StockReturnsAnalysis

September 29, 2021

1 Stock and Benchmark Returns Analysis

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import math
import scipy.stats as scs
from scipy import stats
import statsmodels.api as sm

import warnings
warnings.filterwarnings("ignore")

# fix_yahoo_finance is used to fetch data
import fix_yahoo_finance as yf
yf.pdr_override()
```

[2]:
```
# input
market = "SPY"
symbol = 'AAPL'
start = '2016-01-01'
end = '2019-01-01'

# Read data
data = yf.download(symbol,start,end)
stock_market = yf.download(market, start, end)
```

```
[*********************100%***********************]  1 of 1 downloaded
[*********************100%***********************]  1 of 1 downloaded
```

[3]:
```
from datetime import datetime

def days_between(start, end):
    start = datetime.strptime(start, "%Y-%m-%d")
    end = datetime.strptime(end, "%Y-%m-%d")
    n = abs((end - start).days)
```

```
    return n

days_between(start, end)
```

[3]: 1096

[4]:
```
start = datetime.strptime(start, "%Y-%m-%d")
end = datetime.strptime(end, "%Y-%m-%d")
n = abs((end - start).days)
```

[5]:
```
data.head()
```

[5]:
```
                  Open        High         Low       Close  Adj Close  \
Date
2016-01-04  102.610001  105.370003  102.000000  105.349998  98.446655
2016-01-05  105.750000  105.849998  102.410004  102.709999  95.979675
2016-01-06  100.559998  102.370003   99.870003  100.699997  94.101387
2016-01-07   98.680000  100.129997   96.430000   96.449997  90.129868
2016-01-08   98.550003   99.110001   96.760002   96.959999  90.606438

              Volume
Date
2016-01-04  67649400
2016-01-05  55791000
2016-01-06  68457400
2016-01-07  81094400
2016-01-08  70798000
```

[6]:
```
stock_market.head()
```

[6]:
```
                  Open        High         Low       Close   Adj Close  \
Date
2016-01-04  200.490005  201.029999  198.589996  201.020004  186.836166
2016-01-05  201.399994  201.899994  200.050003  201.360001  187.152161
2016-01-06  198.339996  200.059998  197.600006  198.820007  184.791367
2016-01-07  195.330002  197.440002  193.589996  194.050003  180.357925
2016-01-08  195.190002  195.850006  191.580002  191.919998  178.378235

               Volume
Date
2016-01-04  222353500
2016-01-05  110845800
2016-01-06  152112600
2016-01-07  213436100
2016-01-08  209817200
```

[7]:
```
rf = 0.01
```

[8]:
```
close_px = data[ 'Adj Close']
returns = close_px.pct_change().dropna()
```

[9]:
```
p = np.array(data['Adj Close'])
mp = np.array(stock_market['Adj Close'])

dollar_vol = np.array(data['Volume']*p)
market_dollar_vol = np.array(stock_market['Volume']*mp)
```

[10]:
```
benchmark = stock_market['Adj Close'].pct_change().dropna()
excess_returns = np.array(returns) - np.array(benchmark)
```

[11]:
```
data['returns'] = data[ 'Adj Close'].pct_change().dropna()
benchmark['returns'] = stock_market['Adj Close'].pct_change().dropna()
```

[12]:
```
data['rea_var'] = 252 * np.cumsum(data['returns']**2) / np.arange(len(data))
data['rea_vol'] = np.sqrt(data['rea_var'])
```

Alpha is measure of performance on a risk-adjusted basis Alpha also known as "Jensen index".

Beta is a measure of the volatility, or systematic risk or of a secruity or a portfolio. Beta is used in the capital asset pricing model (CAPM), a model that calculates the expected return of an asset based on its beta and expected market returns.

R-Squared is a statisitcal measure that represents the percentage of a fund or security's movements that can be explained by movements in a benchmark index (S&P 500).

[13]:
```
def adj_close_statistics(close_px):
        sta = scs.describe(close_px)
        print("%14s %15s" % ('statistic', 'value'))
        print(30 * "-")
        print("%14s %15.5f" % ('size', sta[0]))
        print("%14s %15.5f" % ('min', sta[1][0]))
        print("%14s %15.5f" % ('max', sta[1][1]))
        print("%14s %15.5f" % ('mean', sta[2]))
        print("%14s %15.5f" % ('std', np.sqrt(sta[3])))
        print("%14s %15.5f" % ('skew', sta[4]))
        print("%14s %15.5f" % ('kurtosis', sta[5]))

adj_close_statistics(close_px)
```

```
     statistic           value
------------------------------
          size       754.00000
           min        85.39510
           max       227.83980
          mean       142.97159
           std        38.35211
```

