```
import pandas as pd
import numpy as np
import yfinance as yf
import matplotlib.pyplot as plt
import datetime
import seaborn as sns
import math
import scipy.stats as stats
import matplotlib


url = 'https://en.wikipedia.org/wiki/Euro_Stoxx_50'
tables = pd.read_html(url)[4]
eurostoxx50 = tables['Ticker'].to_list()
eurostoxx50.append('^STOXX50E')
# eurostoxx50


# download data for each stock
start_date = '2010-01-01'
end_date = datetime.datetime.now().strftime('%Y-%m-%d')
data = {}
for stock in eurostoxx50:
    data[stock] = yf.download(stock, start=start_date, end=end_date)['Adj Close'
```

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

```
euro_stoxx_50 = pd.DataFrame(data)
euro_stoxx_50.head()
```

	ADS.DE	ADYEN.AS	AD.AS	AI.PA	AIR.PA	ALV.DE	ABI.BR	AS
Date								
2010-01-04	32.097492	NaN	5.786551	34.519089	11.523581	45.519127	27.064257	27.
2010-01-05	33.110306	NaN	5.738072	33.828705	11.486514	45.657921	26.724808	27.
2010-01-06	32.843559	NaN	5.770597	33.620365	11.589478	46.012669	26.531357	27.
2010-01-07	33.131157	NaN	5.644186	33.579517	11.704795	45.483131	26.078762	27.
2010-01-08	33.010281	NaN	5.702482	33.685722	11.820115	45.236355	25.779476	26.

5 rows × 51 columns

```
euro_stoxx_50.to_csv('euro_stoxx_50.csv')
```

```
def log_returns(data):
    log_returns = np.log(data/data.shift(1))*100
    return log_returns
```

```
log_returns = log_returns(euro_stoxx_50)
```

```
log_returns.head()
```

	ADS.DE	ADYEN.AS	AD.AS	AI.PA	AIR.PA	ALV.DE	ABI.BR	ASM
Date								
2010-01-04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2010-01-05	3.106668	NaN	-0.841310	-2.020277	-0.322176	0.304450	-1.262165	0.18
2010-01-06	-0.808893	NaN	0.565220	-0.617771	0.892392	0.773966	-0.726495	1.06
2010-01-07	0.871848	NaN	-2.214938	-0.121571	0.990100	-1.157524	-1.720604	-1.84
2010-01-08	-0.365509	NaN	1.027540	0.315780	0.980417	-0.544045	-1.154259	-3.41

5 rows × 51 columns

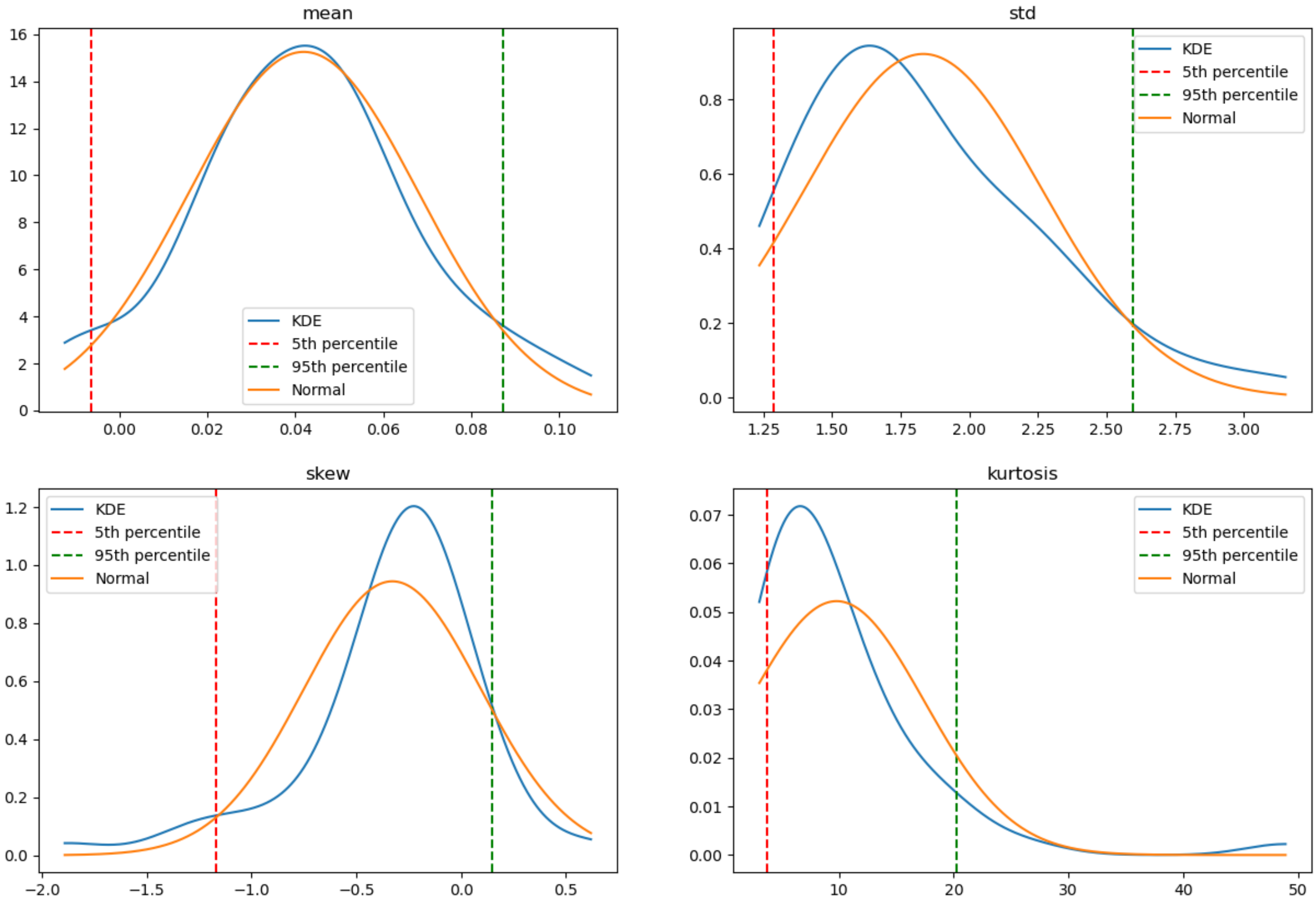
```
def calculate_moments(data):
    stats = pd.DataFrame(columns=['mean', 'variance', 'skew', 'kurtosis'])
    for ticker in data:
        row = data[ticker].dropna()
        T = len(row)
        # print(T)
        if T == 0:
            print("Am I even useful?")
            stats.loc[ticker] = [np.nan, np.nan, np.nan, np.nan]
        else:
            mu=(1/T)*np.sum(row)
            sig2 = 1/(T-1) * np.sum([np.square((x - mu)) for x in row])
            sk = 1/((T-1) * math.pow(math.sqrt(sig2),3)) * np.sum([math.pow(x -
            ku = 1/((T-1) * math.pow(math.sqrt(sig2),4)) * np.sum([math.pow(x -
            stats.loc[ticker] = [mu, sig2, sk, ku]

    return stats


stats_df = calculate_moments(log_returns)
```

```
stats_df.head()
```

	mean	variance	skew	kurtosis
ADS.DE	0.057064	3.530545	0.195313	9.080755
ADYEN.AS	0.078179	9.934566	-1.890501	48.925537
AD.AS	0.043959	1.569733	-0.305689	5.802424
AI.PA	0.046819	1.668186	-0.199129	4.700700
AIR.PA	0.074330	4.583646	-0.369811	13.802397

```
def plot_moments(stats_data):
    fig, ax = plt.subplots(2,2, figsize=(15,10))
    for i, col in enumerate(stats_data.columns):
        m = stats_data[col].dropna().mean()
        s = stats_data[col].dropna().std()
        kde = stats.gaussian_kde(stats_data[col].dropna())
        x = np.linspace(np.min(stats_data[col]), np.max(stats_data[col]), 1000)

        ax[int(i/2), i%2].plot(x, kde(x), label='KDE')
        percentile_5 = np.percentile(stats_data[col], 5)
        percentile_95 = np.percentile(stats_data[col], 95)
        ax[int(i/2), i%2].axvline(percentile_5, color='r', linestyle='--', labe
        ax[int(i/2), i%2].axvline(percentile_95, color='g', linestyle='--', lab

        # normal distribution
        norm = stats.norm.pdf(x, m, s)
        ax[int(i/2), i%2].plot(x, norm, label='Normal')
        ax[int(i/2), i%2].set_title(col)
        ax[int(i/2), i%2].legend()
    plt.show()
```

```
plot_moments(stats_df)
```



The mean of means is very normal compared to the s&p500 with not very fat tails.

Standard deviation can also be considered close to the one-sided normal distribution but with less fatter tails than the s&p500.

Most of the log returns evenly spread out to smaller returns than the bigger negative returns.

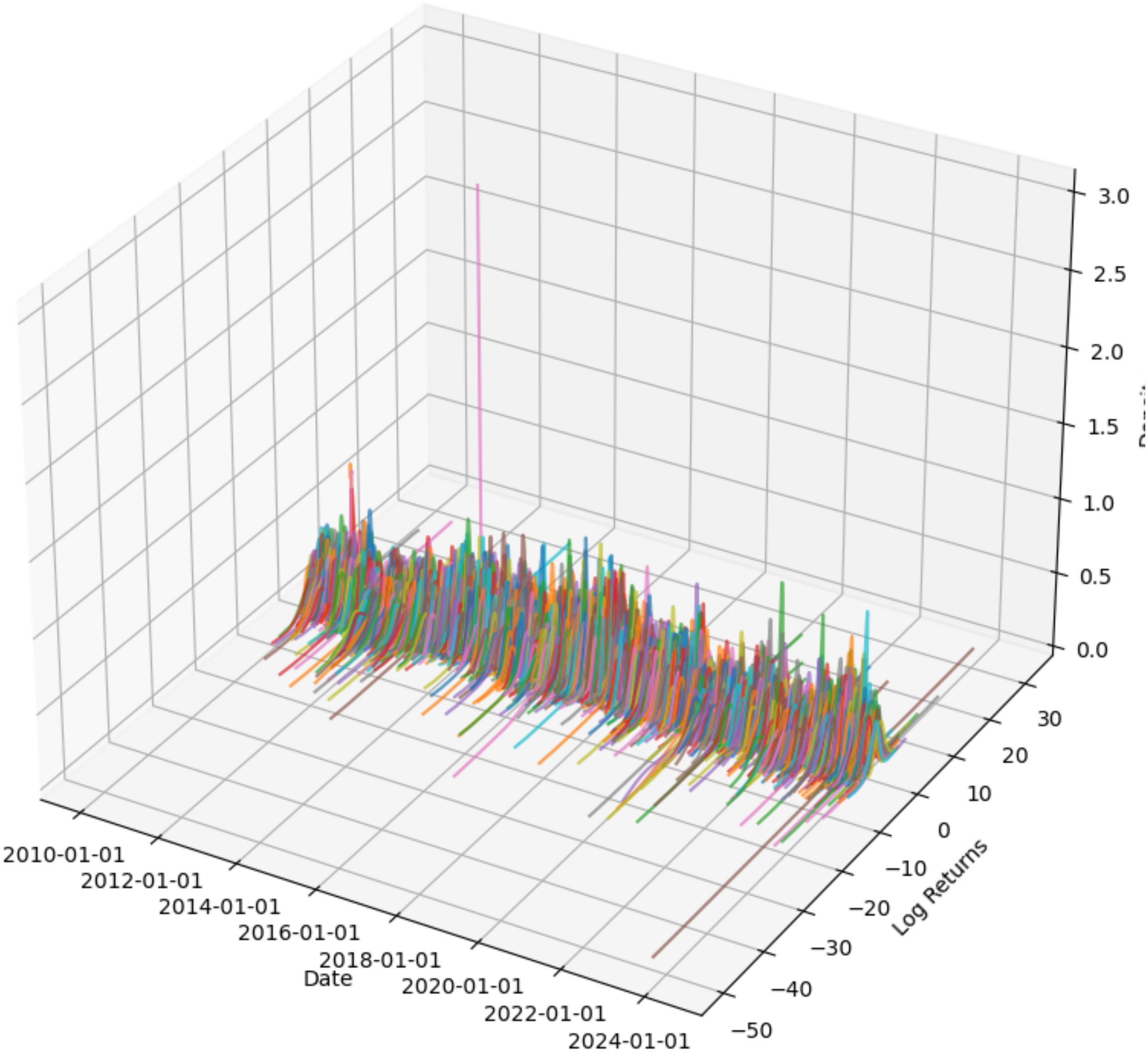
The kurtosis shows huge values so, most log returns have fat tails.

