quantitative-analysis-prices

August 29, 2023

[23]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf
import seaborn as sns
import scipy.stats as stats
```

[39]:
```
!pip install statsmodels
```

```
Requirement already satisfied: statsmodels in /usr/local/lib/python3.10/dist-
packages (0.14.0)
Requirement already satisfied: numpy>=1.18 in /usr/local/lib/python3.10/dist-
packages (from statsmodels) (1.23.5)
Requirement already satisfied: scipy!=1.9.2,>=1.4 in
/usr/local/lib/python3.10/dist-packages (from statsmodels) (1.10.1)
Requirement already satisfied: pandas>=1.0 in /usr/local/lib/python3.10/dist-
packages (from statsmodels) (1.5.3)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-
packages (from statsmodels) (0.5.3)
Requirement already satisfied: packaging>=21.3 in
/usr/local/lib/python3.10/dist-packages (from statsmodels) (23.1)
Requirement already satisfied: python-dateutil>=2.8.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.0->statsmodels) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=1.0->statsmodels) (2023.3)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
(from patsy>=0.5.2->statsmodels) (1.16.0)
```

[2]:
```
datasets = ["BAC"]


for dataset in datasets:
    Ticker = yf.Ticker(dataset)
    data = Ticker.history(start="2023-08-01", end="2023-08-28")
    filename = f"{dataset}_data.csv"
    data.to_csv(filename)
    print(f"Download data for {dataset} and saved as {filename}")
```

Download data for BAC and saved as BAC_data.csv

[3]:
```
Ticker = 'BAC'
start_date = '2023-08-01'
end_date = '2023-08-28'
data = yf.download(Ticker, start=start_date, end=end_date)
```

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

[4]:
```
Ticker = 'BAC'
start_date = '2023-08-01'
end_date = '2023-08-28'
```

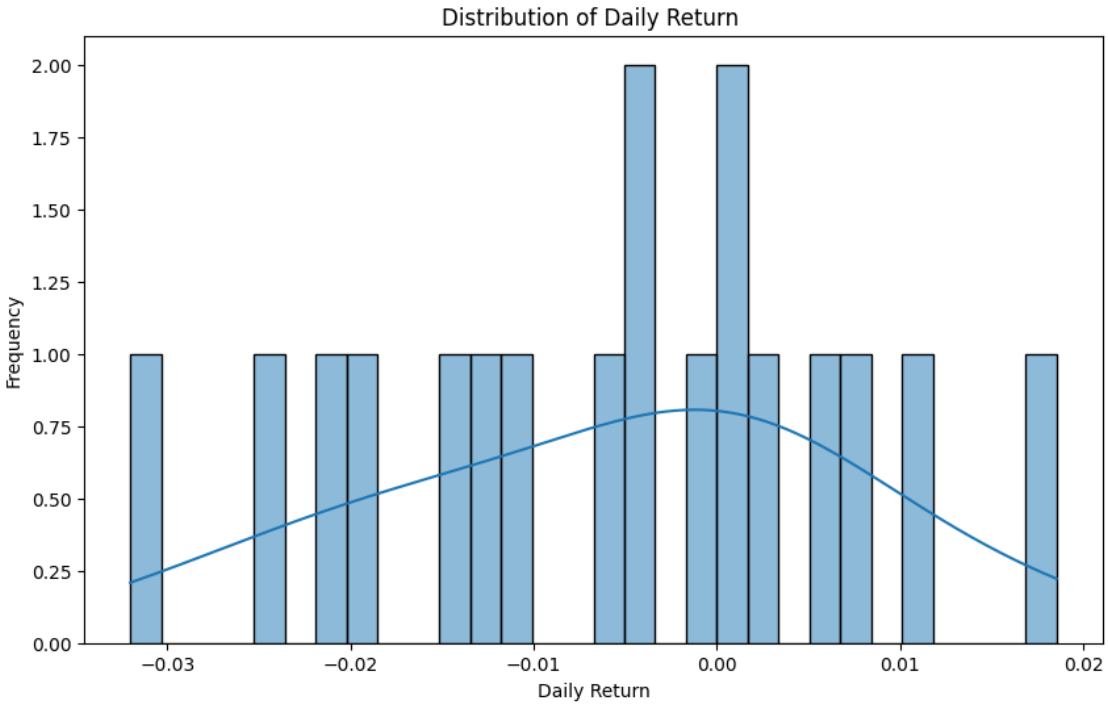
[ ]:
```
stock_data = yf.download(Ticker, start=start_date, end=end_date)
```

0.1 Descriptive Statistics

[6]:
```
stock_data['Daily_return'] = stock_data['Adj Close'].pct_change()
```

[7]:
```
statistics = stock_data['Daily_return'].describe()
```

[9]:
```
plt.figure(figsize=(10, 6))
sns.histplot(stock_data['Daily_return'].dropna(), bins=30, kde=True)
plt.title('Distribution of Daily Return ')
plt.xlabel('Daily Return')
plt.ylabel('Frequency')
plt.show()
```



[10]:
```
print("Descriptive Statistics of Daily Return:\n")
print(statistics)
```

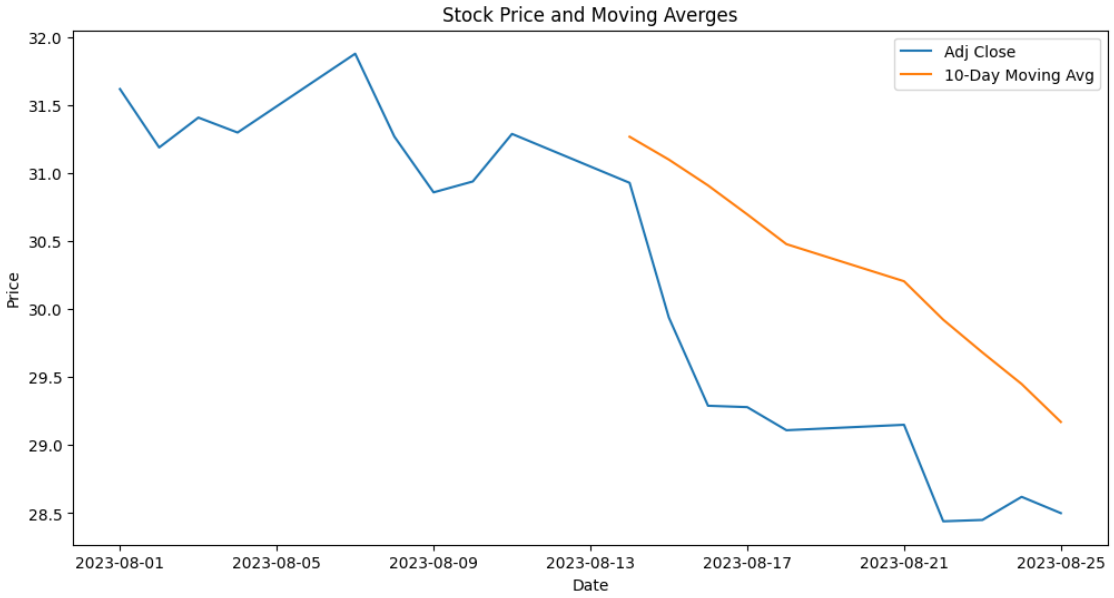
```
Descriptive Statistics of Daily Return:

count    18.000000
mean     -0.005671
std       0.013273
min      -0.032008
25%      -0.013477
50%      -0.003847
75%       0.002288
max       0.018530
Name: Daily_return, dtype: float64
```

1 Time Series Analysis

[11]:
```
stock_data['10-day MA'] = stock_data['Adj Close'].rolling(window=10).mean()
```

[12]:
```
plt.figure(figsize=(12, 6))
plt.plot(stock_data['Adj Close'], label='Adj Close')
plt.plot(stock_data['10-day MA'], label='10-Day Moving Avg')
plt.title('Stock Price and Moving Averges')
```

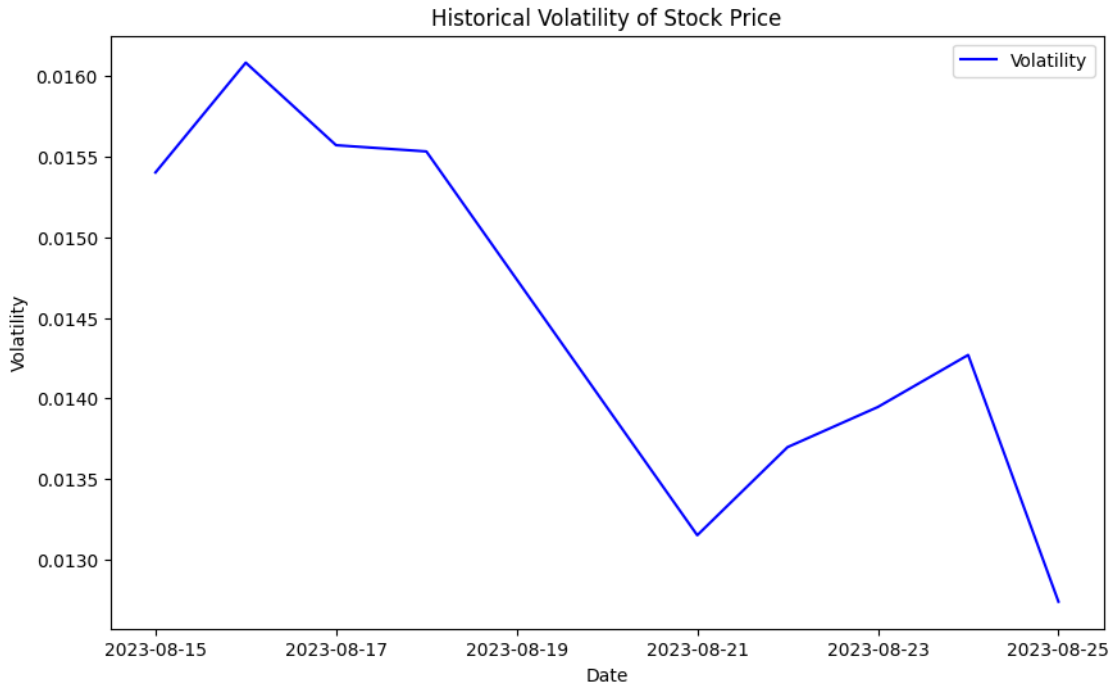
```
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()
```



2 *Volatility*

[21]:
```
window_size = 10
stock_data['Volatility'] = stock_data['Daily_return'].
  ↪rolling(window=window_size).std()
```

[22]:
```
plt.figure(figsize=(10, 6))
plt.plot(stock_data['Volatility'], color='blue', label='Volatility')
plt.title('Historical Volatility of Stock Price')
plt.xlabel('Date')
plt.ylabel('Volatility')
plt.legend()
plt.show()
```

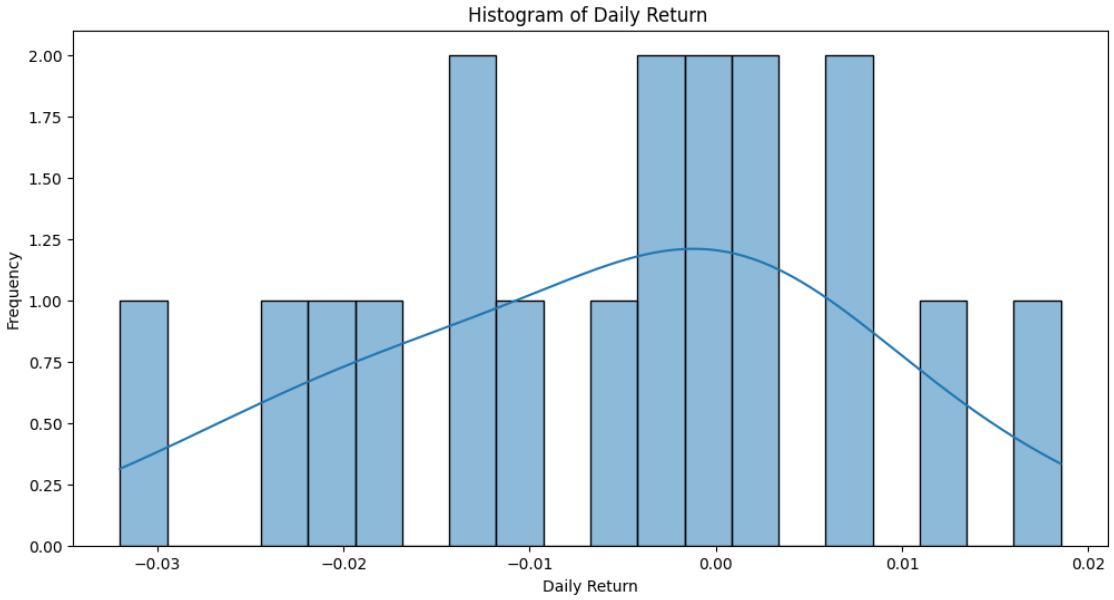


3 Statistical Tests

[25]:
```
stock_data['Daily_Return']= stock_data['Adj Close'].pct_change()
shapiro_test_statistic, shapiro_p_value = stats.
 ↪shapiro(stock_data['Daily_Return'].dropna())
