Analysis of Key Statistics in Asset Returns

[27]:
```
import os
import time
import datetime
import numpy as np
import pandas as pd
import scipy.stats as scs
from pylab import plt, mpl

plt.style.use('seaborn-v0_8')
mpl.rcParams['savefig.dpi'] = 300
mpl.rcParams['font.family'] = 'serif'
pd.set_option('mode.chained_assignment', None)
pd.set_option('display.float_format', '{:.4f}'.format)
np.set_printoptions(suppress=True, precision=4)
os.environ['PYTHONHASHSEED'] = '0'
```

An In-depth Analysis

In the realm of financial theories, the normality assumption plays a pivotal role in dealing with uncertain returns of financial instruments. In this section, we analyze four historical financial time series using real-world data.

Data

We will analyze four historical financial time series, two for technology stocks and two for exchange traded funds (ETFs).

[28]:
```
dataFrame = pd.read_csv('data_.csv',
                        index_col=0, parse_dates=True).dropna()
columns =  ['SPY', 'GLD', 'AAPL.O', 'MSFT.O']
dFrame = dataFrame[columns].dropna()
dFrame.info() # display dataset information
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2516 entries, 2010-01-04 to 2019-12-31
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   SPY     2516 non-null   float64
```

```
 1   GLD     2516 non-null   float64
 2   AAPL.O  2516 non-null   float64
 3   MSFT.O  2516 non-null   float64
dtypes: float64(4)
memory usage: 98.3 KB
```

[29]:
```
dFrame.head() # display the first five rows
```

[29]:

Date	SPY	GLD	AAPL.O	MSFT.O
2010-01-04	113.3300	109.8000	30.5728	30.9500
2010-01-05	113.6300	109.7000	30.6257	30.9600
2010-01-06	113.7100	111.5100	30.1385	30.7700
2010-01-07	114.1900	110.8200	30.0828	30.4520
2010-01-08	114.5700	111.3700	30.2828	30.6600

Normalize the financial instruments' prices

Figure 1-1. shows the normalized prices of the financial assets

[30]:
```
(dFrame / dFrame.iloc[0] * 100).plot(figsize=(10, 6))
plt.figtext(0.5, 0.0001, 'Fig 1-1. Normalized prices of the financial assets',
 ↪style='italic', ha='center')
plt.show()
```

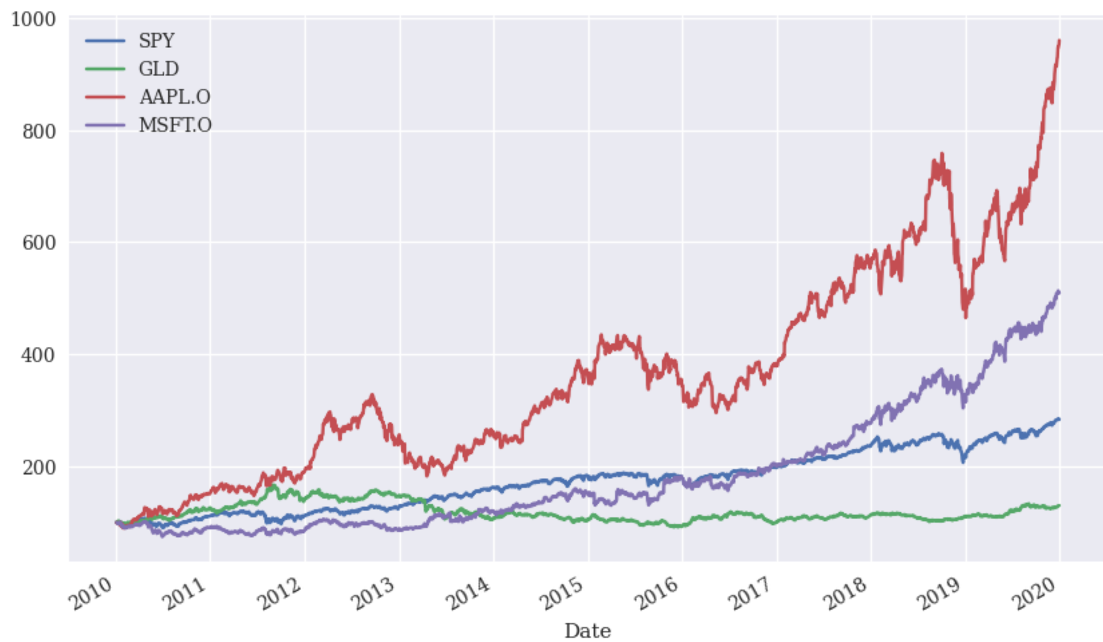


Fig 1-1. Normalized prices of the financial assets

Log returns of the financial instruments as histograms

[31]:
```
log_returns = np.log(dFrame / dFrame.shift(1))
log_returns.dropna(inplace=True)
log_returns.head()
```