Q2

```
import matplotlib.dates as mdates
from numpy.linalg import LinAlgError

def plot_3d(data):
    fig = plt.figure(figsize=(15,10))
    ax = fig.add_subplot(111, projection='3d')

    for i, (date, row) in enumerate(data.iterrows()):
        values = row.dropna().values
        if len(values) > 1:
            try:
                kde = stats.gaussian_kde(values)
                x = np.linspace(np.min(values), np.max(values), 1000)
                density = kde(x)
                date_num = matplotlib.dates.date2num(date)
                ax.plot([date_num]*len(x), x, density, label=row[0], alpha=0.7)
            except LinAlgError as e:
                print(f"Error for {i} {date} \nERROR: {e}")
                continue


    ax.xaxis.set_major_locator(mdates.AutoDateLocator())
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
    ax.set_xlabel('Date')
    ax.set_ylabel('Log Returns')
    ax.set_zlabel('Density')
```

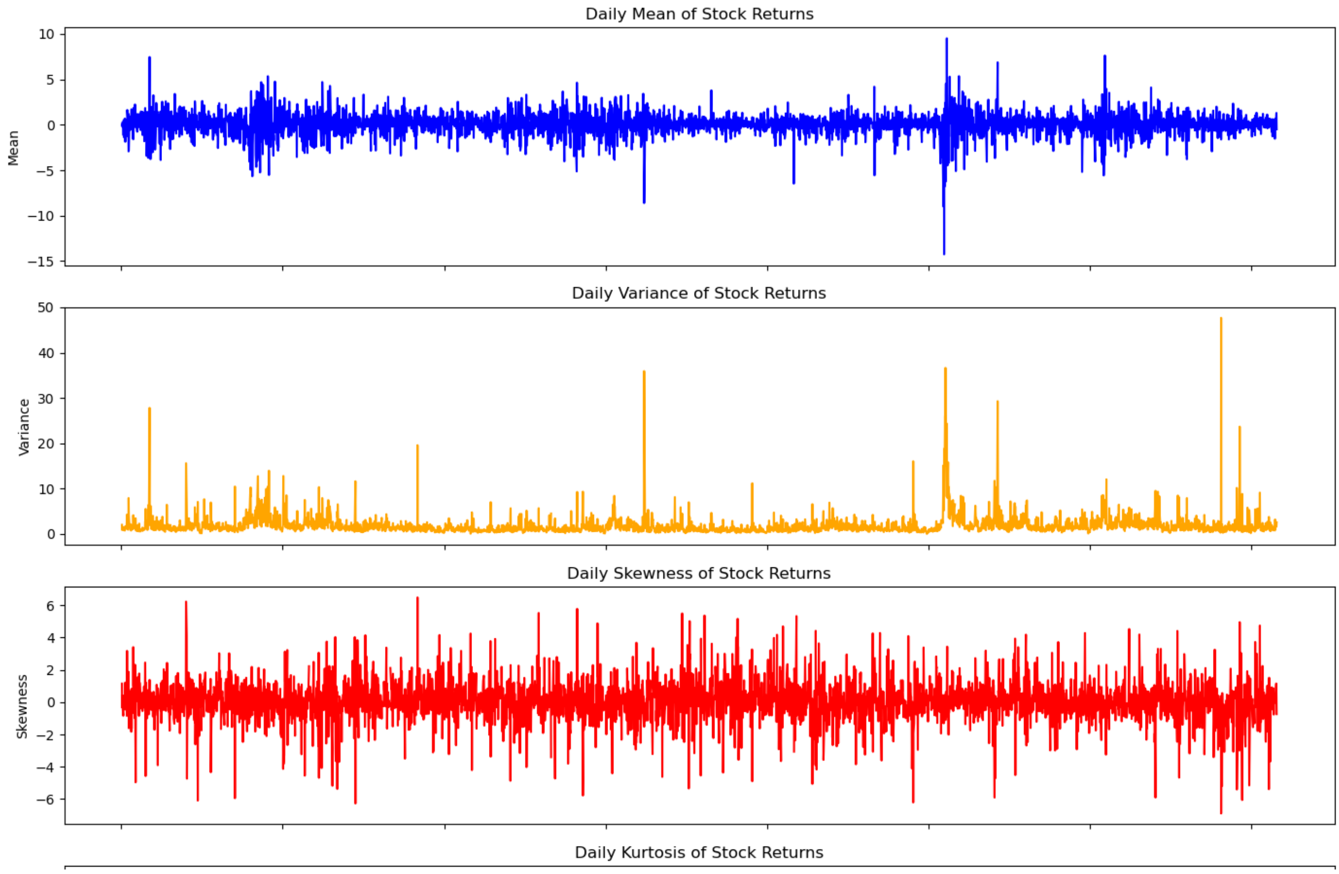
```
plot_3d(log_returns)
```

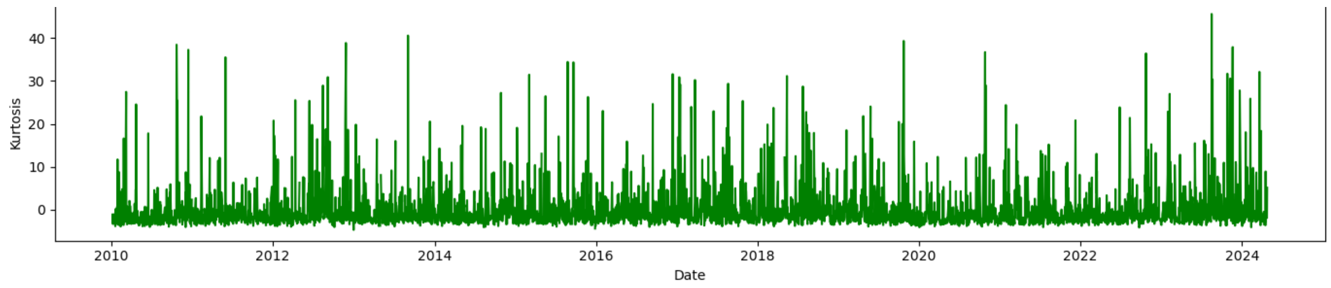
```
/var/folders/jl/1kyt0fzd167dx2t_r6lh3tw00000gn/T/ipykernel_42705/2693115732
  ax.plot([date_num]*len(x), x, density, label=row[0], alpha=0.7)
Error for 2563 2019-12-25 00:00:00
ERROR: The data appears to lie in a lower-dimensional subspace of the space
```




```
def moments_timeseries(data):
```

```
    daily_mean = data.mean(axis=1)
    daily_variance = data.var(axis=1)
    daily_skewness = data.skew(axis=1)
    daily_kurtosis = data.kurtosis(axis=1)-3

    fig, axes = plt.subplots(4, 1, figsize=(14, 12), sharex=True)

    axes[0].plot(data.index, daily_mean, label='Daily Mean', color='blue')
    axes[0].set_title('Daily Mean of Stock Returns')
    axes[0].set_ylabel('Mean')

    axes[1].plot(data.index, daily_variance, label='Daily Variance', color='ora
    axes[1].set_title('Daily Variance of Stock Returns')
    axes[1].set_ylabel('Variance')

    axes[2].plot(data.index, daily_skewness, label='Daily Skewness', color='rec
    axes[2].set_title('Daily Skewness of Stock Returns')
    axes[2].set_ylabel('Skewness')

    axes[3].plot(data.index, daily_kurtosis, label='Daily Kurtosis', color='gre
    axes[3].set_title('Daily Kurtosis of Stock Returns')
    axes[3].set_ylabel('Kurtosis')

    axes[3].set_xlabel('Date')

    plt.tight_layout()
    plt.show()

moments_timeseries(log_returns)
```





40
30
20
10
0
Kurtosis
2010
2012
2014
2016
2018
2020
2022
2024
Date

The mean has been comparatively static fro 2017-2020 and otherwise very volatile.

The variance is lesser than s&p500, so the stocks in eurostoxx have similar trends resulating in lesser daily variance compared to the s&p500. Maybe this could alsoe be because of the size of the index.

Similarly for skewness the values vary a lot from 0 compared to the s&p500.

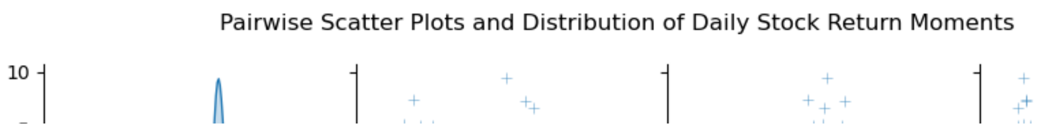
The kurtosis in s&p500 has very large extreme values compared to eurostoxx50. This again could be because of the size of the index.

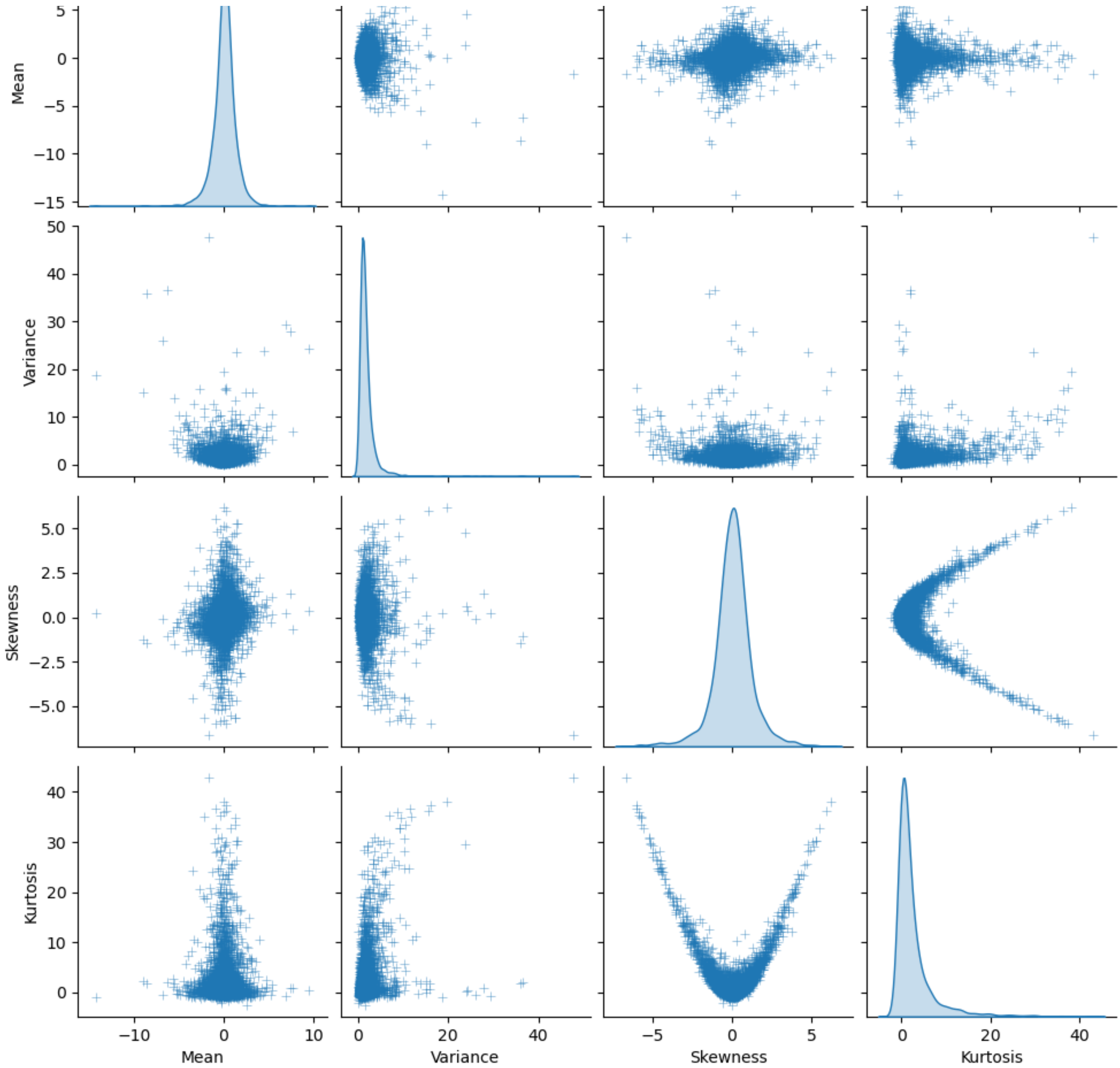
Q3

```
def calculate_statistics(returns):
    T = len(returns)
    if T == 0:
        return np.nan, np.nan, np.nan, np.nan  # Handle empty data

    mu = (1/T) * np.sum(returns)
    sig2 = np.var(returns, ddof=1)
    sk = (1 / ((T - 1) * np.power(np.sqrt(sig2), 3))) * np.sum(np.power(returns
    ku = (1 / ((T - 1) * np.power(np.sqrt(sig2), 4))) * np.sum(np.power(returns

    return mu, sig2, sk, ku


# Calculate daily statistics
daily_stats = log_returns.apply(lambda x: calculate_statistics(x.dropna()), axi

# Convert the list of tuples into a DataFrame
daily_stats_df = pd.DataFrame(list(daily_stats), index=log_returns.index, colum
```

```
    /var/folders/jl/1kyt0fzd167dx2t_r6lh3tw00000gn/T/ipykernel_42705/407290991.
      sk = (1 / ((T - 1) * np.power(np.sqrt(sig2), 3))) * np.sum(np.power(retur
    /var/folders/jl/1kyt0fzd167dx2t_r6lh3tw00000gn/T/ipykernel_42705/407290991.
      sk = (1 / ((T - 1) * np.power(np.sqrt(sig2), 3))) * np.sum(np.power(retur
    /var/folders/jl/1kyt0fzd167dx2t_r6lh3tw00000gn/T/ipykernel_42705/407290991.
      ku = (1 / ((T - 1) * np.power(np.sqrt(sig2), 4))) * np.sum(np.power(retur
    /var/folders/jl/1kyt0fzd167dx2t_r6lh3tw00000gn/T/ipykernel_42705/407290991.
      ku = (1 / ((T - 1) * np.power(np.sqrt(sig2), 4))) * np.sum(np.power(retur
```

```
sns.pairplot(daily_stats_df, diag_kind='kde', markers='+', plot_kws={'alpha': 0
plt.suptitle('Pairwise Scatter Plots and Distribution of Daily Stock Return Mom
plt.show()
```