normality_threshold = 0.05
```

[26]:
```
t_statistic, t_p_value = stats.ttest_1samp(stock_data['Daily_Return'].dropna(),␣
 ↪0)
```

[27]:
```
plt.figure(figsize=(12, 6))
sns.histplot(stock_data['Daily_Return'].dropna(), bins=20, kde=True)
plt.title('Histogram of Daily Return ')
plt.xlabel('Daily Return')
plt.ylabel('Frequency')
plt.show()
```



[28]:
```
print("Shapiro-Wilk Test for Normality:")
print(f"test Statistics : {shapiro_test_statistic}")
print(f"P-value: {shapiro_p_value}")
if shapiro_p_value < normality_threshold:
   print("The data is not normally distributed.")
else:
   print("The data is normally distributed.")
```

```
Shapiro-Wilk Test for Normality:
test Statistics : 0.9846869111061096
P-value: 0.985351026058197
The data is normally distributed.
```

[30]:
```
print("\nOne-sample T-test:")
print(f"T-statistic: {t_statistic}")
print(f"P-value : {t_p_value}")
if t_p_value < normality_threshold:
   print("Reject the null hypothesis (mean is not zero).")
else:
   print("Fail to reject the null hypothesis (mean is zero).")
```

```
One-sample T-test:
T-statistic: -1.8127563826244655
P-value : 0.08756678870762409
Fail to reject the null hypothesis (mean is zero).
```