FutureWarning
```

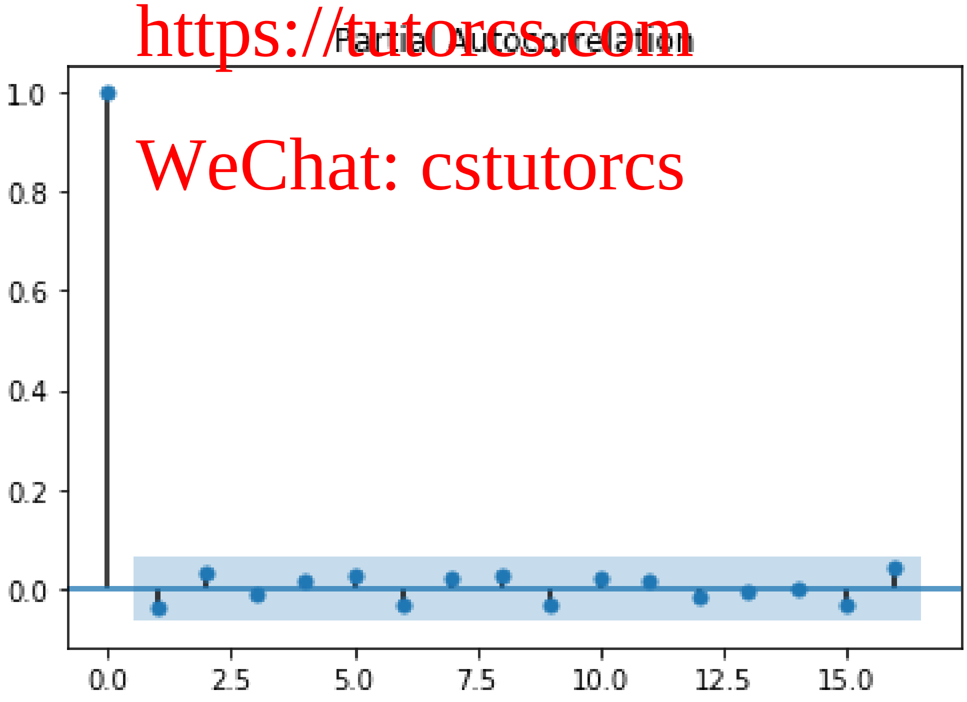
## 14 Standardised Residuals Squared

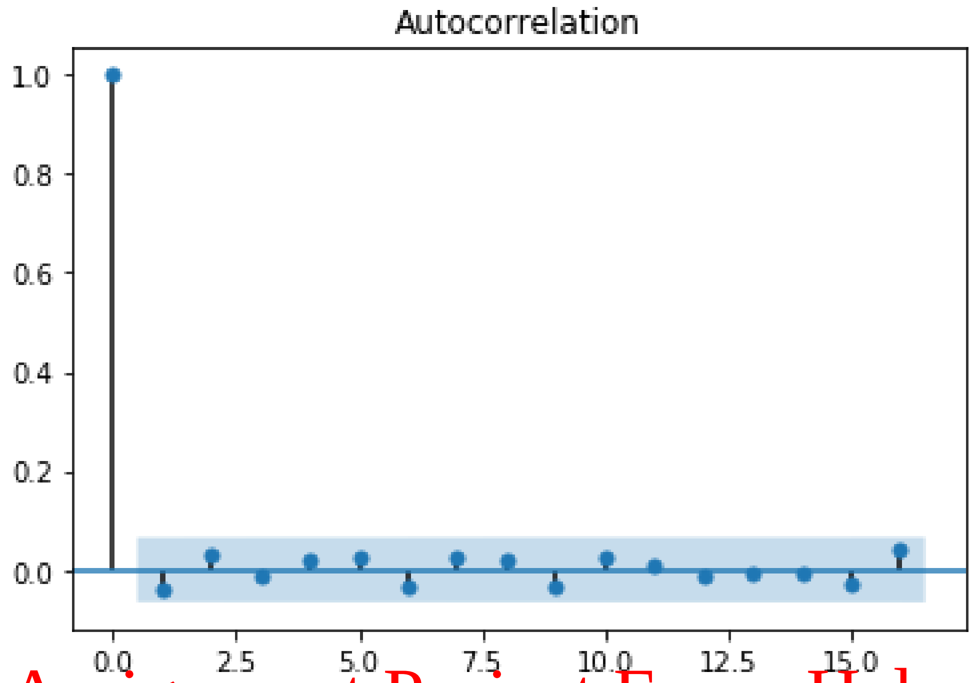
[19]:

```
sm.graphics.tsa.plot_pacf(dts.values.squeeze(),lags=16)
sm.graphics.tsa.plot_acf(dts.values.squeeze(),lags=16)
```

[19]:


Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0


Partial Autocorrelation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0



[20]:

```
r,q,p=sm.tsa.acf(dta.values.squeeze(),qstat=True)
data = np.c_[range(1,41),r[1:],q,p]
table =pd.DataFrame(data,columns =['lag',"AC","Q","Prob(>Q)"])
print(table.set_index('lag'))
```

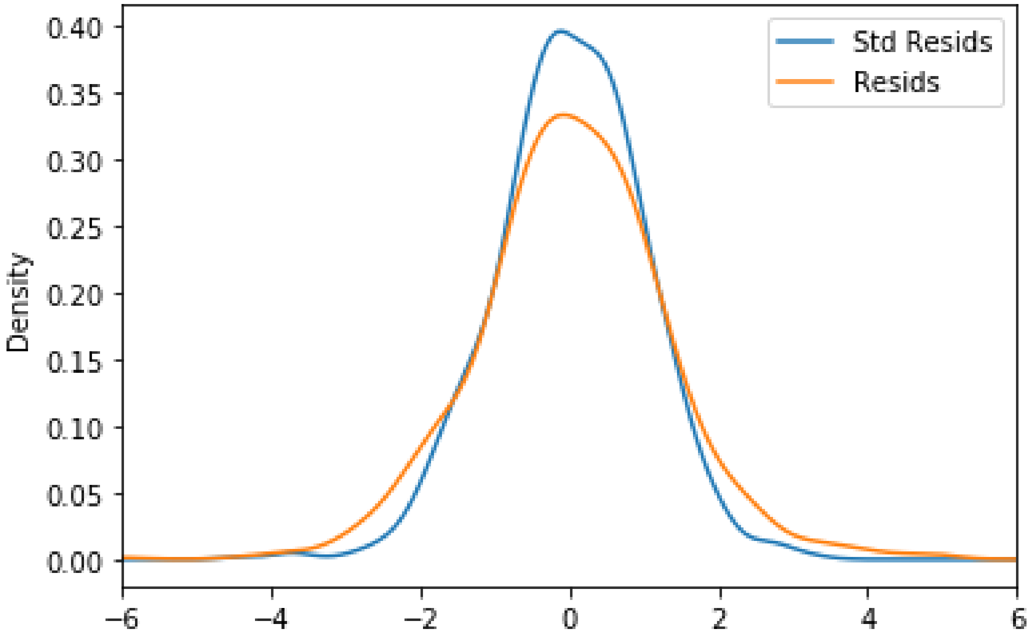
```
             AC          Q  Prob(>Q)
lag
1.0   -0.038583   1.342757  0.246548
2.0    0.031452   2.236045  0.326926
3.0   -0.012011   2.366469  0.499907
4.0    0.017259   2.636040  0.620452
5.0    0.022684   3.102243  0.684226
6.0   -0.035242   4.228776  0.645748
7.0    0.024418   4.770187  0.687985
8.0    0.020524   5.153137  0.741090
9.0   -0.034055   6.208615  0.718871
10.0   0.023009   6.690968  0.754262
11.0   0.008363   6.754767  0.818581
12.0  -0.010135   6.848560  0.867457
13.0  -0.005896   6.880340  0.908212
14.0  -0.004423   6.898247  0.938563
15.0  -0.026918   7.562152  0.940136
16.0   0.039820   9.016709  0.912724
17.0   0.034510  10.110429  0.898916
18.0   0.026269  10.744899  0.904867
```

```
19.0  0.038315  12.096183  0.881473
20.0  0.002313  12.101114  0.912552
21.0  0.107708  22.803556  0.354541
22.0 -0.015359  23.021423  0.400521
23.0 -0.043127  24.741213  0.363765
24.0  0.018051  25.042830  0.403436
25.0  0.063808  28.816092  0.271677
26.0  0.011376  28.936166  0.313993
27.0  0.083034  35.340649  0.130445
28.0 -0.003951  35.355164  0.159763
29.0  0.032513  36.339372  0.163886
30.0 -0.045949  38.307351  0.142017
31.0  0.031797  39.250832  0.146821
32.0  0.074017  44.369182  0.071658
33.0  0.034348  45.472696  0.072746
34.0 -0.029492  46.287188  0.077851
35.0  0.001479  46.289238  0.096038
36.0 -0.031647  47.229249  0.099705
37.0 -0.006140  47.264678  0.120226
38.0 -0.020120  47.645511  0.135735
39.0 -0.009904  47.737896  0.159112
40.0 -0.024120  48.286462  0.172926
```