Mean
Variance
Skewness
Kurtosis
Mean
Variance
Skewness
Kurtosis

Mean vs. Variance

There's a discernible trend where points spread outwards in a funnel shape as we move right or left, this indicates heteroskedasticity — variance increasing with the absolute mean. This suggests that days with higher average returns may also exhibit greater variability or risk, which is a common pattern in financial returns due to volatile markets or specific events impacting stock prices.

Mean vs. Skewness

The points are spread around the center, which could indicate that the relationship between average returns and asymmetry of returns is weak or complex. A lack of strong pattern here suggests that the mean return does not reliably predict the direction or magnitude of skewness on a given day.But most distributions have high absolute skewness and high absolute mean at the same time.

Mean vs. Kurtosis

The distributions which have means close to 0 have larger likelihood to have fatter/heavier tails, though most distribtuions have mean close to 0 and large kurtosis. And the distributions which have more extreme means have smaller likelihood to have fatter/heavier tails. It means that on most very bad/good days, most stocks changed in roughly the same direction.

Variance vs. Skewness

The scatter implies that the variance of 500 returns doesn't necessarily predict how asymmetric the return distribution is.

Variance vs. Kurtosis

Most Kurtosis are very large on most days, so it is hard to find out the relationship between this two. In a range, there is a positive relationship. But it is rare to see really extreme variance and really extreme kurtosis.

Skewness vs. Kurtosis

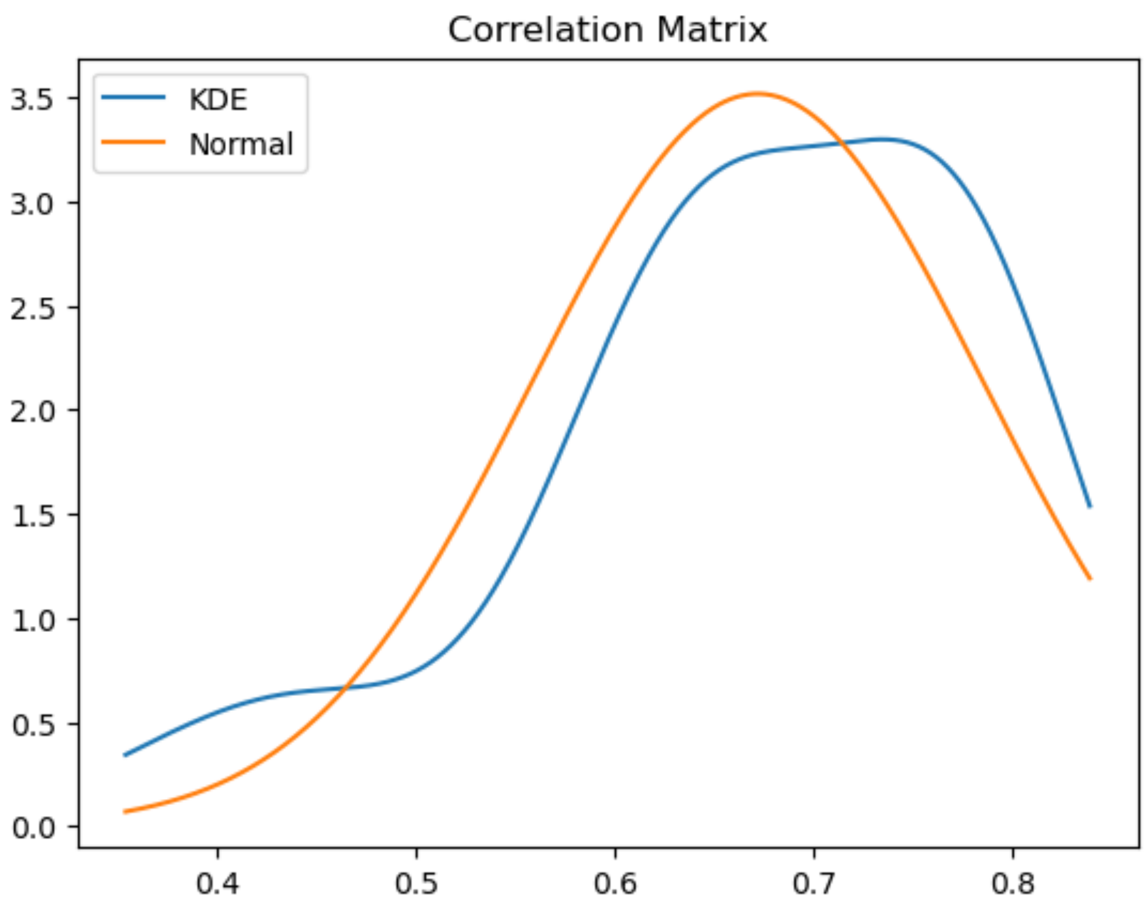
Like the relationship between variance and mean, higher moments are not independent.Larger absolute values of skewness always coexists with larger kurtosis. It indicates that the days with asymmetric return distributions (skewed to the right or left) also tend to have heavier or lighter tails than a normal distribution. This could have implications for the probability of extreme returns.

Q4

```
correlation_matrix = log_returns.drop(columns=["^STOXX50E"]).corrwith(log_retu
correlation_matrix
```

```
ALV.DE       0.838751
CS.PA        0.820592
SU.PA        0.809065
SIE.DE       0.797569
BNP.PA       0.796951
INGA.AS      0.795481
BAS.DE       0.787166
SGO.PA       0.784273
DG.PA        0.784196
SAN.MC       0.781938
MBG.DE       0.767459
ISP.MI       0.764728
AI.PA        0.763942
BBVA.MC      0.763709
MC.PA        0.761496
ENEL.MI      0.741417
BMW.DE       0.740650
ENI.MI       0.740649
MUV2.DE      0.738924
DHL.DE       0.726131
NDA-FI.HE    0.721089
TTE.PA       0.712467
IBE.MC       0.688821
UCG.MI       0.684882
STLAM.MI     0.681547
DTE.DE       0.678859
OR.PA        0.676840
SAP.DE       0.664178
KER.PA       0.662690
BAYN.DE      0.657975
ITX.MC       0.651567
IFX.DE       0.649196
ABI.BR       0.637554
AIR.PA       0.636189
SAF.PA       0.635167
VOW.DE       0.633604
RACE.MI      0.626352
EL.PA        0.621811
ASML.AS      0.598697
SAN.PA       0.590526
DB1.DE       0.587306
BN.PA        0.587218
ADS.DE       0.584819
RI.PA        0.579540
RMS.PA       0.518443
NOKIA.HE     0.476557
PRX.AS       0.456521
ADYEN.AS     0.422529
AD.AS        0.406663
FLTR.L       0.353953
dtype: float64
```

```
kde = stats.gaussian_kde(correlation_matrix.dropna())
x = np.linspace(np.min(correlation_matrix), np.max(correlation_matrix), 1000)
density = kde(x)

plt.plot(x, density, label='KDE')
# normal distribution
m = correlation_matrix.mean()
s = correlation_matrix.std()
norm = stats.norm.pdf(x, m, s)
plt.plot(x, norm, label='Normal')
plt.legend()
plt.title('Correlation Matrix')
plt.show()
```



Compared to s&p500 the eurostoxx50 is more correlated to the stocks.

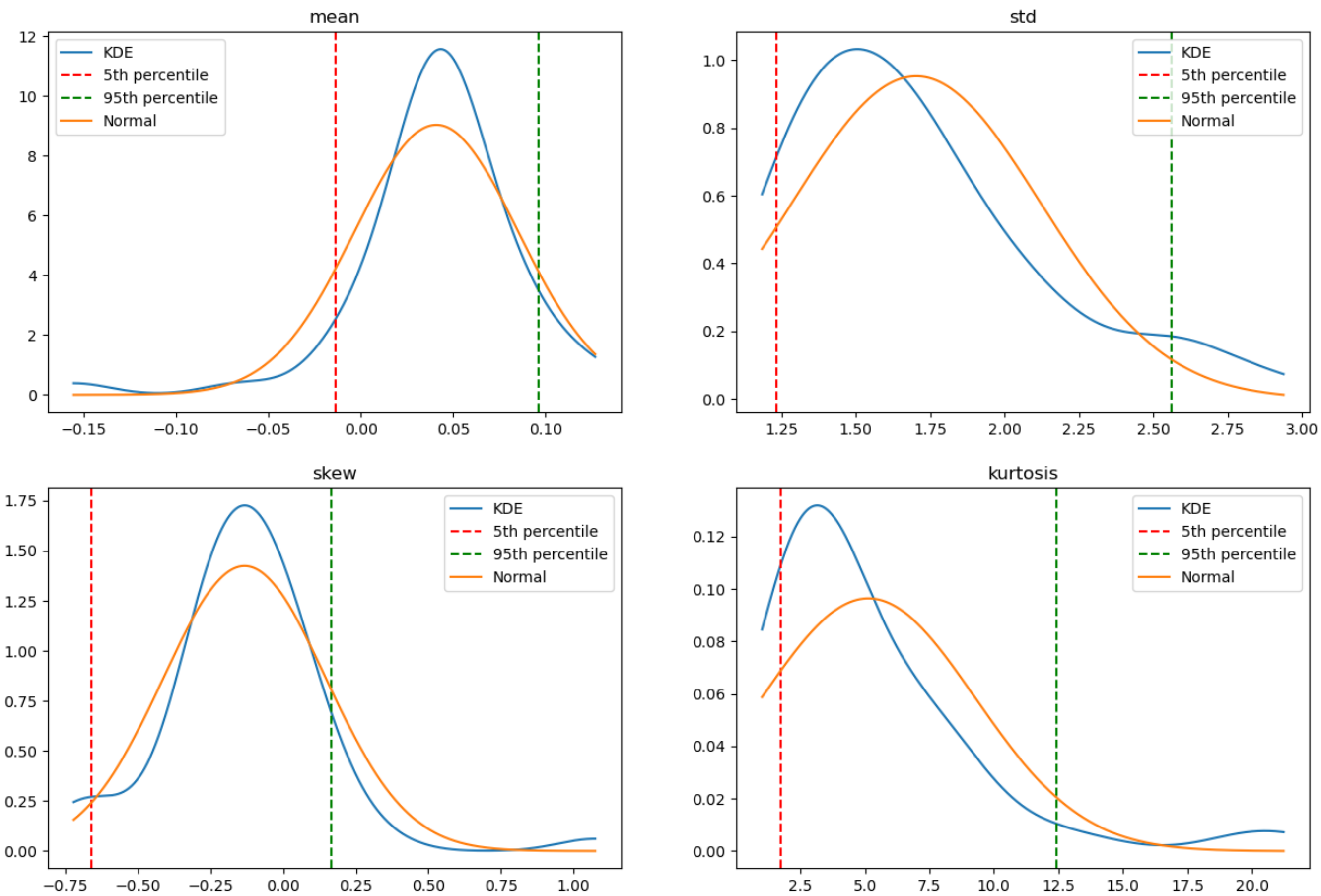
Q5

```
euro_stoxx_50_2020 = euro_stoxx_50.loc[euro_stoxx_50.index < '2020-01-01']
```

