hw6

20989977 Zhang Mingtao

2023/11/10

1.

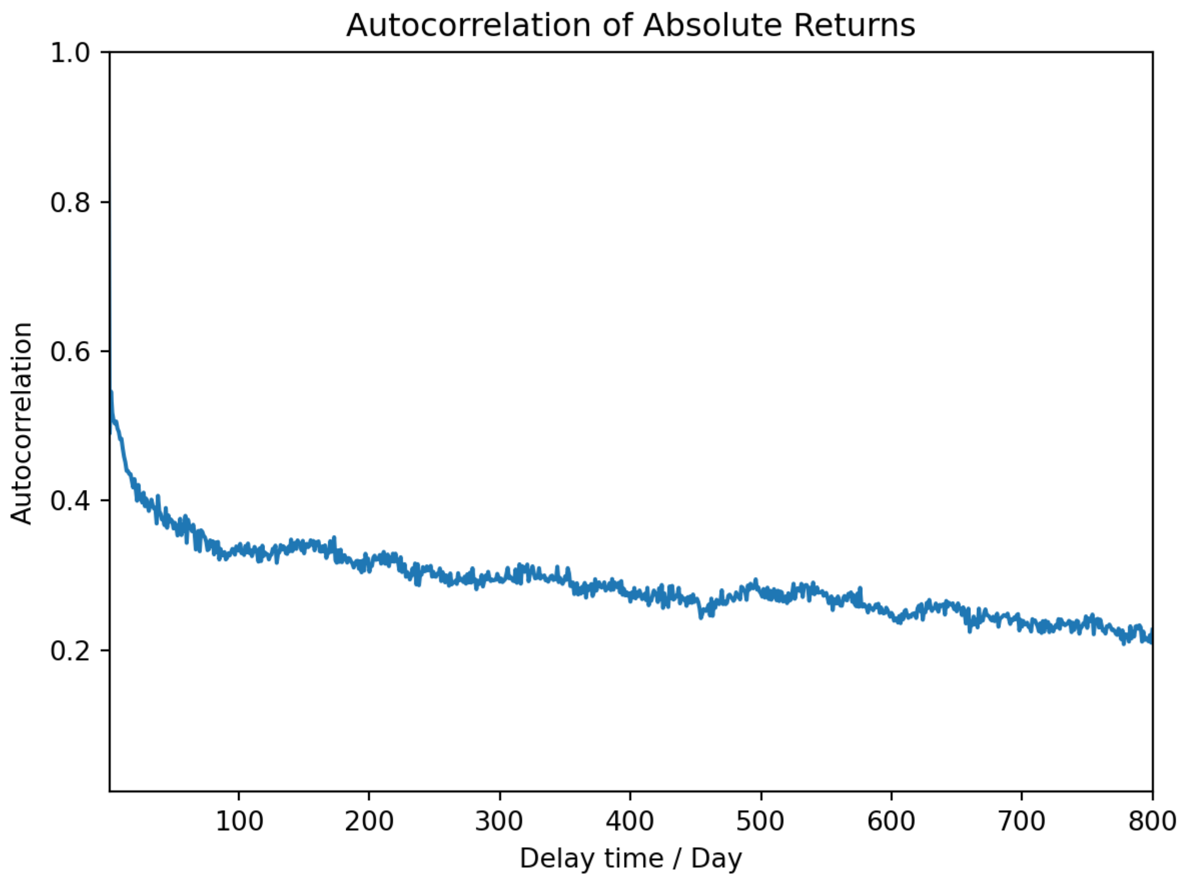
```
#1
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf

symbol = "^NYA"
data = yf.download(symbol, period="max")
```

```
##
[*********************100%%**********************]  1 of 1 completed
```

```
data.dropna(inplace=True)

absreturns = data['Adj Close'].diff().abs()[1:]

average = absreturns.mean()

ts = absreturns - average

lag = 800

fig = plt.figure()

res = plt.acorr(ts, maxlags=lag, usevlines = False, linestyle='solid', marker='')

plt.title('Autocorrelation of Absolute Returns')
plt.xlabel('Delay time / Day')
plt.ylabel('Autocorrelation')
plt.axis([1,lag,1.e-2,1.])
```