[31]:

Date	SPY	GLD	AAPL.O	MSFT.O
2010-01-05	0.0026	-0.0009	0.0017	0.0003
2010-01-06	0.0007	0.0164	-0.0160	-0.0062
2010-01-07	0.0042	-0.0062	-0.0019	-0.0104
2010-01-08	0.0033	0.0050	0.0066	0.0068
2010-01-11	0.0014	0.0132	-0.0089	-0.0128

Figure 1-2. shows the log returns of the financial instruments as histograms

[32]:
```
log_returns.hist(bins=50, figsize=(10, 8));
plt.figtext(0.5, 0.0001, 'Fig 1-2. Histograms of log returns for financial␣
↪instruments', style='italic', ha='center')
plt.show()
```

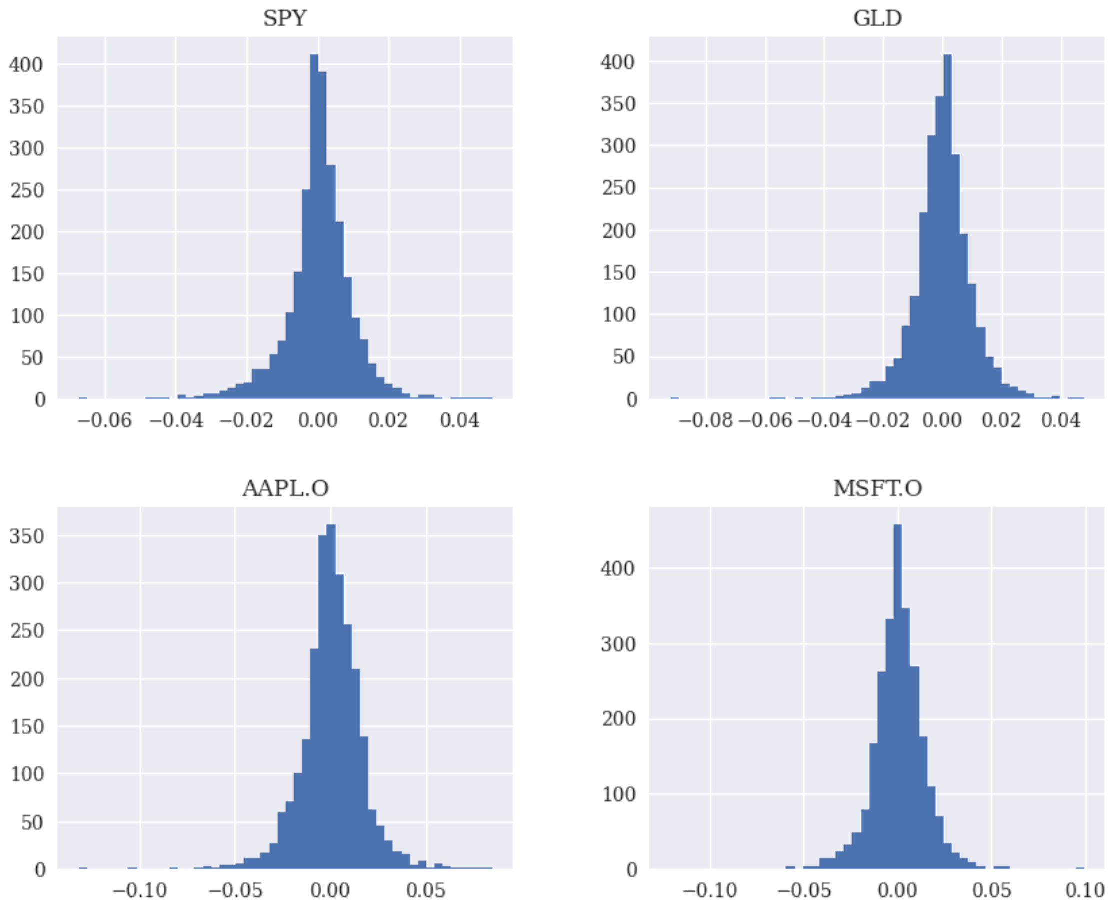


Fig 1-2. Histograms of log returns for financial instruments

Examining key statistics

We will use the function print_statistics(), which generates a better (human) readable key statistics output.