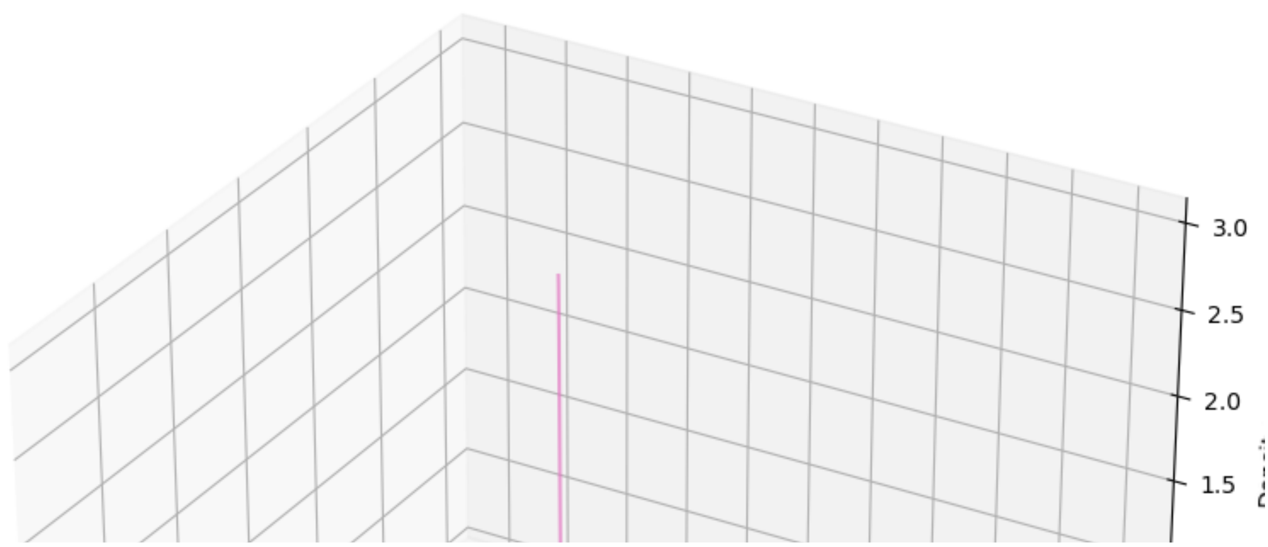
```
log_returns_2020 = log_returns.loc[log_returns.index < '2020-01-01']
```

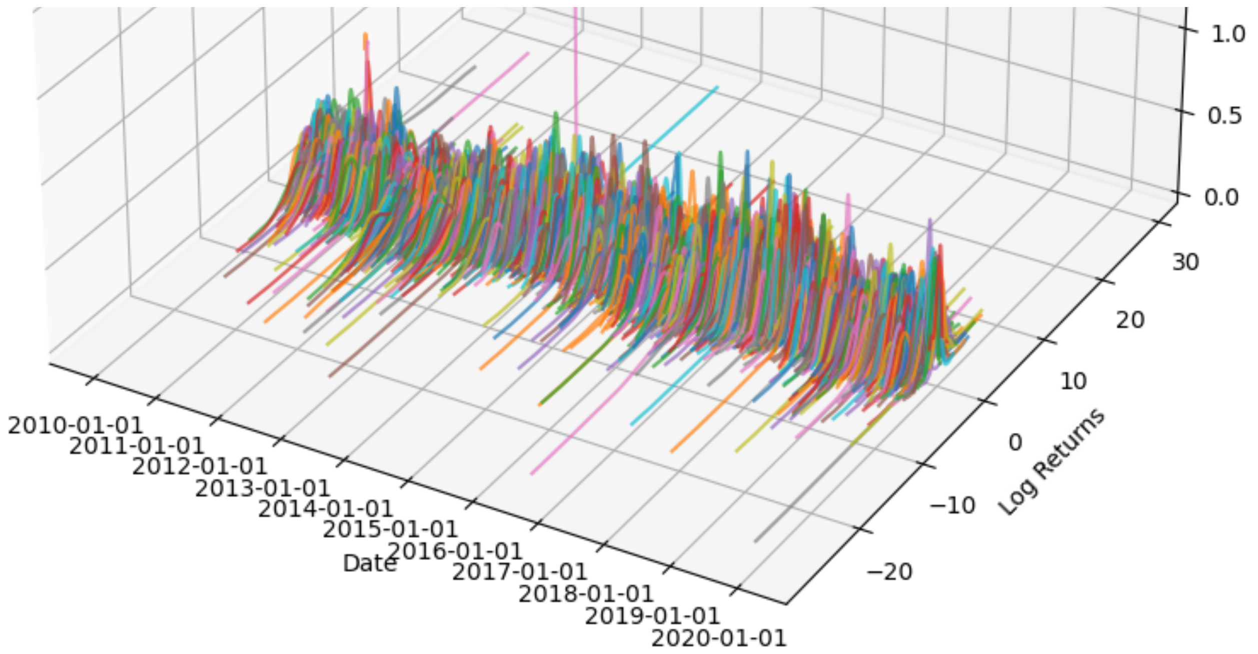
data before 2020

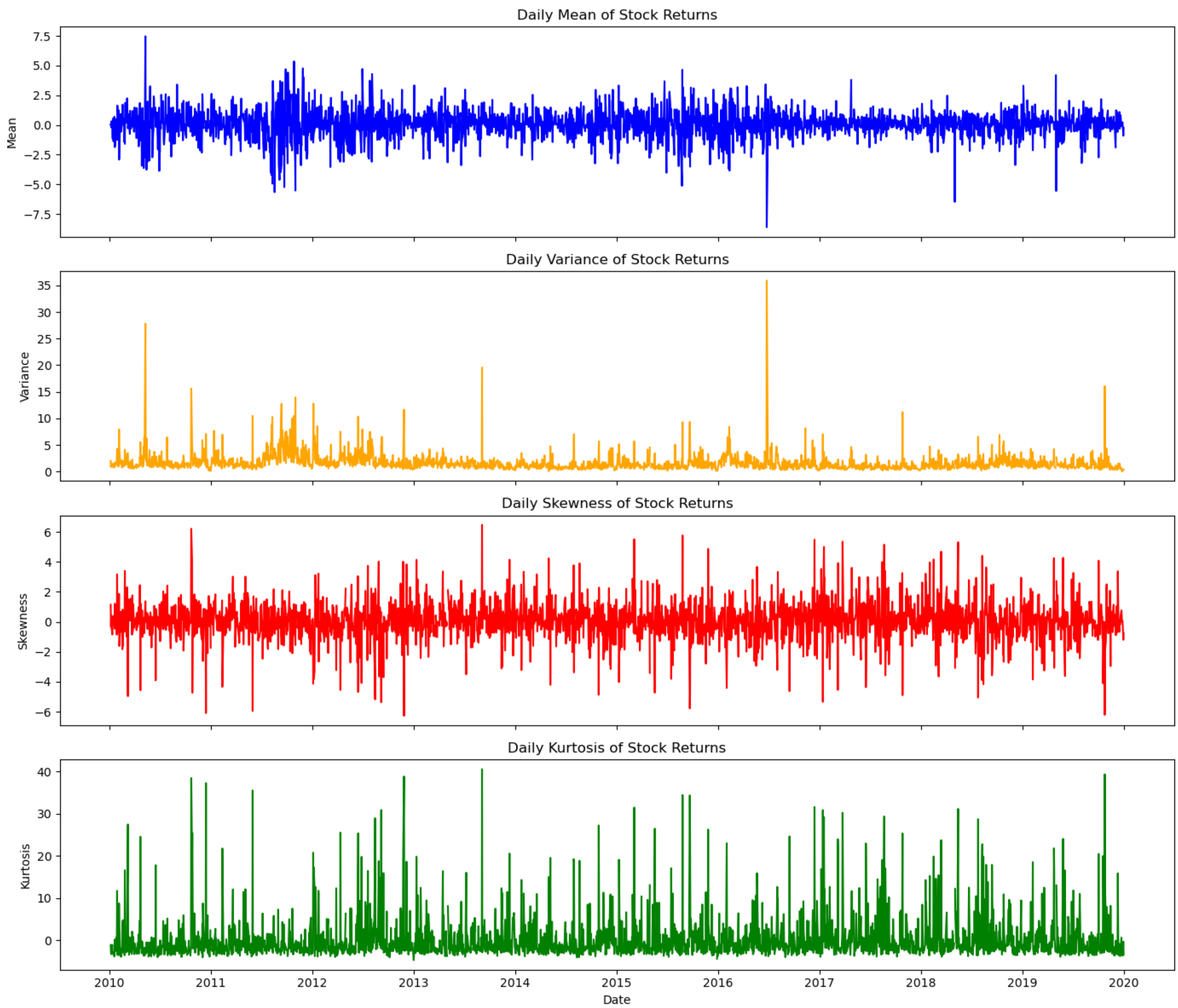
```
stats_df_2020 = calculate_moments(log_returns_2020)
plot_moments(stats_df_2020)
plot_3d(log_returns_2020)
moments_timeseries(log_returns_2020)
```




```
/var/folders/jl/1kyt0fzd167dx2t_r61h3tw00000gn/T/ipykernel_42705/3553293981
  ax.plot([date_num]*len(x), x, density, label=row[0], alpha=0.7)
Error for 2563 2019-12-25 00:00:00
ERROR: The data appears to lie in a lower-dimensional subspace of the space
```

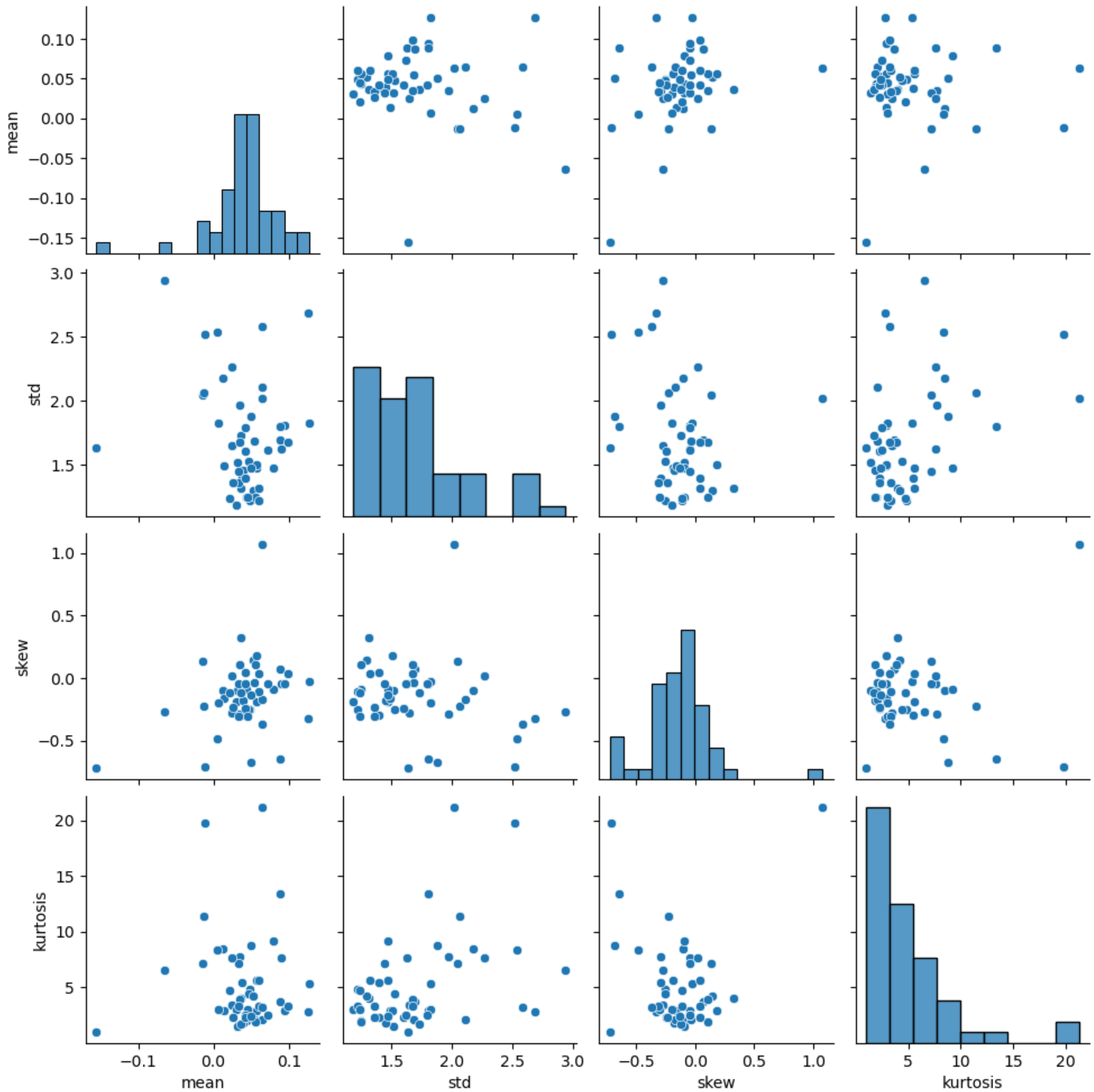



Date
Log Returns
2010-01-01
2011-01-01
2012-01-01
2013-01-01
2014-01-01
2015-01-01
2016-01-01
2017-01-01
2018-01-01
2019-01-01
2020-01-01
−20
−10
0
10
20
30
0.0
0.5
1.0