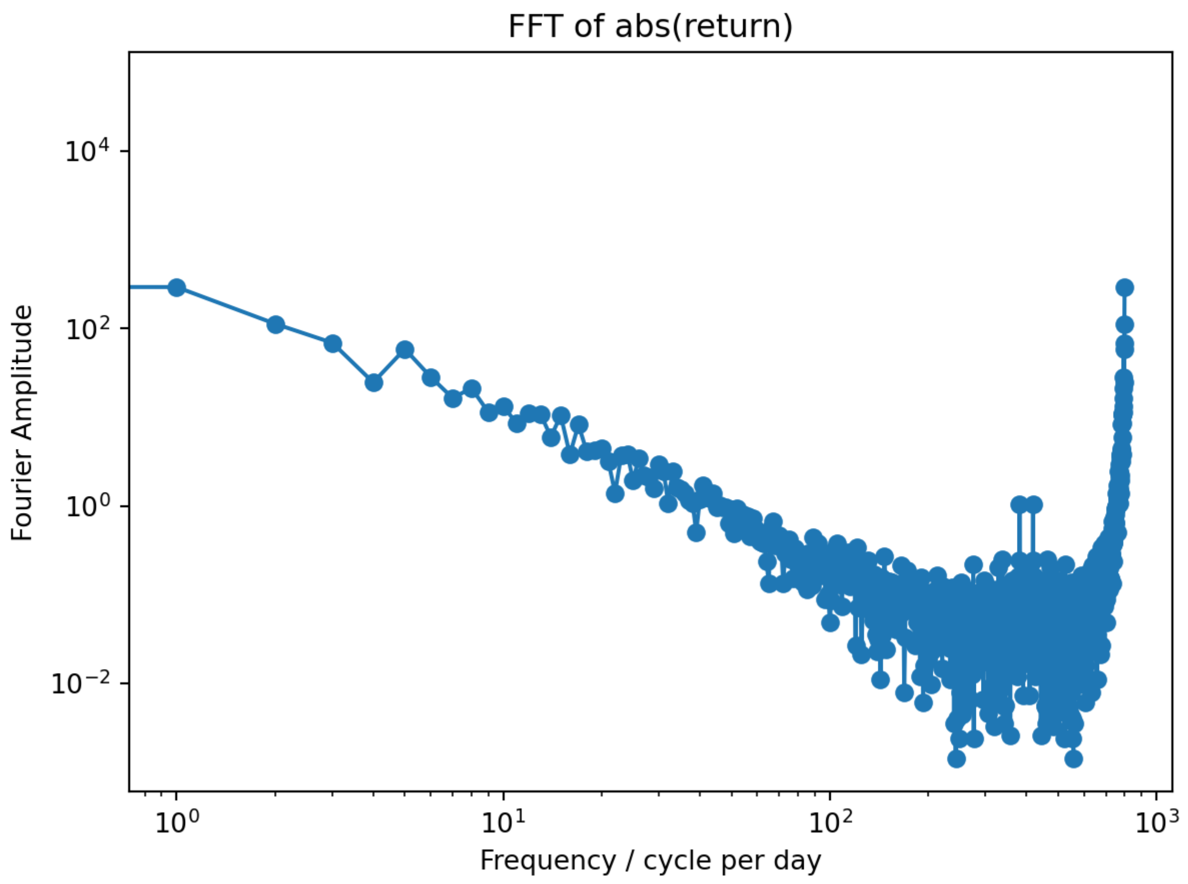
```
## (1.0, 800.0, 0.01, 1.0)
```

```
plt.show()

# for tau in range(lag, 2*lag + 1):
#     print(res[0][tau],res[1][tau])
```



2.

```
#2
from numpy.fft import fft
from sklearn.linear_model import LinearRegression

amp = np.abs(fft(res[1][lag+1:]))

psd = amp**2

# 绘制功率谱密度图
fig = plt.figure()

plt.plot(psd, marker='o')
plt.title('FFT of abs(return)')
plt.xlabel('Frequency / cycle per day')
plt.ylabel('Fourier Amplitude')
plt.xscale('log')
plt.yscale('log')
# plt.axis([0.001,0.5,0.1,30.])
plt.show()

# 使用回归计算指数β
```