[33]:
```
def print_statistics(array):
    '''Prints selected statistics.
    Parameters
    ==========
    array: ndarray
        object to generate statistics on
    '''
    sta = scs.describe(array)
    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]))
```

[34]:
```
for col in columns:
    print('\nResults for column {}'.format(col))
    print(30 * '-')
    log_data = np.array(log_returns[col].dropna())
    print_statistics(log_data)
```

```

Results for column SPY
------------------------------
     statistic           value
------------------------------
          size      2515.00000
           min        -0.06734
           max         0.04929
          mean         0.00042
           std         0.00930
          skew        -0.50079
      kurtosis         4.45059

Results for column GLD
------------------------------
     statistic           value
------------------------------
          size      2515.00000
           min        -0.09191
           max         0.04795
          mean         0.00010
           std         0.00978
          skew        -0.58102
      kurtosis         5.89970

Results for column AAPL.O
------------------------------
     statistic           value
------------------------------
          size      2515.00000
           min        -0.13187
           max         0.08502
          mean         0.00090
           std         0.01625
```

```
     skew       -0.33706
 kurtosis        4.80934

Results for column MSFT.O
------------------------------
    statistic          value
------------------------------
         size     2515.00000
          min       -0.12103
          max        0.09941
         mean        0.00065
          std        0.01433
         skew       -0.10594
     kurtosis        6.41239
```

Analysis

Kurtosis, the fourth moment, characterizes the shape of the distribution's tails. It measures the peakedness or flatness of the distribution. In all four of our data sets, the high positive kurtosis results in a fatter tail on the right side, indicating a more pronounced peak compared to a normal distribution.