Daily Mean of Stock Returns
Mean
Daily Variance of Stock Returns
Variance
Daily Skewness of Stock Returns
Skewness
Daily Kurtosis of Stock Returns
Kurtosis
Date
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020

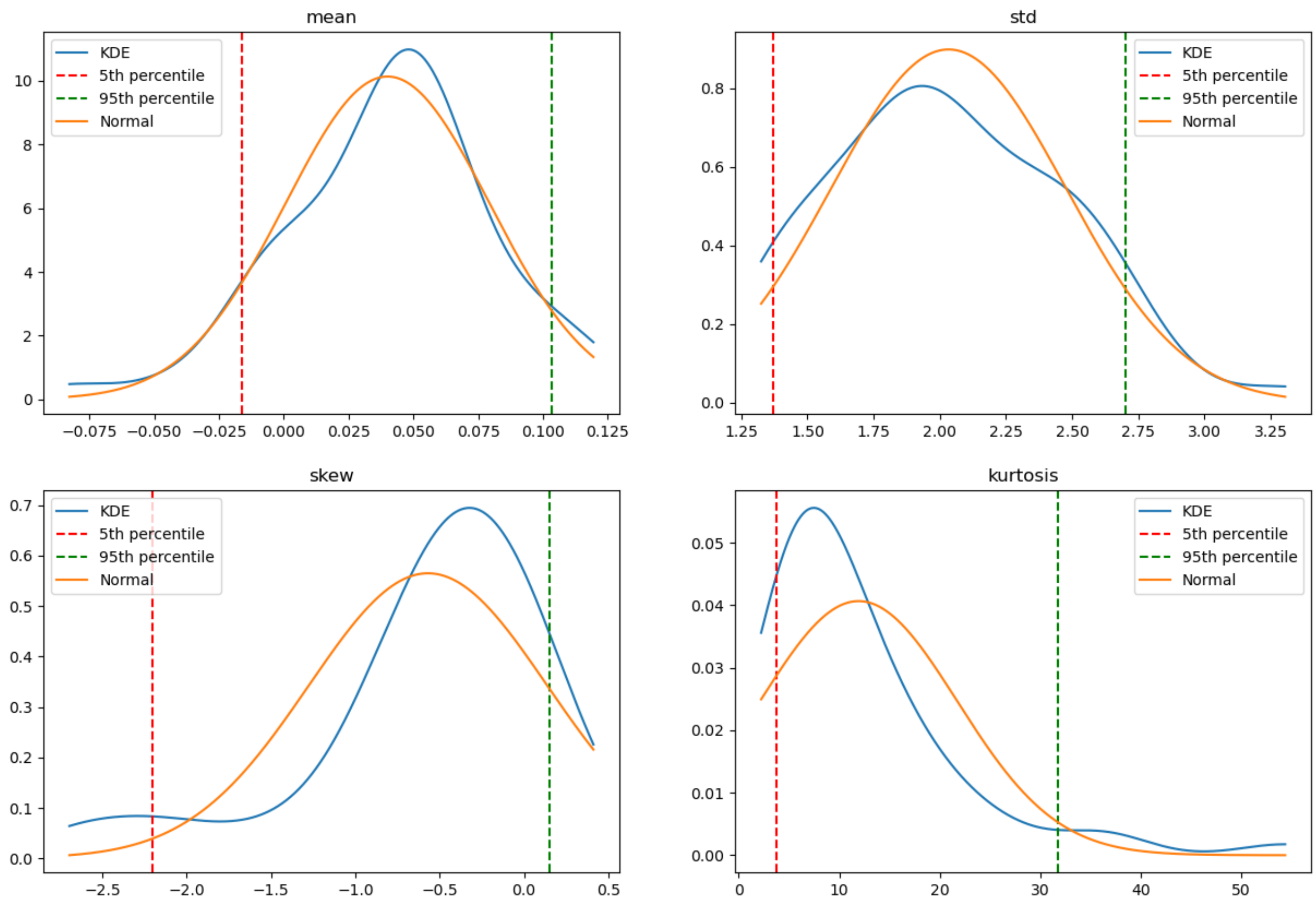
```
sns.pairplot(stats_df_2020.dropna(), kind='scatter')
```

```
&lt;seaborn.axisgrid.PairGrid at 0x13b19a550&gt;
```



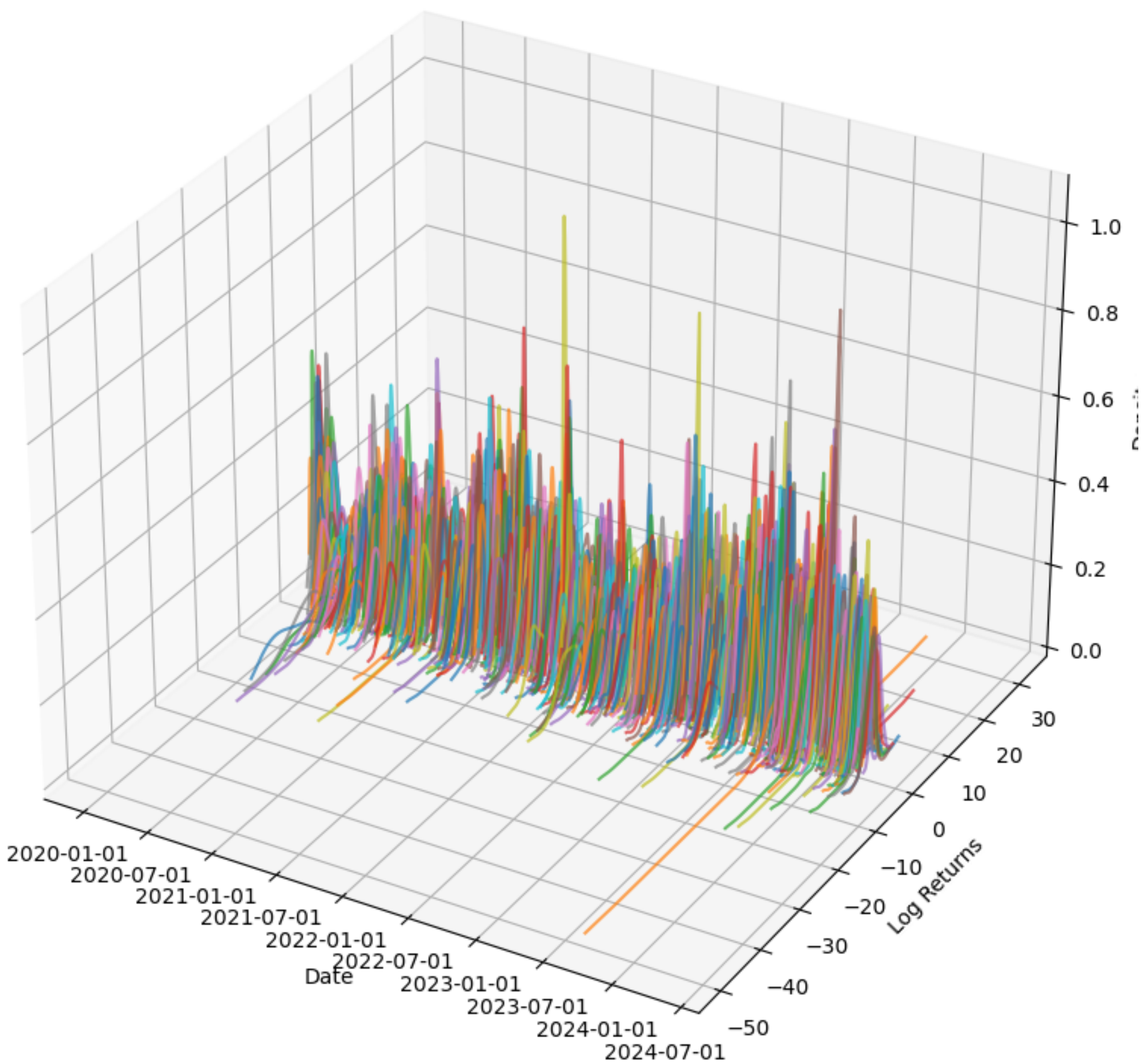
data from 2020 to current

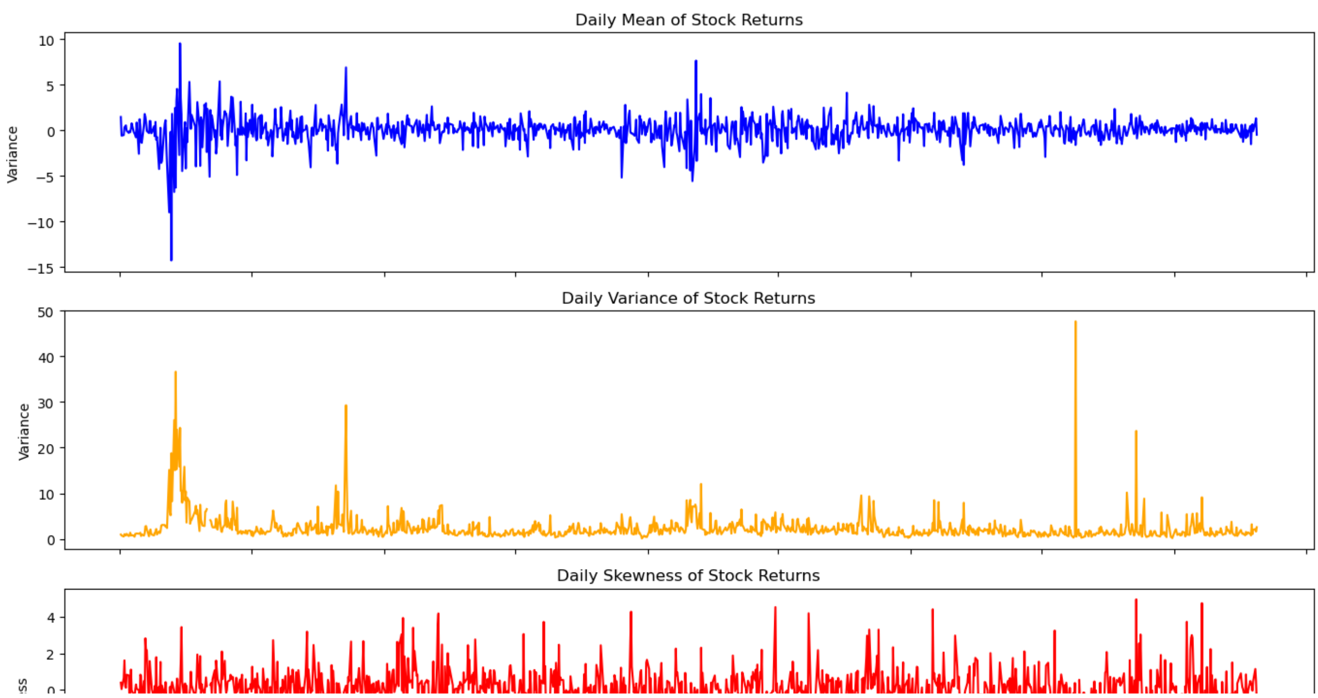
```
euro_stoxx_50_p_2020 = euro_stoxx_50.loc[euro_stoxx_50.index >= '2020-01-01']
log_returns_p_2020 = log_returns.loc[log_returns.index >= '2020-01-01']


stats_p_2020 = calculate_moments(log_returns_p_2020)
plot_moments(stats_p_2020)
plot_3d(log_returns_p_2020)
moments_timeseries(log_returns_p_2020)
sns.pairplot(stats_p_2020.dropna(), kind='scatter')
```