```
     skew         0.22706
 kurtosis        -0.96764
```

[14]:

```
def print_stock_statistics(data):
    print("RETURN SAMPLE STATISTICS")
    print("---------------------------------------------")
    print("Mean of Daily  Log Returns %9.6f" % np.mean(returns))
    print("Std  of Daily  Log Returns %9.6f" % np.std(returns))
    print("Mean of Annua. Log Returns %9.6f" % (np.mean(returns) * 252))
    print("Std  of Annua. Log Returns %9.6f" % (np.std(returns) * math.
↪sqrt(252)))
    print("---------------------------------------------")
    print("Skew of Sample Log Returns %9.6f" % scs.skew(returns))
    print("Skew Normal Test p-value   %9.6f" % scs.skewtest(returns)[1])
    print("---------------------------------------------")
    print("Kurt of Sample Log Returns %9.6f" % scs.kurtosis(returns))
    print("Kurt Normal Test p-value   %9.6f" % \
             scs.kurtosistest(returns)[1])
    print("---------------------------------------------")
    print("Normal Test p-value        %9.6f" % \
             scs.normaltest(returns)[1])
    print("---------------------------------------------")
    print("Realized Volatility        %9.6f" % data['rea_vol'].iloc[-1])
    print("Realized Variance          %9.6f" % data['rea_var'].iloc[-1])
    print("---------------------------------------------")
    print("Anderson Normality Test:                   ")
    print(stats.anderson(returns))
    print("---------------------------------------------")
    print("Shapiro_Wilk Test:                                              ")
    print(stats.shapiro(returns))
    print("Sharpe Ratio of Daily Returns:                          ")
    print("{0:.8f}".format(np.mean(returns) / np.std(returns)))
    print("Trading Sharpe for Daily:                               ")
    print("{0:.8f}".format((n*6.5) * (np.mean(returns)-rf // np.std(returns)*np.
↪sqrt(n*6.5))))
    print("Sharpe of Annua. Returns w/ days:              ")
    print("{0:.8f}".format((252) * (np.mean(returns)-rf // np.std(returns)*np.
↪sqrt(252))))
    print("Sharpe of Annua. Returns w/ days & hours:")
    print("{0:.8f}".format((252*6.5) * (np.mean(returns)-rf // np.
↪std(returns)*np.sqrt(252*6.5))))
    print("---------------------------------------------")
    print("Amihud Illiquidity                   %9.6g" % np.mean(np.
↪divide(abs(returns),dollar_vol[1:])))
    print("---------------------------------------------")
    print("Kelly Formula:                               ")
    print("{0:.8f}".format(np.mean(returns) - rf // (np.std(returns))**2))
```

```python
    print("Compounded Levered Return:             ")
    print("{0:.8f}".format(rf + (((252) * (np.mean(returns)-rf / np.
 ↪std(returns)*np.sqrt(252)))**2) // 2))
    print("Compounded Unlevered Return:    ")
    print("{0:.8f}".format(((np.mean(returns))*252)-(((np.std(returns))*np.
 ↪sqrt(252))**2) // 2))
    return

print_stock_statistics(data)
```

```
RETURN SAMPLE STATISTICS
---------------------------------------------
Mean of Daily  Log Returns  0.000718
Std  of Daily  Log Returns  0.014907
Mean of Annua. Log Returns  0.180839
Std  of Annua. Log Returns  0.236637
---------------------------------------------
Skew of Sample Log Returns -0.057076
Skew Normal Test p-value    0.518853
---------------------------------------------
Kurt of Sample Log Returns  3.603211
Kurt Normal Test p-value    0.000000
---------------------------------------------
Normal Test p-value         0.000000
---------------------------------------------
Realized Volatility         0.236911
Realized Variance           0.056127
---------------------------------------------
Anderson Normality Test:
AndersonResult(statistic=11.294812364065251, critical_values=array([0.573,
0.653, 0.783, 0.913, 1.086]), significance_level=array([15. , 10. ,  5. ,  2.5,
1. ]))
---------------------------------------------
Shapiro_Wilk Test:
(0.9423431754112244, 1.6331307413298203e-16)
Sharpe Ratio of Daily Returns:
0.04814037
Trading Sharpe for Daily:
5.11229805
Sharpe of Annua. Returns w/ days:
0.18083929
Sharpe of Annua. Returns w/ days & hours:
1.17545539
---------------------------------------------
Amihud Illiquidity                2.07985e-12
---------------------------------------------
Kelly Formula:
```