Normality Test

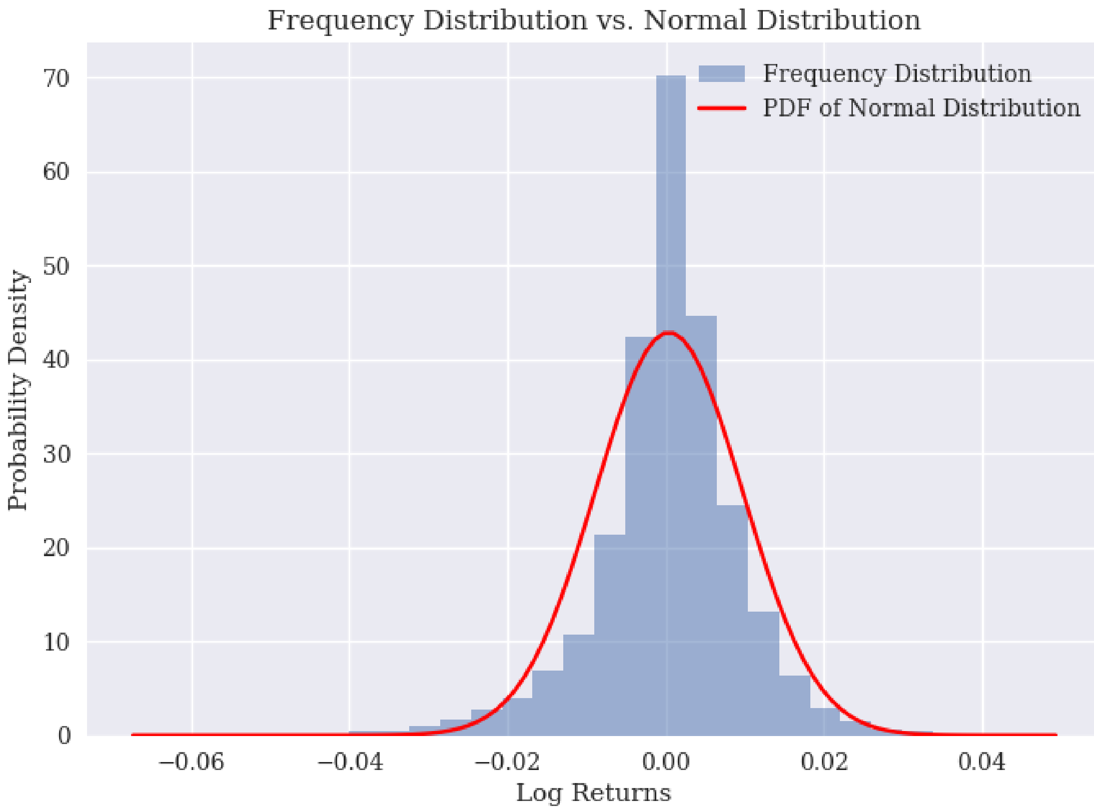
a. Graphical Normality Test

1. Compare the histograms and the PDF for normal distribution

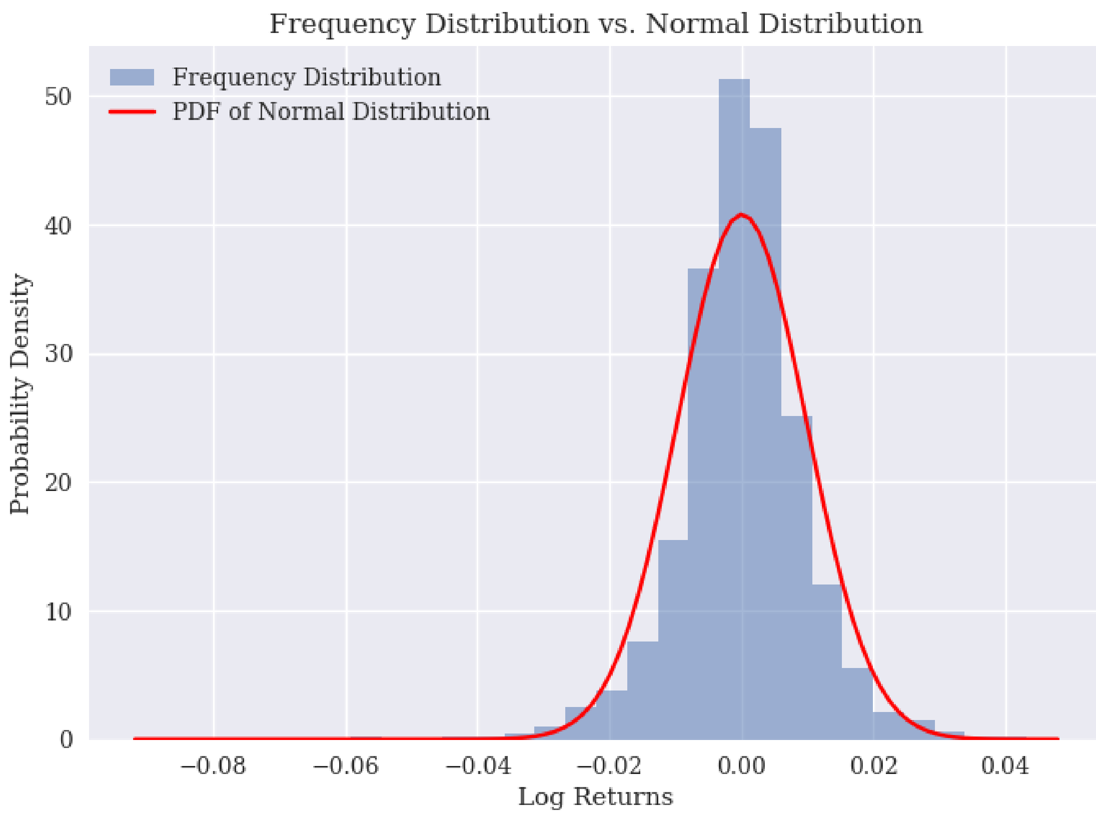
[35]:
```
from scipy.stats import norm
for col in columns:
    print('\nResults for column {}'.format(col))
    print(30 * '-')
    log_data = np.array(log_returns[col].dropna())

    # Plotting histogram and PDF for for the assets
    plt.hist(log_data, bins=30, density=True, alpha=0.5, label='Frequency␣
 ↪Distribution')
    mu, sigma = log_data.mean(), log_data.std()
    x = np.linspace(log_data.min(), log_data.max(), 100)
    plt.plot(x, norm.pdf(x, mu, sigma), 'r-', label='PDF of Normal␣
 ↪Distribution')
    plt.xlabel('Log Returns')
    plt.ylabel('Probability Density')
    plt.title('Frequency Distribution vs. Normal Distribution')
    plt.legend()
    plt.show()
```