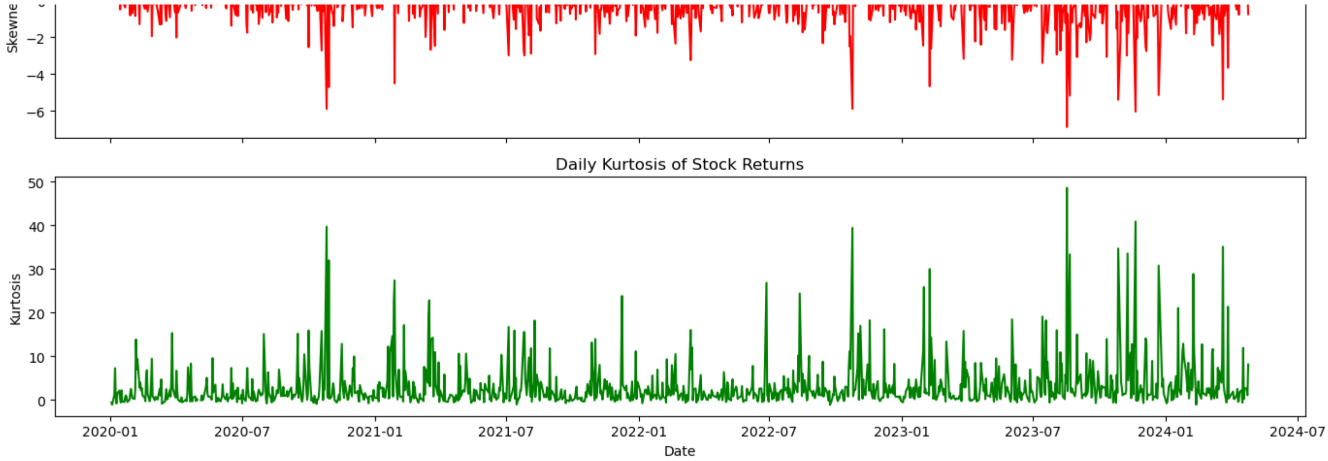



```
/var/folders/jl/1kvt0fzd167dx2t_r6lb3tw00000gp/T/ipykernel_42705/3553293981
```

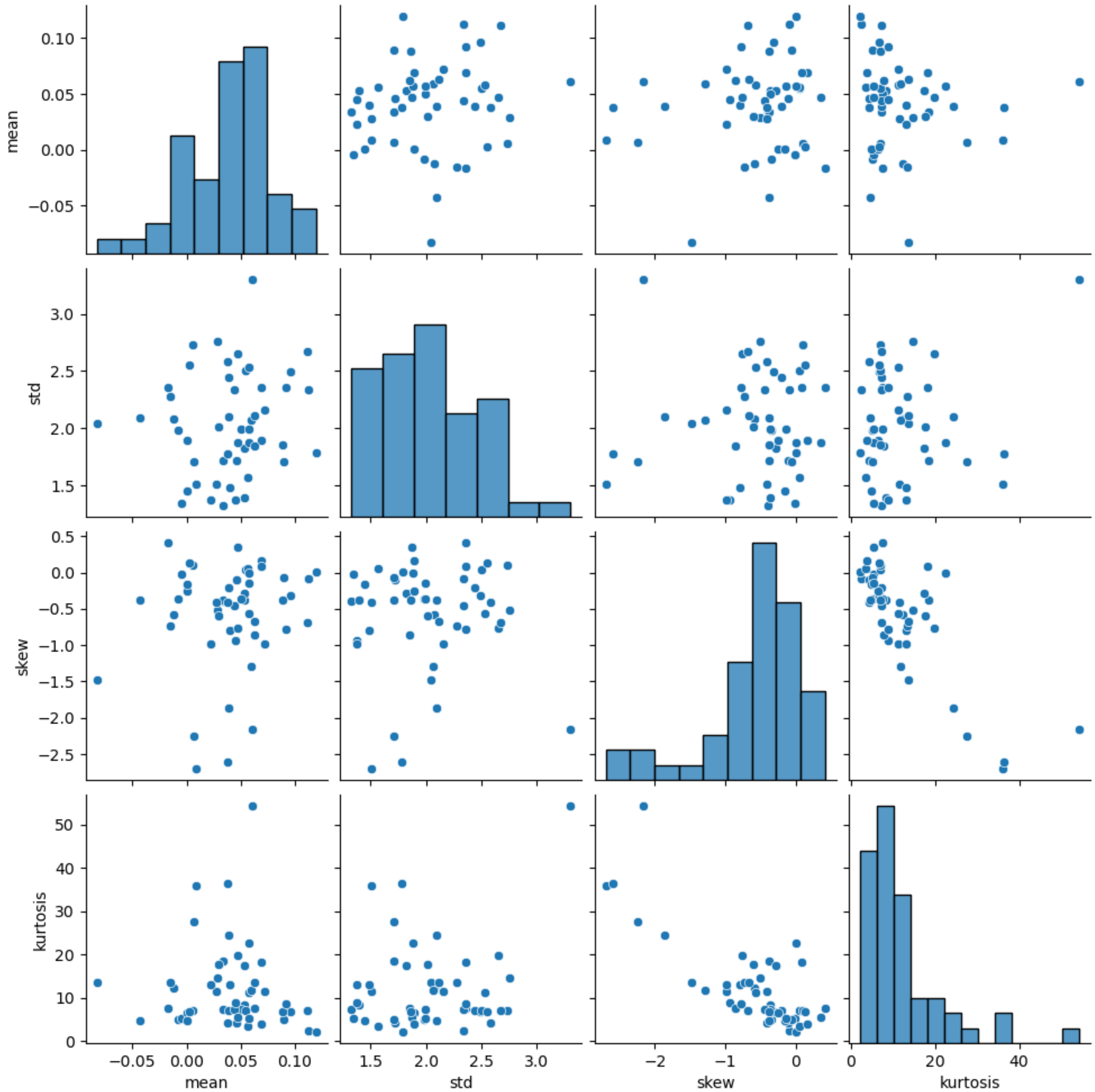
```
  ax.plot([date_num]*len(x), x, density, label=row[0], alpha=0.7)
```







<seaborn.axisgrid.PairGrid at 0x13aab9810>



As we compare the the mean and skewness we see that there have been more negative values for the moments post 2020 which results in the different shape of density for mean and skewness than variance and kurtosis.

Q6

```
from scipy.stats import ks_2samp

# Perform the KS test
ks_stat, p_value = ks_2samp(log_returns_2020['^STOXX50E'], log_returns_p_2020['

print("KS statistic:", ks_stat)
print("P-value:", p_value)
```

```
KS statistic: 0.03046652950036495
P-value: 0.4578654774680238
```

The P value is less than 0.05 so we can accept the null hypothesis that the density of index has been same after 2020 as it was before.

Q7

```
last_buying_day_before_2022 = euro_stoxx_50.loc[euro_stoxx_50.index >= '2021-12
last_selling_day_in_2022 = euro_stoxx_50.loc[euro_stoxx_50.index <= '2022-12-31


start_price_2022 = euro_stoxx_50.loc[last_buying_day_before_2022]
end_price_2022 = euro_stoxx_50.loc[last_selling_day_in_2022]

non_nan_columns = start_price_2022.dropna().index.intersection(end_price_2022.d

filtered_end_price_2021 = start_price_2022[non_nan_columns]
filtered_end_price_2022 = end_price_2022[non_nan_columns]



yearly_returns_2022 = (filtered_end_price_2022 - filtered_end_price_2021) / fil


# Sorting the returns to find top ten and bottom ten
sorted_returns = yearly_returns_2022.sort_values()

# Top ten stocks with the highest returns
top_ten = sorted_returns.tail(10).sort_values(ascending=False)

# Bottom ten stocks with the lowest returns
bottom_ten = sorted_returns.head(10)
```

```
import pandas as pd

# Create a DataFrame to neatly display the results
top_bottom_ten = pd.DataFrame({
    'Top Ten Stocks': top_ten.index,
    'Top Ten Returns (%)': top_ten.values,
    'Bottom Ten Stocks': bottom_ten.index,
    'Bottom Ten Returns (%)': bottom_ten.values
})

print("Top and Bottom Ten Performing Stocks in 2022:")
print(top_bottom_ten)
```

```
Top and Bottom Ten Performing Stocks in 2022:
  Top Ten Stocks  Top Ten Returns (%) Bottom Ten Stocks  \
0         TTE.PA            40.660313            ADS.DE
1        MUV2.DE            22.275053          ADYEN.AS
2         DTE.DE            18.667602            DHL.DE
3         ENI.MI            16.197664            VOW.DE
4        BBVA.MC            15.109112            KER.PA
5         DB1.DE            11.902074            IFX.DE
6         SAF.PA             9.738576           ASML.AS
7         IBE.MC             9.720891            SGO.PA
8         ABI.BR             6.206159           ENEL.MI
9        BAYN.DE             6.195800             SU.PA

   Bottom Ten Returns (%)
0              -48.722766
1              -44.810451
2              -37.778564
3              -33.112481
4              -31.331948
5              -29.662826
6              -28.209010
7              -24.109509
8              -23.692374
9              -22.610783
```

```
import requests
from bs4 import BeautifulSoup

def fetch_esg_score(ticker):
    url = f"https://finance.yahoo.com/quote/{ticker}/sustainability/"
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWe
    try:
        response = requests.get(url, headers=headers)  # Adding headers to mimi
        if response.status_code == 200:
            soup = BeautifulSoup(response.text, 'html.parser')
            # Check if the class has changed or multiple elements are found
            esg_score = soup.find('h4', class_='svelte-y3c2sq')
            if esg_score:
                return float(esg_score.text.strip())
            else:
                print(f"No ESG score found for {ticker}")
        else:
            print(f"Failed to retrieve data for {ticker}, HTTP Status: {respons
    except Exception as e:
        print(f"Error fetching ESG data for {ticker}: {e}")
    return None


# Example tickers known to have ESG scores
test_tickers = ['TTE.PA', 'SU.PA', 'ABI.BR']

# Fetch ESG scores using your function
for ticker in test_tickers:
    esg_score = fetch_esg_score(ticker)
    print(f"Ticker: {ticker}, ESG Score: {esg_score}")
```

```
Ticker: TTE.PA, ESG Score: 28.3
Ticker: SU.PA, ESG Score: 11.3
Ticker: ABI.BR, ESG Score: 23.6
```

```
top_ten_tickers = top_ten.index.tolist()
bottom_ten_tickers = bottom_ten.index.tolist()

top_ten_esg_scores = {ticker: fetch_esg_score(ticker) for ticker in top_ten_tic
bottom_ten_esg_scores = {ticker: fetch_esg_score(ticker) for ticker in bottom_t

print(top_ten_esg_scores)
```

```
No ESG score found for ADYEN.AS
No ESG score found for VOW.DE
{'TTE.PA': 28.3, 'MUV2.DE': 15.1, 'DTE.DE': 18.7, 'ENI.MI': 29.0, 'BBVA.MC'
```

```
# Convert scores to float and prepare lists, filtering out None values
top_scores = [float(score) for score in top_ten_esg_scores.values() if score is
bottom_scores = [float(score) for score in bottom_ten_esg_scores.values() if sc

# Perform ANOVA test
anova_result = stats.f_oneway(top_scores, bottom_scores)
print('ANOVA test result:', anova_result)
```

```
ANOVA test result: F_onewayResult(statistic=8.878774667216028, pvalue=0.008
```

The Anova test provides p value of 0.008 we reject the null hypothesis that the ESG scores of top 10 are similar to that of the bottom 10.