```
-44.99928238
Compounded Levered Return:
3600372.01000000
Compounded Unlevered Return:
0.18083929
```

[15]:
```python
def print_market_information(benchmark):
    print("RETURN BENCHMARK STATISTICS")
    print("---------------------------------------------")
    print("Mean of Daily  Log Returns %9.6f" % np.mean(benchmark['returns']))
    print("Std  of Daily  Log Returns %9.6f" % np.std(benchmark['returns']))
    print("Mean of Annua. Log Returns %9.6f" % (np.mean(benchmark['returns']) *␣
 ↪252))
    print("Std  of Annua. Log Returns %9.6f" % (np.std(benchmark['returns']) *␣
 ↪math.sqrt(252)))
    print("---------------------------------------------")
    print("Skew of Sample Log Returns %9.6f" % scs.skew(benchmark['returns']))
    print("Skew Normal Test p-value   %9.6f" % scs.
 ↪skewtest(benchmark['returns'])[1])
    print("---------------------------------------------")
    print("Kurt of Sample Log Returns %9.6f" % scs.
 ↪kurtosis(benchmark['returns']))
    print("Kurt Normal Test p-value   %9.6f" % scs.
 ↪kurtosistest(benchmark['returns'])[1])
    print("---------------------------------------------")
    print("Normal Test p-value        %9.6f" % scs.
 ↪normaltest(benchmark['returns'])[1])
    print("---------------------------------------------")
    print("Anderson Normality Test:                    ")
    print(stats.anderson(benchmark['returns']))
    return

print_market_information(benchmark)
```

```
RETURN BENCHMARK STATISTICS
---------------------------------------------
Mean of Daily  Log Returns  0.000401
Std  of Daily  Log Returns  0.008165
Mean of Annua. Log Returns  0.101160
Std  of Annua. Log Returns  0.129618
---------------------------------------------
Skew of Sample Log Returns -0.534591
Skew Normal Test p-value    0.000000
---------------------------------------------
Kurt of Sample Log Returns  5.163861
Kurt Normal Test p-value    0.000000
---------------------------------------------
```

```
Normal Test p-value         0.000000
---------------------------------------------
Anderson Normality Test:
AndersonResult(statistic=18.24201097391233, critical_values=array([0.573, 0.653,
0.783, 0.913, 1.086]), significance_level=array([15. , 10. ,  5. ,  2.5,  1. ]))
```

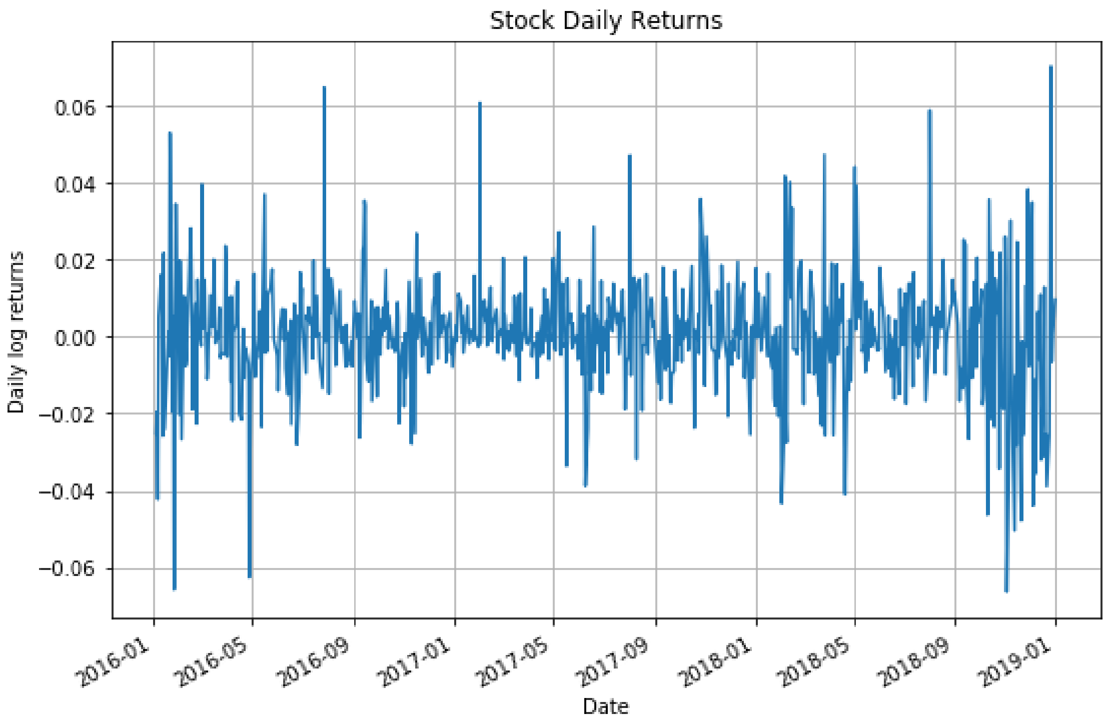
[16]:

```
def linreg(returns, benchmark):
    X = benchmark
    y = returns
    beta, intercept, r_squared, p_value, std_err = stats.linregress(X, y)
    alpha = np.mean(y) - beta * np.mean(X)
    print("alpha          = ", alpha)
    print("beta           = ", beta)
    print("r_squared         = ", r_squared)
    return beta, alpha, r_squared

def print_market_stock(returns, benchmark):
        y = returns
        x = benchmark
        x = sm.add_constant(x)
        model = sm.OLS(y,x)
        results = model.fit()
        print(results.summary())

# daily quotes and log returns
def quotes_returns(returns):
    ''' Plots quotes and returns. '''
    plt.figure(figsize=(9, 6))
    data['returns'].plot()
    plt.title('Stock Daily Returns')
    plt.ylabel('Daily log returns')
    plt.grid(True)
    plt.axis('tight')

# histogram of annualized daily log returns
def return_histogram(returns):
    ''' Plots a histogram of the returns. '''
    plt.figure(figsize=(9, 5))
    x = np.linspace(min(returns), max(returns), 100)
    plt.hist(np.array(returns), bins=50, normed=True)
    y = dN(x, np.mean(returns), np.std(returns))
    plt.plot(x, y, linewidth=2)
    plt.xlabel('Log Returns')
    plt.ylabel('Frequency/Probability')
    plt.grid(True)

# Q-Q plot of annualized daily log returns
```

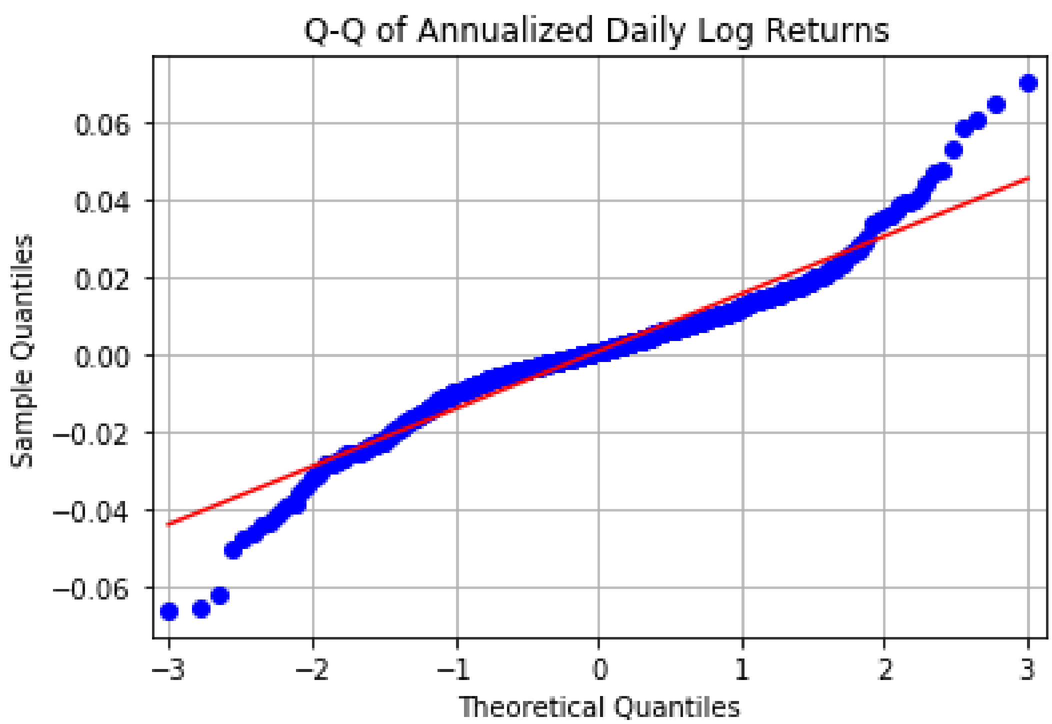
```
def return_qqplot(returns):
    ''' Generates a Q-Q plot of the returns.'''
    plt.figure(figsize=(9, 5))
    sm.qqplot(returns, line='s')
    plt.grid(True)
    plt.title('Q-Q of Annualized Daily Log Returns')
    plt.xlabel('Theoretical Quantiles')
    plt.ylabel('Sample Quantiles')