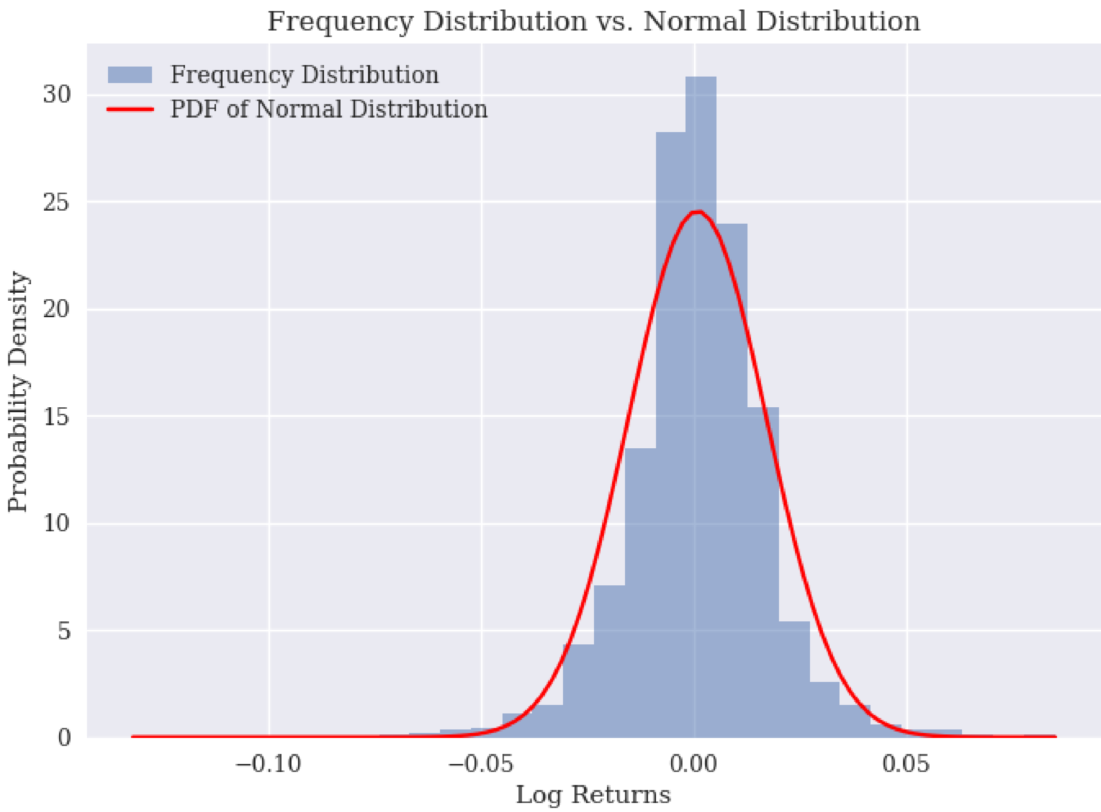
```
Results for column SPY
```



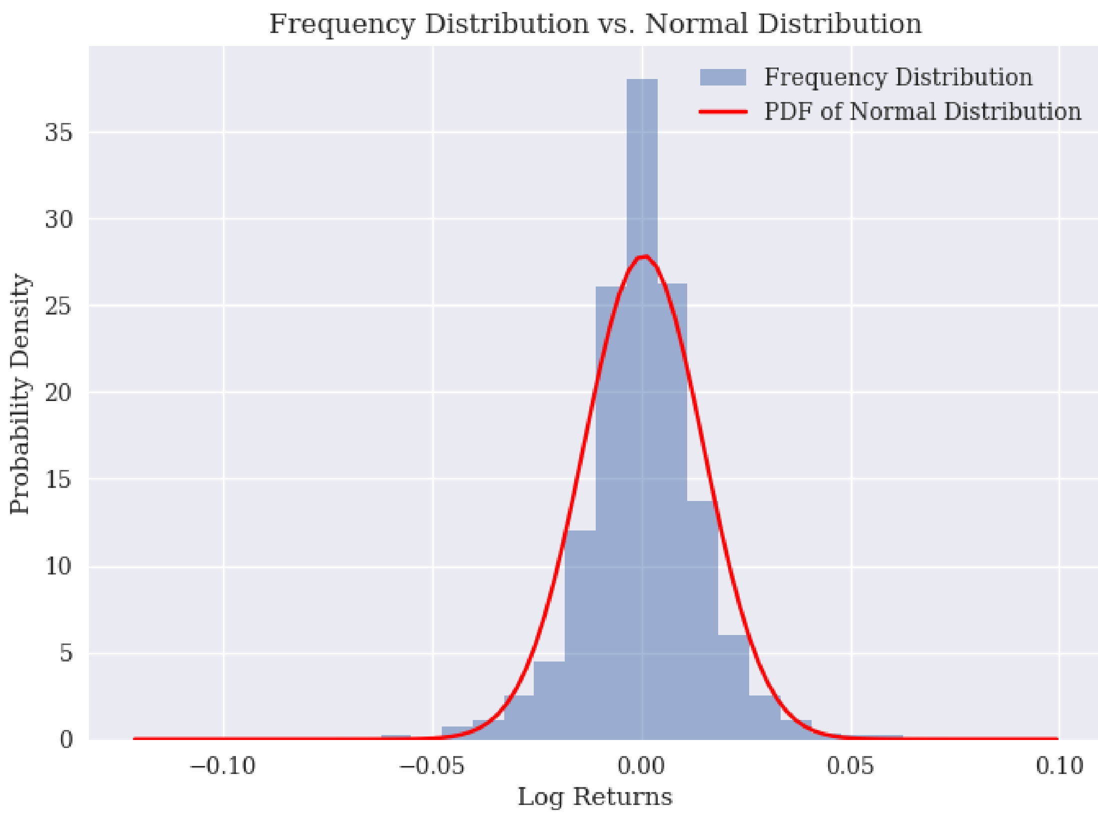
Results for column GLD




```
Results for column AAPL.O
------------------------------
```




```
Results for column MSFT.O
------------------------------
```




```
[ ]:
```