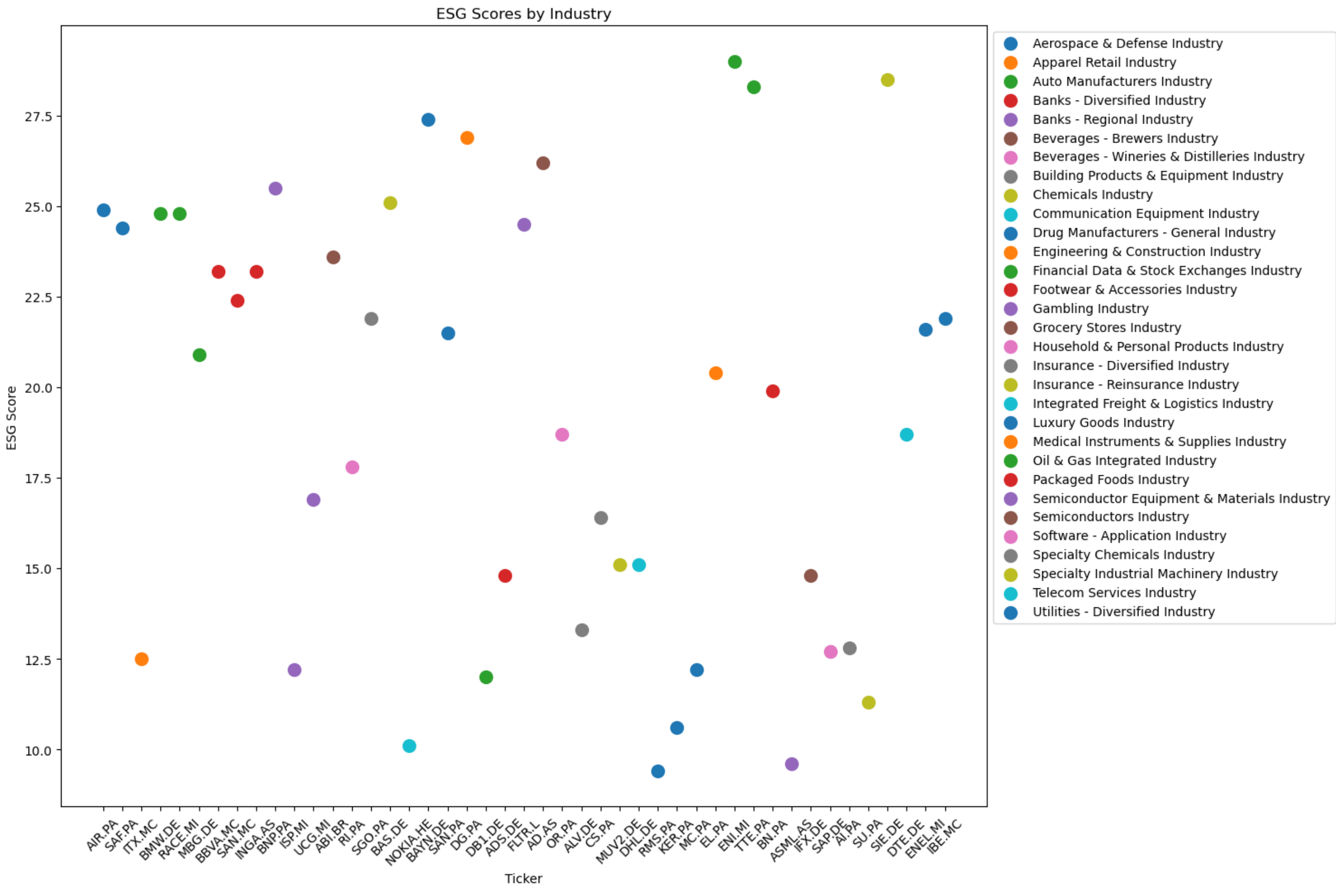
```
def get_stock_info(tickers):
    # Dictionary to hold the data
    data = {
        'Ticker': [],
        'Industry': [],
        'Sector': [],
        'ESG Score': []
    }

    for ticker in tickers:
        stock = yf.Ticker(ticker)
        info = stock.info  # This retrieves a dictionary of stock information

        # Extract industry and sector information
        data['Ticker'].append(ticker)
        data['Industry'].append(info.get('industry', 'N/A'))  # Get industry, r
        data['Sector'].append(info.get('sector', 'N/A'))  # Get sector, return
        data['ESG Score'].append(fetch_esg_score(ticker))

    # Convert the dictionary to a DataFrame
    df = pd.DataFrame(data)
    return df

industry_info = get_stock_info(eurostoxx50)
print(industry_info)
```

```
No ESG score found for ADYEN.AS
No ESG score found for NDA-FI.HE
No ESG score found for PRX.AS
No ESG score found for STLAM.MI
No ESG score found for VOW.DE
No ESG score found for ^STOXX50E
       Ticker                      Industry             Sector
0      ADS.DE        Footwear & Accessories   Consumer Cyclical
1    ADYEN.AS    Software - Infrastructure          Technology
2       AD.AS               Grocery Stores   Consumer Defensive
3       AI.PA           Specialty Chemicals     Basic Materials
```

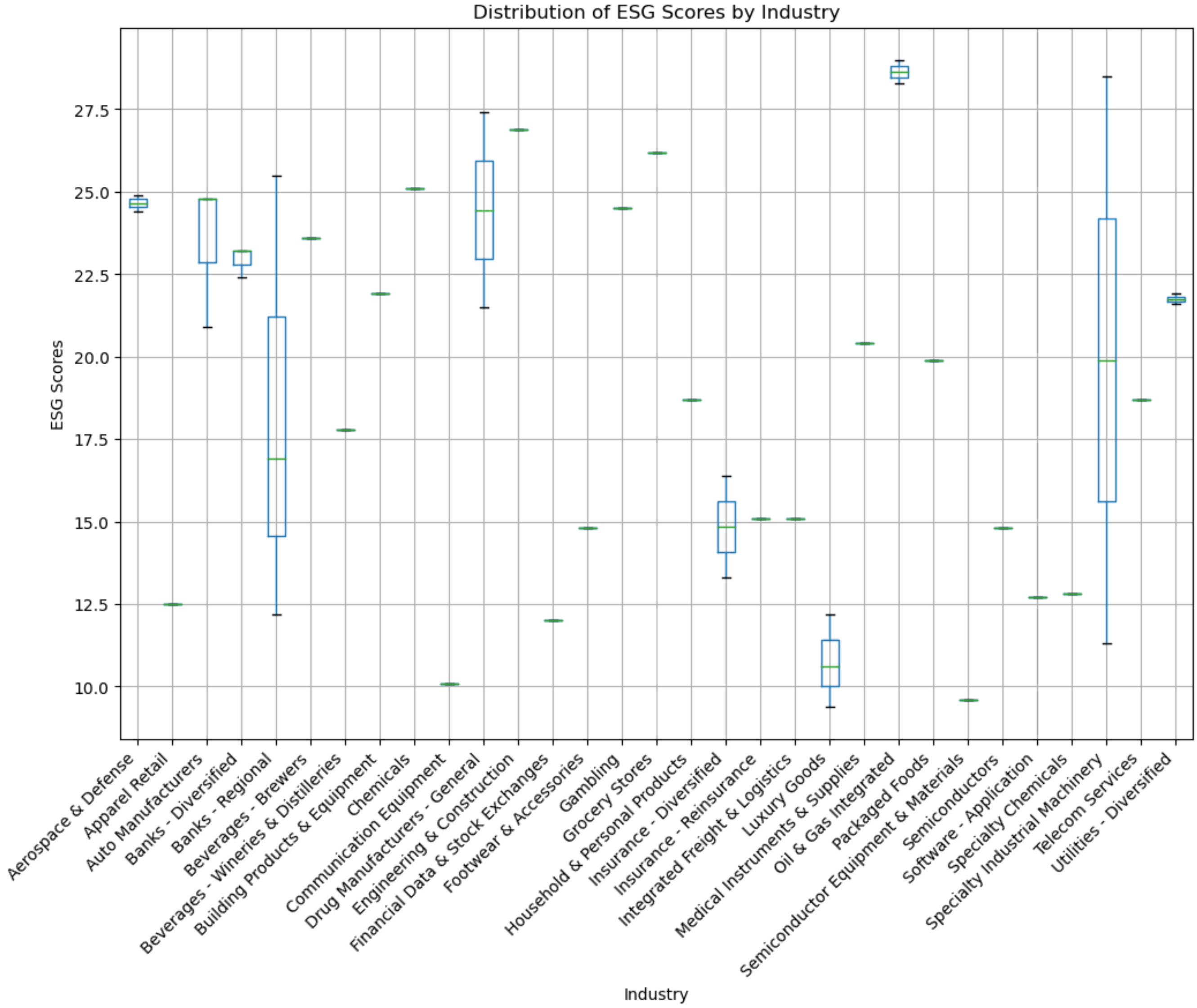
```
4      AIR.PA                  Aerospace & Defense             Industrials
5      ALV.DE              Insurance - Diversified      Financial Services
6      ABI.BR                  Beverages - Brewers      Consumer Defensive
7     ASML.AS  Semiconductor Equipment & Materials              Technology
8       CS.PA              Insurance - Diversified      Financial Services
9      BAS.DE                            Chemicals         Basic Materials
10    BAYN.DE         Drug Manufacturers - General              Healthcare
11    BBVA.MC                  Banks - Diversified      Financial Services
12     SAN.MC                  Banks - Diversified      Financial Services
13     BMW.DE                    Auto Manufacturers       Consumer Cyclical
14     BNP.PA                     Banks - Regional      Financial Services
15      BN.PA                       Packaged Foods      Consumer Defensive
16     DB1.DE     Financial Data & Stock Exchanges      Financial Services
17     DHL.DE       Integrated Freight & Logistics             Industrials
18     DTE.DE                     Telecom Services  Communication Services
19    ENEL.MI              Utilities - Diversified               Utilities
20     ENI.MI                 Oil & Gas Integrated                  Energy
21      EL.PA       Medical Instruments & Supplies              Healthcare
22    RACE.MI                   Auto Manufacturers       Consumer Cyclical
23     FLTR.L                             Gambling       Consumer Cyclical
24     RMS.PA                         Luxury Goods       Consumer Cyclical
25     IBE.MC              Utilities - Diversified               Utilities
26     ITX.MC                       Apparel Retail       Consumer Cyclical
27     IFX.DE                       Semiconductors              Technology
28    INGA.AS                  Banks - Diversified      Financial Services
29     ISP.MI                     Banks - Regional      Financial Services
30     KER.PA                         Luxury Goods       Consumer Cyclical
31      OR.PA        Household & Personal Products      Consumer Defensive
32      MC.PA                         Luxury Goods       Consumer Cyclical
33     MBG.DE                   Auto Manufacturers       Consumer Cyclical
34    MUV2.DE              Insurance - Reinsurance      Financial Services
35   NOKIA.HE              Communication Equipment              Technology
36  NDA-FI.HE                     Banks - Regional      Financial Services
37      RI.PA  Beverages - Wineries & Distilleries      Consumer Defensive
38     PRX.AS       Internet Content & Information  Communication Services
39     SAF.PA                  Aerospace & Defense             Industrials
40     SGO.PA        Building Products & Equipment             Industrials
41     SAN.PA         Drug Manufacturers - General              Healthcare
42     SAP.DE               Software - Application              Technology
43      SU.PA       Specialty Industrial Machinery             Industrials
44     SIE.DE       Specialty Industrial Machinery             Industrials
45   STLAM.MI                   Auto Manufacturers       Consumer Cyclical
46     TTE.PA                 Oil & Gas Integrated                  Energy
47      DG.PA          Engineering & Construction             Industrials
48     UCG.MI                     Banks - Regional      Financial Services
49     VOW.DE                   Auto Manufacturers       Consumer Cyclical
50  ^STOXX50E                                  N/A                     N/A
```

```
industry_groups = industry_info.groupby('Industry')

fig, ax = plt.subplots(figsize=(15, 10))  # Adjust the figure size

for name, group in industry_groups:
    filtered_group = group.dropna(subset=['ESG Score'])
```

```
    if not filtered_group.empty:  # Only plot if there's data to plot
        ax.scatter(filtered_group['Ticker'], filtered_group['ESG Score'], label

ax.set_xlabel('Ticker')
ax.set_ylabel('ESG Score')
ax.set_title('ESG Scores by Industry')
ax.legend(loc='upper left', bbox_to_anchor=(1, 1))  # Move the legend to the up
plt.xticks(rotation=45)  # Rotate x-axis labels

plt.tight_layout()
plt.show()
```