# realized volatility
def realized_volatility(returns):
    ''' Plots the realized volatility. '''
    plt.figure(figsize=(9, 5))
    data['rea_vol'].plot()
    plt.title('Stock Volatility')
    plt.ylabel('Realized Volatility')
    plt.grid(True)

# mean return, volatility and correlation (252 days moving = 1 year)
def rolling_statistics(returns):
    ''' Calculates and plots rolling statistics (mean, std, correlation). '''
    plt.figure(figsize=(11, 8))

    plt.subplot(311)
    mr = returns.rolling(252).mean() * 252
    mr.plot()
    plt.grid(True)
    plt.ylabel('returns (252d)')
    plt.axhline(mr.mean(), color='r', ls='dashed', lw=1.5)

    plt.subplot(312)
    vo = returns.rolling(252).std() * math.sqrt(252)
    vo.plot()
    plt.grid(True)
    plt.ylabel('volatility (252d)')
    plt.axhline(vo.mean(), color='r', ls='dashed', lw=1.5)
    vx = plt.axis()

    plt.subplot(313)
    co = mr.rolling(252).corr(vo, 252)
    co.plot()
    plt.grid(True)
    plt.ylabel('correlation (252d)')
    cx = plt.axis()
    plt.axis([vx[0], vx[1], cx[2], cx[3]])
    plt.axhline(co.mean(), color='r', ls='dashed', lw=1.5)
```

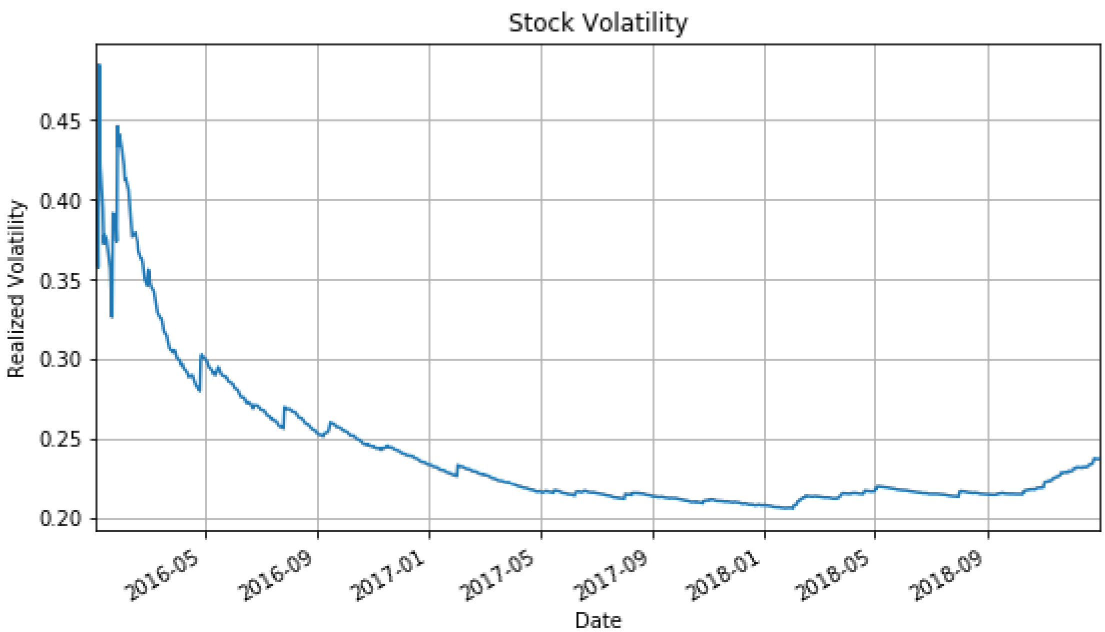
```python
if __name__ == '__main__':
    linreg(returns, benchmark['returns'])
    print_market_stock(returns, benchmark['returns'])
    quotes_returns(returns)
    return_qqplot(returns)
    realized_volatility(returns)
    rolling_statistics(returns)
```

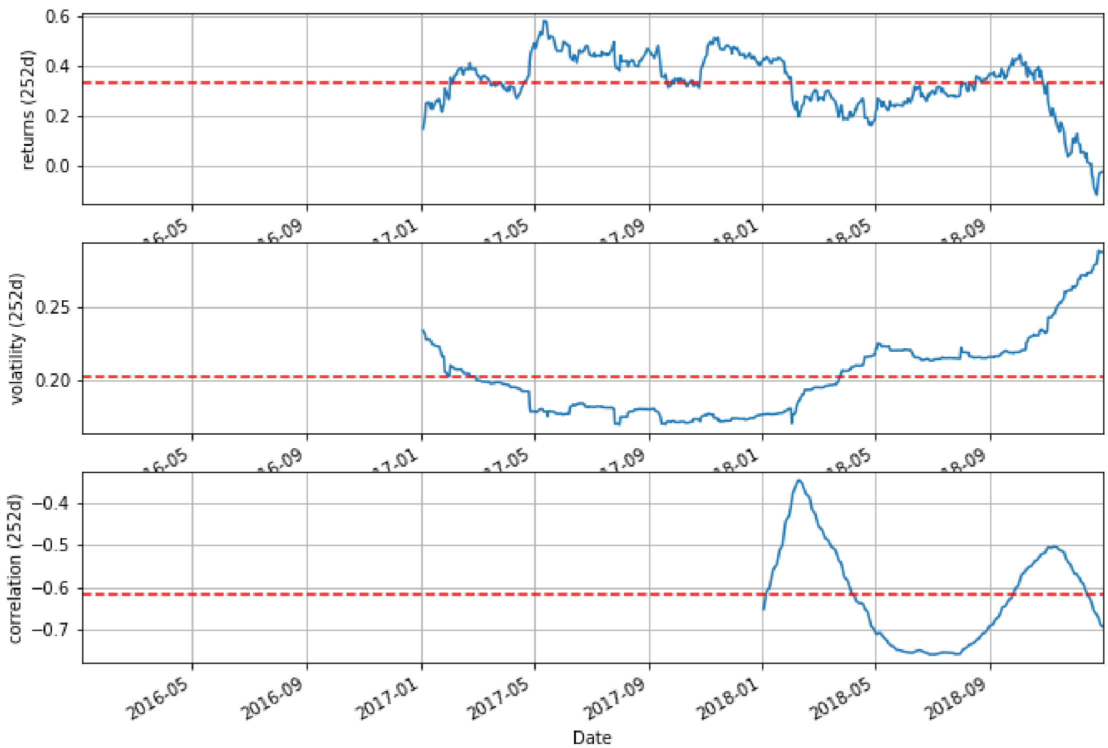
```
alpha   =  0.00024005764515231987
beta    = 1.1896463453886688
r_squared       =  0.6516289975275974
                            OLS Regression Results