Analysis

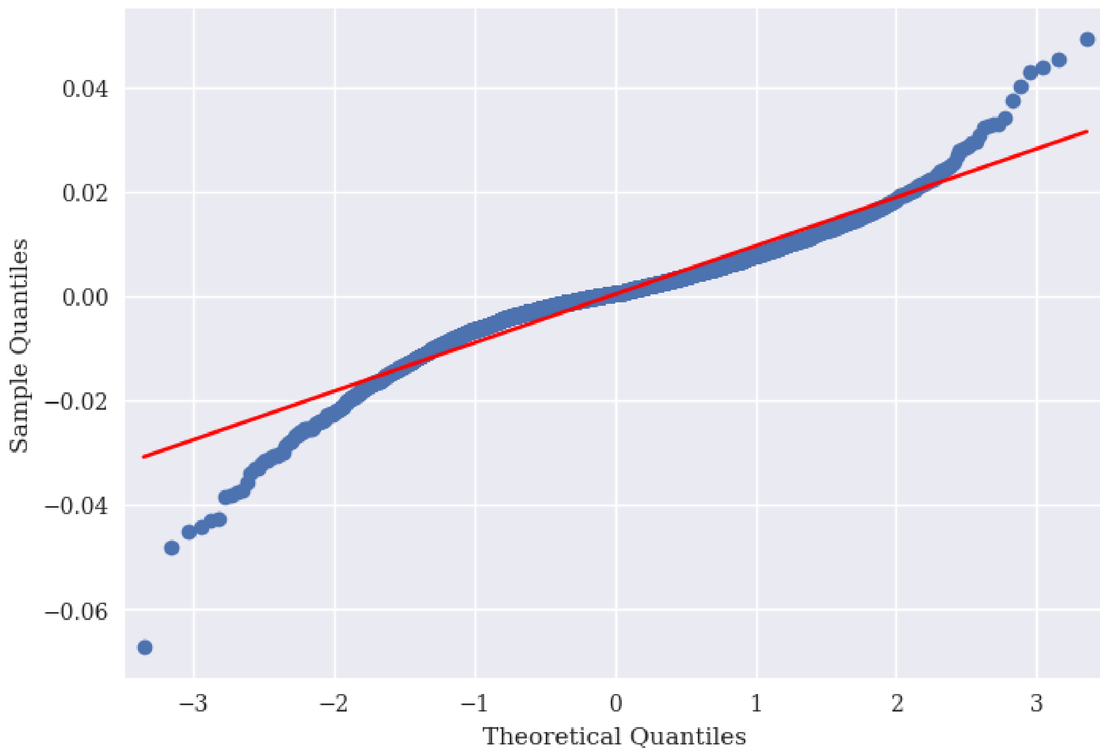
The histograms above represent the frequency distribution of the log returns, while the PDF represents the expected distribution if the log returns followed a normal distribution. By visually comparing the histograms and the PDF, we can assess if there is a good fit between the log returns and the normal distribution. In our case, we observe a slightly left-skewed distribution, indicating a longer left tail compared to a normal distribution.

2. Quantile-quantile plots

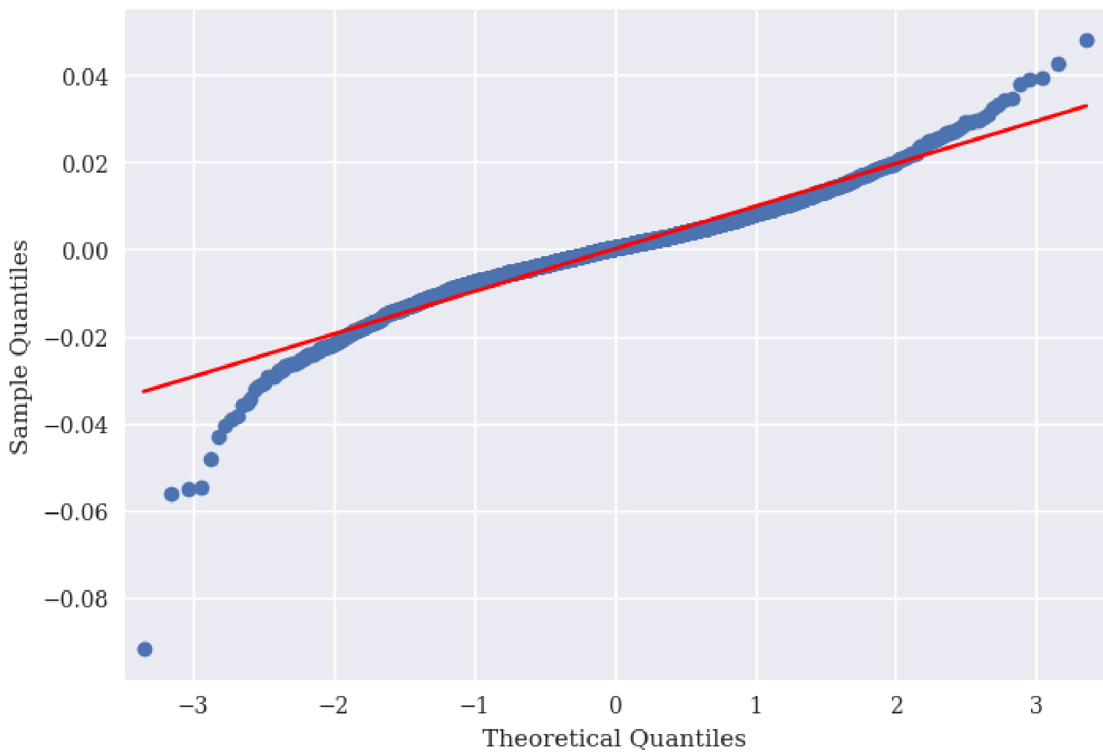
```
[36]: import statsmodels.api as sm
for col in columns:
    print('\nResults for column {}'.format(col))
    print(30 * '-')
    log_data = np.array(log_returns[col].dropna())

    #Generate the Q-Q plot
    sm.qqplot(log_data, line='s')
    plt.show()
```