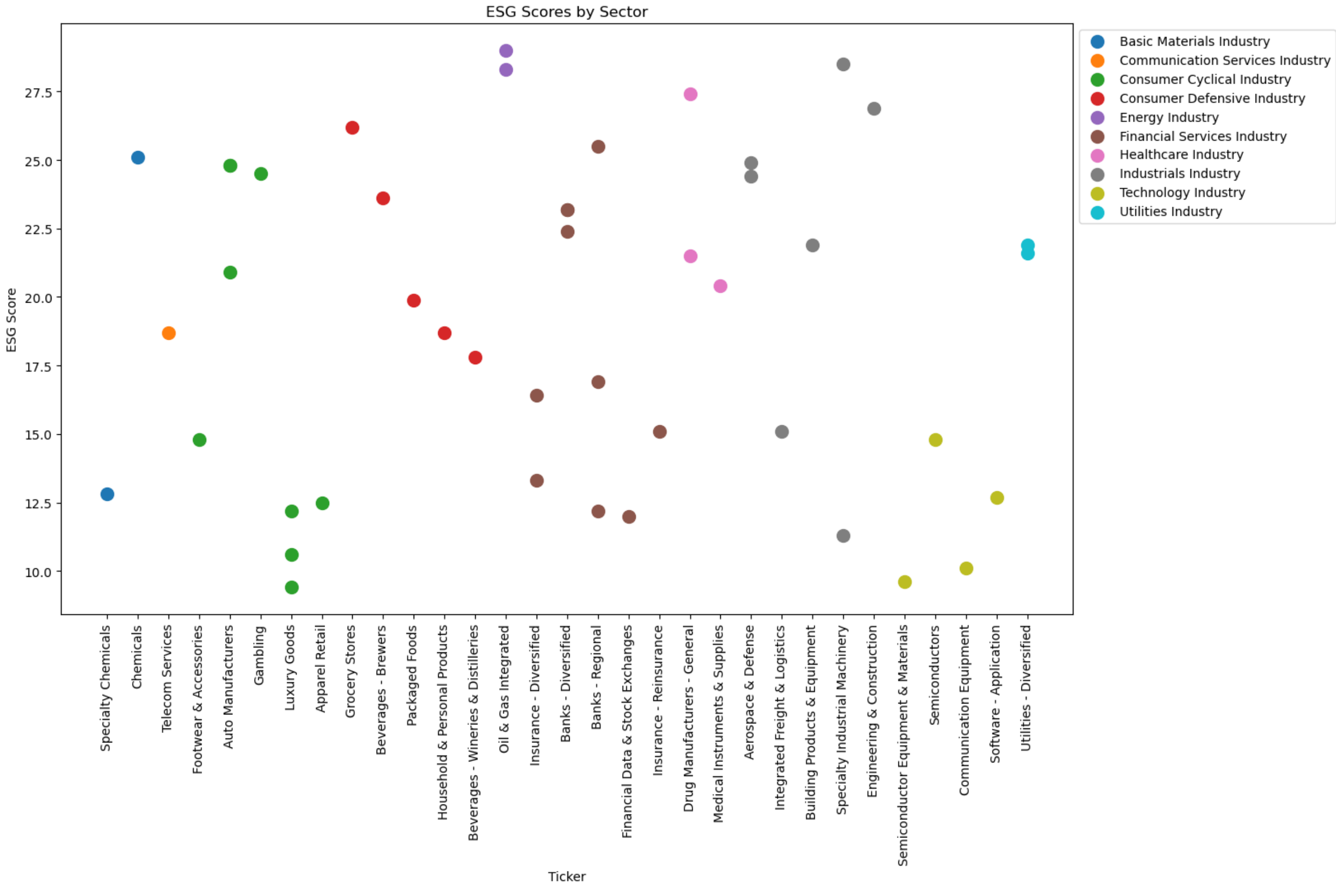


```
# Group data by Industry
industry_groups = industry_info.groupby('Industry')['ESG Score']

# Prepare a new DataFrame for plotting
data_to_plot = {name: group.values for name, group in industry_groups if not gr

# Create a DataFrame from the dictionary
plot_df = pd.DataFrame(dict([(k, pd.Series(v)) for k,v in data_to_plot.items()]

# Create box plot
plt.figure(figsize=(12, 8))
boxplot = plot_df.boxplot()
plt.xticks(rotation=45, ha='right')
plt.title('Distribution of ESG Scores by Industry')
plt.xlabel('Industry')
plt.ylabel('ESG Scores')
plt.show()
```

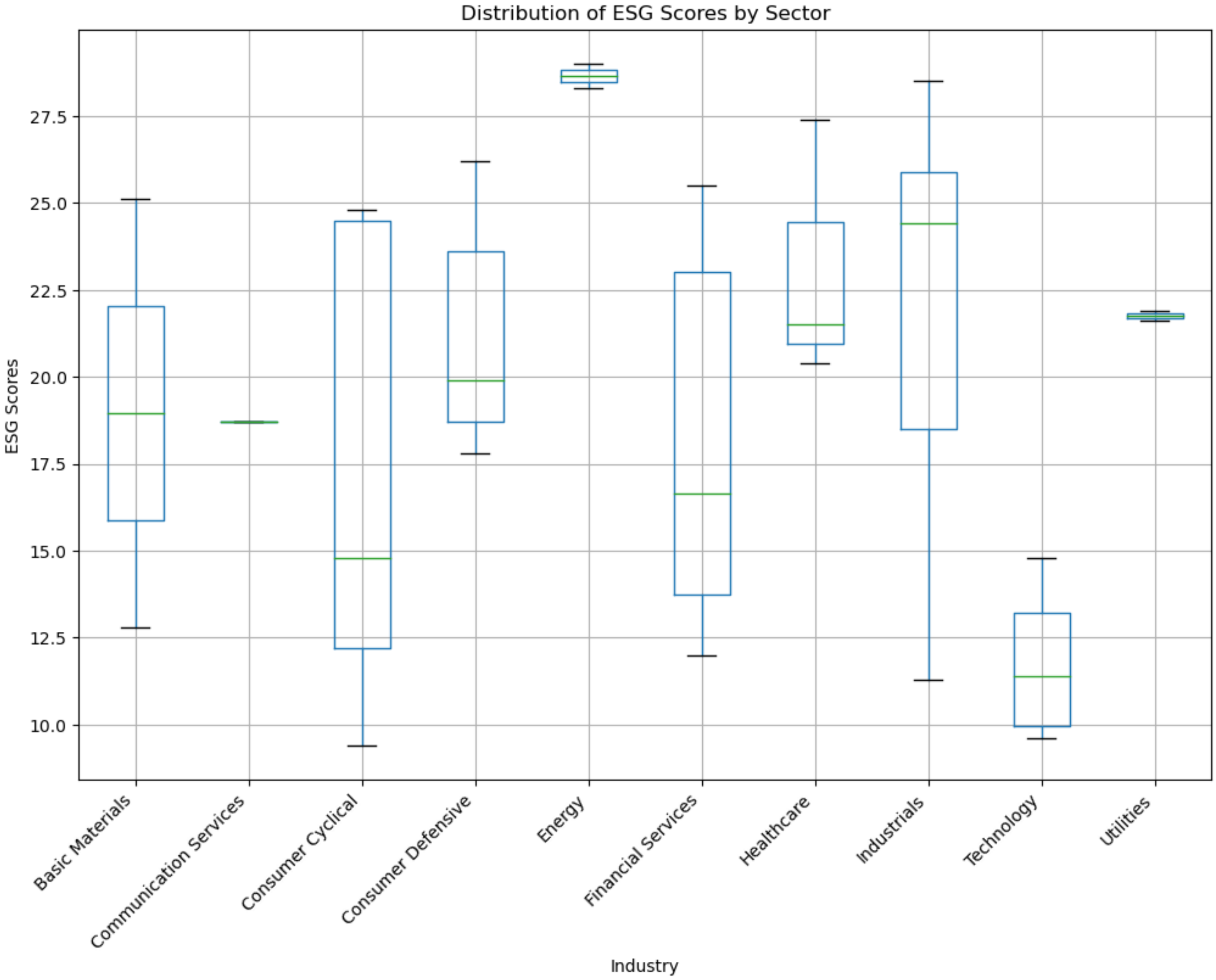



```
industry_groups = industry_info.groupby('Sector')
```

```
fig, ax = plt.subplots(figsize=(15, 10))  # Adjust the figure size

for name, group in industry_groups:
    filtered_group = group.dropna(subset=['ESG Score'])

    if not filtered_group.empty:  # Only plot if there's data to plot
        ax.scatter(filtered_group['Industry'], filtered_group['ESG Score'], lab

ax.set_xlabel('Ticker')
ax.set_ylabel('ESG Score')
ax.set_title('ESG Scores by Sector')
ax.legend(loc='upper left', bbox_to_anchor=(1, 1))  # Move the legend to the up
plt.xticks(rotation=90)  # Rotate x-axis labels

plt.tight_layout()
plt.show()
```



```
# Group data by Sector
sector_groups = industry_info.groupby('Sector')['ESG Score']

# Prepare a new DataFrame for plotting
data_to_plot = {name: group.values for name, group in sector_groups if not grou

# Create a DataFrame from the dictionary
plot_df = pd.DataFrame(dict([(k, pd.Series(v)) for k,v in data_to_plot.items()]

# Create box plot
plt.figure(figsize=(12, 8))
boxplot = plot_df.boxplot()
plt.xticks(rotation=45, ha='right')
plt.title('Distribution of ESG Scores by Sector')
plt.xlabel('Industry')
plt.ylabel('ESG Scores')
plt.show()
```




```
print(industry_info)
```

```
      Ticker                        Industry              Sector
0     ADS.DE        Footwear & Accessories   Consumer Cyclical
1   ADYEN.AS      Software - Infrastructure          Technology
2      AD.AS                  Grocery Stores  Consumer Defensive
```

3	AI.PA	Specialty Chemicals	Basic Materials
4	AIR.PA	Aerospace & Defense	Industrials
5	ALV.DE	Insurance - Diversified	Financial Services
6	ABI.BR	Beverages - Brewers	Consumer Defensive
7	ASML.AS	Semiconductor Equipment & Materials	Technology
8	CS.PA	Insurance - Diversified	Financial Services
9	BAS.DE	Chemicals	Basic Materials
10	BAYN.DE	Drug Manufacturers - General	Healthcare
11	BBVA.MC	Banks - Diversified	Financial Services
12	SAN.MC	Banks - Diversified	Financial Services
13	BMW.DE	Auto Manufacturers	Consumer Cyclical
14	BNP.PA	Banks - Regional	Financial Services
15	BN.PA	Packaged Foods	Consumer Defensive
16	DB1.DE	Financial Data & Stock Exchanges	Financial Services
17	DHL.DE	Integrated Freight & Logistics	Industrials
18	DTE.DE	Telecom Services	Communication Services
19	ENEL.MI	Utilities - Diversified	Utilities
20	ENI.MI	Oil & Gas Integrated	Energy
21	EL.PA	Medical Instruments & Supplies	Healthcare
22	RACE.MI	Auto Manufacturers	Consumer Cyclical
23	FLTR.L	Gambling	Consumer Cyclical
24	RMS.PA	Luxury Goods	Consumer Cyclical
25	IBE.MC	Utilities - Diversified	Utilities
26	ITX.MC	Apparel Retail	Consumer Cyclical
27	IFX.DE	Semiconductors	Technology
28	INGA.AS	Banks - Diversified	Financial Services
29	ISP.MI	Banks - Regional	Financial Services
30	KER.PA	Luxury Goods	Consumer Cyclical
31	OR.PA	Household & Personal Products	Consumer Defensive
32	MC.PA	Luxury Goods	Consumer Cyclical
33	MBG.DE	Auto Manufacturers	Consumer Cyclical
34	MUV2.DE	Insurance - Reinsurance	Financial Services
35	NOKIA.HE	Communication Equipment	Technology
36	NDA-FI.HE	Banks - Regional	Financial Services
37	RI.PA	Beverages - Wineries & Distilleries	Consumer Defensive
38	PRX.AS	Internet Content & Information	Communication Services
39	SAF.PA	Aerospace & Defense	Industrials
40	SGO.PA	Building Products & Equipment	Industrials
41	SAN.PA	Drug Manufacturers - General	Healthcare
42	SAP.DE	Software - Application	Technology
43	SU.PA	Specialty Industrial Machinery	Industrials
44	SIE.DE	Specialty Industrial Machinery	Industrials
45	STLAM.MI	Auto Manufacturers	Consumer Cyclical
46	TTE.PA	Oil & Gas Integrated	Energy
47	DG.PA	Engineering & Construction	Industrials
48	UCG.MI	Banks - Regional	Financial Services
49	VOW.DE	Auto Manufacturers	Consumer Cyclical
50	^STOXX50E	N/A	N/A

	ESG Score
0	14.8
1	NaN
2	26.2
3	12.8
4	24.9