==============================================================================
Dep. Variable:              Adj Close   R-squared:                       0.425
Model:                            OLS   Adj. R-squared:                  0.424
Method:                 Least Squares   F-statistic:                     554.2
Date:                Sun, 17 Nov 2019   Prob (F-statistic):           3.25e-92
Time:                        17:56:19   Log-Likelihood:                 2306.7
No. Observations:                 753   AIC:                            -4609.
Df Residuals:                     751   BIC:                            -4600.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0002      0.000      0.581      0.561      -0.001       0.001
Adj Close      1.1896      0.051     23.542      0.000       1.090       1.289
==============================================================================
Omnibus:                      129.078   Durbin-Watson:                   1.710
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1646.468
Skew:                           0.307   Prob(JB):                         0.00
Kurtosis:                      10.218   Cond. No.                         122.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```




```
<Figure size 648x360 with 0 Axes>
```




Stock Volatility
Realized Volatility
0.45
0.40
0.35
0.30
0.25
0.20
2016-05
2016-09
2017-01
2017-05
2017-09
2018-01
2018-05
2018-09
Date


returns (252d)
0.6
0.4
0.2
0.0
volatility (252d)
0.25
0.20
correlation (252d)
-0.4
-0.5
-0.6
-0.7
2016-05
2016-09
2017-01
2017-05
2017-09
2018-01
2018-05
2018-09
Date