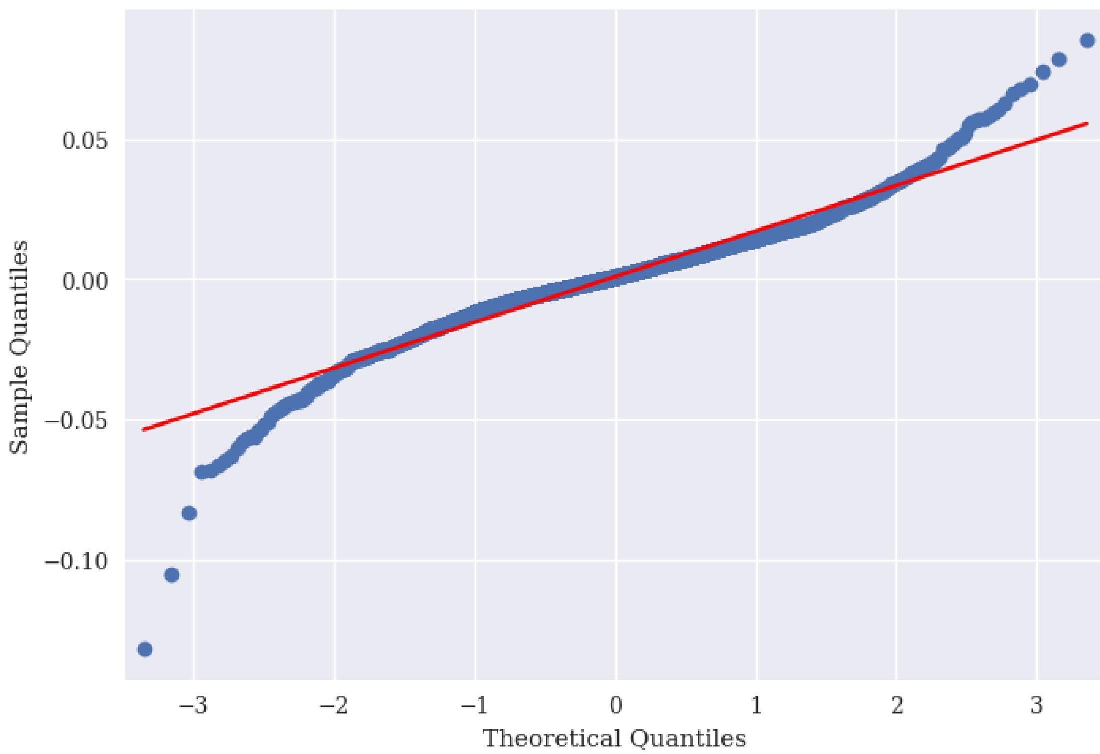
```
Results for column SPY
------------------------------
```



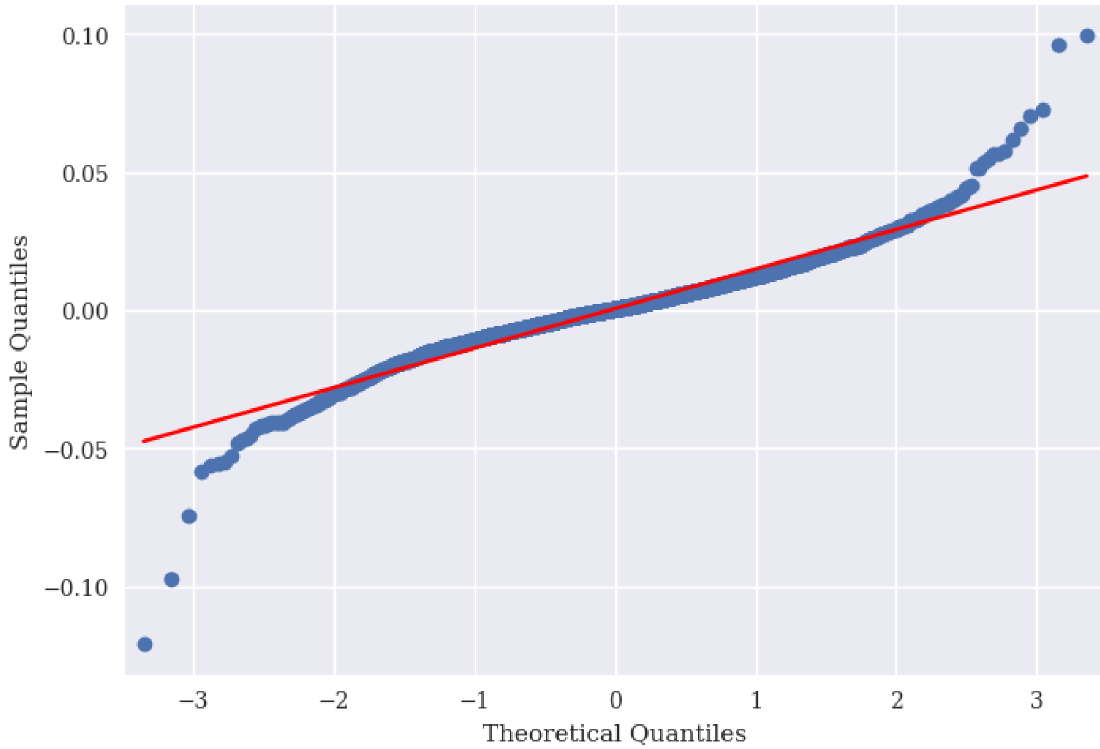

```
Results for column GLD
------------------------------
```




```
Results for column AAPL.O
------------------------------
```




```
Results for column MSFT.O
------------------------------
```