```
C:\Users\rluck\anaconda3\lib\site-packages\statsmodels\tsa\stattools.py:572:
FutureWarning: fft=True will become the default in a future version of
statsmodels. To suppress this warning, explicitly set fft=False.
  FutureWarning
```

## 15 Standardised Residuals Statistics

[21]:
```
std_resid = res.resid / res.conditional_volatility
resid = res.resid
df = pd.concat([std_resid, resid], 1)
df.columns = ['Std Resids', 'Resids']
subplot = df.plot(kind='kde', xlim=(-6, 6))
```



## 16 Standardised Residuals Statistics

[22]:
```
stats.describe(dt)
```

[22]:
```
DescribeResult(nobs=899, minmax=(-4.4341245633175355, 3.298921120454863),
mean=0.010834927485779064, variance=0.9995416472236635,
skewness=-0.23125990208915737, kurtosis=0.7657947338486237)
```

[23]:
```
skewness =-0.23125990208915737
kurtosis =0.7657947338486237
nobs =899
JB =(skewness**2+0.25*(kurtosis**2))*nobs/6
JB
```

[23]:
```
29.980381797460023
```

[24]:
```
dt.describe()
```

[24]:
```
count    899.000000
mean       0.010835
std        0.999771
min       -4.434125
25%       -0.611477
50%        0.024652
75%        0.675022
```

```
max        3.298921
dtype: float64
```

## 17 Residuals Statistics

[25]:
```
stats.describe(resid)
```

[25]:
```
DescribeResult(nobs=899, minmax=(-7.036783685554021, 4.971570575558916),
mean=0.03060410465469678, variance=1.693610695043089,
skewness=-0.12413029499760052, kurtosis=2.041118218908278)
```

[26]:
```
skewness =-0.12413029499760052
kurtosis =2.041118218908278
nobs =899
JB =(skewness**2+0.25*(kurtosis**2))*nobs/6
JB
```

[26]:
```
158.36622569956484
```

[27]:
```
resid.describe()
```

[27]:
```
count    899.000000
mean       0.030604
std        1.301388
min       -7.036784
25%       -0.711097
50%        0.034797
75%        0.815742
max        4.971571
Name: resid, dtype: float64
```

## 18 Forecasts

[28]:
```
forecasts =res.forecast()
s=forecasts.variance.tail(1)
s
```

[28]:
```
          h.1
899  1.632889
```

[29]:
```
sd= forecasts.residual_variance.iloc[-1:]
sd
```

[29]:
```
          h.1
899  1.632889
```

[30]:
```
sm =forecasts.mean.tail(1)
sm
```

[30]:
```
          h.1
899 -0.006974
```

## 19 Value-at-Risk (VaR)

[31]:
```
q= dt.quantile(0.01)
q
```

[31]:
```
-2.4238806396103247
```

[32]:
```
res = model.fit(last_obs=(2001,28,7), update_freq=5)
forecasts = res.forecast(horizon=1)
print(forecasts.variance.dropna().head())
```

```
Iteration:      5,   Func. Count:     50,   Neg. LLF: 1447.6823394342935
Iteration:     10,   Func. Count:     87,   Neg. LLF: 1444.6700658480654
Optimization terminated successfully.    (Exit mode 0)
            Current function value: 1444.6671831233068
            Iterations: 14
            Function evaluations: 116
            Gradient evaluations: 14
          h.1
899  1.632889
```

[33]:
```
cond_mean=forecasts.mean
cond_mean.tail(1)
```

[33]:
```
          h.1
899 -0.006974
```

[34]:
```
cond_var=forecasts.variance
cond_var.tail(1)
```

[34]:
```
          h.1
899  1.632889
```

[35]:
```
P= 10000000
VaR = (cond_mean - np.sqrt(cond_var)* q)*P/100
VaR.tail(1)
```

[35]:
```
               h.1
899  309037.145464
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```