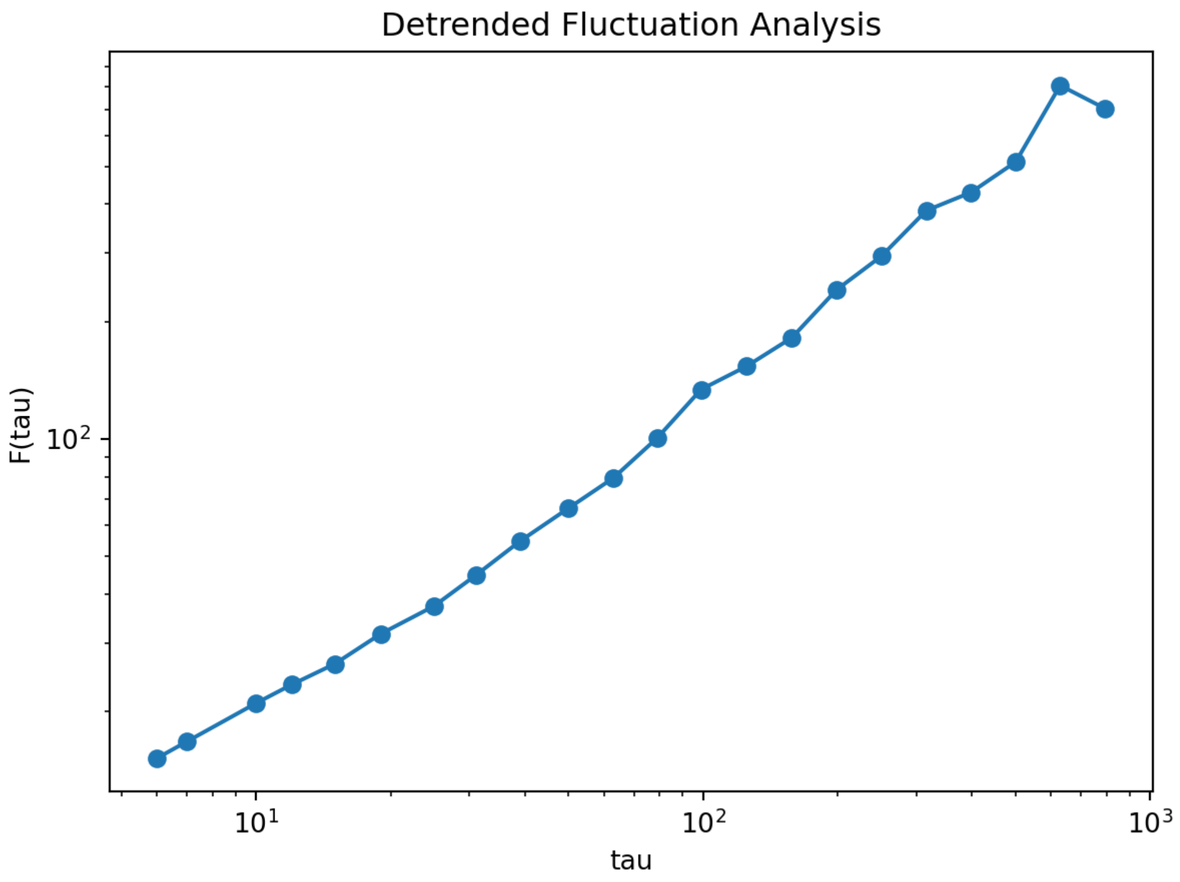


```
X = np.log10(np.arange(len(psd))+1).reshape(-1, 1)
y = np.log10(psd)
regression_model = LinearRegression().fit(X, y)
beta = -regression_model.coef_[0]

print("Exponent β:", beta)
```

```
## Exponent β: 0.7371538399359836
```

3.

```
#3

# 计算滑动窗口内的均方差
def calculate_window_fluctuations(window_data):
    x = np.arange(len(window_data))
    x_mean = (len(x) - 1) / 2
    x_squared_mean = (len(x) - 1) * (2 * len(x) - 1) / 6

    y_mean = np.mean(window_data)
    y_squared_mean = np.mean(window_data ** 2)
    xy_mean = np.dot(x, window_data) / len(x)

    numerator = y_squared_mean - y_mean ** 2 - (xy_mean - x_mean * y_mean) ** 2 / (x_squared_me
an - x_mean ** 2)
    fluctuations = np.sqrt(numerator)

    return fluctuations


# 按照步骤进行去趋势波动分析
def detrended_fluctuation_analysis(prices):
    cumulative_sum = np.cumsum(prices)
    T_max = 800

    taus = []
    fluctuations = []

    # 按照指数方式增加 tau 的取值
    start_exponent = 0.1 * int(10 * np.log10(7))
    end_exponent = 0.1 * int(10 * np.log10(T_max))

    for exponent in np.arange(start_exponent, end_exponent + 0.1, 0.1):
        tau = int(10 ** exponent)
        x = np.arange(0, len(cumulative_sum) - tau + 1, tau)
        window_fluctuations = []
        for j in range(len(x)-1):
            segments = cumulative_sum[x[j]:x[j+1]]
            window_fluctuations.append(calculate_window_fluctuations(segments))

        average_fluctuation = np.mean(window_fluctuations)

        taus.append(tau)
        fluctuations.append(average_fluctuation)

    # 对数坐标绘制图像
    fig = plt.figure()

    plt.plot(taus, fluctuations, marker='o')
    plt.xlabel('tau')
    plt.ylabel('F(tau)')
    plt.xscale('log')
    plt.yscale('log')
    plt.title('Detrended Fluctuation Analysis')
    plt.show()
    return taus,fluctuations

t,f = detrended_fluctuation_analysis(ts)
```



```
model2 = LinearRegression().fit(np.log(t).reshape(-1, 1), np.log(f))

model2.coef_[0]
```

```
## 0.8355215804720764
```

```
model2.intercept_
```

```
## 1.04130906375039
```

```
alpha = model2.coef_[0]

print("Exponent α:", alpha)
```

```
## Exponent α: 0.8355215804720764
```

```
def compare_numbers(num1, num2, precision):
    abs_diff = abs(num1 - num2)
    if abs_diff <= precision:
        return True
    else:
        return False

compare_numbers(beta, 2*alpha - 1, 0.1)
```

```
## True
```