Analysis

The Q-Q plots displayed above depict the log returns, with the red line representing a perfect fit to a normal distribution. By visually examining the plot, we can assess the goodness of fit of the log returns to a normal distribution. It is evident that the points on the plot deviate from the reference line, indicating non-normality. On both the left and right sides, there are numerous values that lie significantly below and above the line, respectively. This observation provides evidence of a skewed or fat-tailed distribution, signifying a departure from normality.

b. Analytical Normality Test

However enticing the graphical approaches may appear, they are generally unable to substitute more rigorous testing procedures. In the following example, the function normality_tests() integrates three distinct statistical tests:

Skewness test (skewtest()): Tests whether the skew of the log returns is "normal" (i.e., has a value close enough to zero).

Kurtosis test (kurtosistest()): Similarly, this tests whether the kurtosis of the log returns is "normal" (again, close enough to zero).

Normality test (normaltest()): Combines the other two test approaches to test for normality.

[37]:
```
def normality_tests(arr):
    ''' Tests for normality distribution.
```

```python
    Parameters
    ==========
    array: ndarray
        object to generate statistics on
    '''
    print('Skew of data set %14.5f' % scs.skew(arr))
    print('Skew test p-value %14.5f' % scs.skewtest(arr)[1])
    print('Kurt of data set %14.5f' % scs.kurtosis(arr))
    print('Kurt test p-value %14.5f' % scs.kurtosistest(arr)[1])
    print('Norm test p-value %14.5f' % scs.normaltest(arr)[1])

for col in columns:
    print('\nResults for column {}'.format(col))
    print(30 * '-')
    log_data = np.array(log_returns[col].dropna())
    normality_tests(log_data)
```

```
Results for column SPY
------------------------------
Skew of data set       -0.50079
Skew test p-value       0.00000
Kurt of data set        4.45059
Kurt test p-value       0.00000
Norm test p-value       0.00000

Results for column GLD
------------------------------
Skew of data set       -0.58102
Skew test p-value       0.00000
Kurt of data set        5.89970
Kurt test p-value       0.00000
Norm test p-value       0.00000

Results for column AAPL.O
------------------------------
Skew of data set       -0.33706
Skew test p-value       0.00000
Kurt of data set        4.80934
Kurt test p-value       0.00000
Norm test p-value       0.00000

Results for column MSFT.O
------------------------------
Skew of data set       -0.10594
Skew test p-value       0.03005
Kurt of data set        6.41239
```

```
Kurt test p-value        0.00000
Norm test p-value        0.00000
```

Analysis

The log returns data with a skewness of -(0.) and a kurtosis of 3+ exhibits interesting characteristics.

The negative skewness indicates that the log returns' distribution is slightly left-skewed, meaning that the tail on the left side is more spread out than the tail on the right side, an indication there may be more extreme negative returns in the data compared to extreme positive returns.

The kurtosis value of 3+ indicates heavy tails and a higher peak compared to a normal distribution, suggesting a higher likelihood of observing extreme returns.

The p-values obtained from the different tests are all zero, indicating a complete rejection of the test hypothesis, which suggests that the log returns do not follow a normal distribution. This finding challenges the validity of the normal assumption for stock market returns and other asset classes, such as those embodied in the geometric Brownian motion model. Consequently, it becomes apparent that the utilization of richer models, capable of generating fat tails, such as jump diffusion models or models with stochastic volatility, may be necessary in order to accurately capture the underlying dynamics.