Pellegrin Gabrielle n°88015
Butté Marine n°87701
Foumigué Rémi n°87814
Romea Kelvin n°88056

Financial Econometrics with Python

Question:

Does the time series you have downloaded following the preliminary steps present features in line with all the 8 Stylized Facts of financial prices / returns discussed in "Lecture 1: Financial Returns: Description and Stylized Facts''?

The **BEL20 Index (^BFX)** is a stock market index tracking the performance of the 20 largest companies listed on **Euronext Brussels, Belgium**. Similar to other major indices, the BEL20 operates as a total return index. This means it incorporates various financial elements, including dividends, interest, rights offerings, and other distributions, providing a comprehensive measure of performance. The index's base date is set at 18/03/1991, with an initial base value of 1000. The historical sample covers data from the first trading day on **04/10/1991** to **31/12/2022**. This dataset spans **7926 trading days**, equivalent to **1629 weeks**, **373 months**, or approximately **30 years** of financial data, reaching back to December 1991.

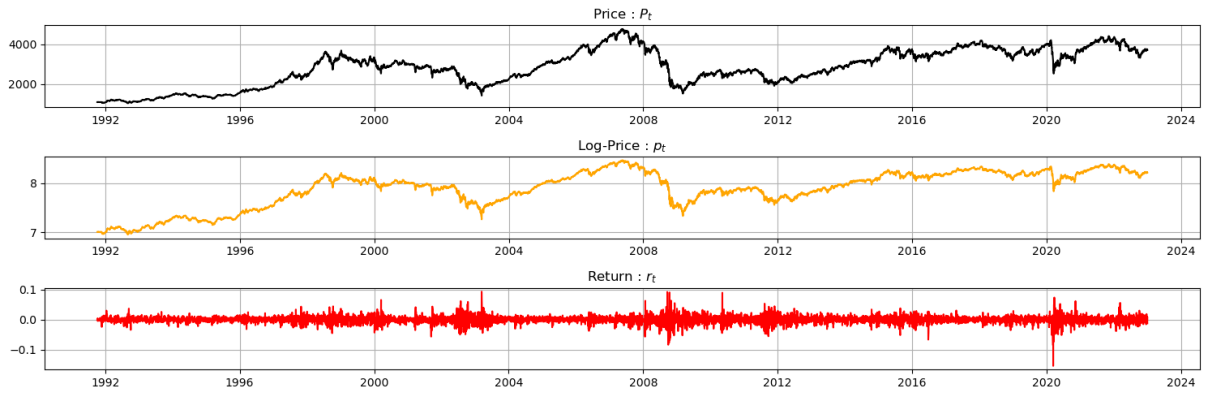

Figure 1: BEL20 Prices, log-prices and log-returns at daily frequency from "adjusted closing" prices taken from yahoo finance for sample: 04-10-1991 to 31-12-2022.

1-Statistic summary:

After retrieving the adjusted-closing sample data from Yahoo Finance and subsequently computing log returns for daily, weekly, monthly and annual frequencies, the following summary statistics table is generated.

Statistics	Daily	Weekly	Monthly	Yearly
Mean	0.0153	0.07395	0.33293	3.96334
St.Deviation	1.16704	2.65758	4.85192	22.54883
Diameter.C.I.Mean	0.02569	0.12906	0.4924	8.06899
Skewness	-0.38129	-1.38786	-1.03325	-1.51051
Kurtosis	9.89823	11.5855	3.40209	3.25951
Excess.Kurtosis	6.89823	8.5855	0.40209	0.25951
Min	-15.32755	-26.11094	-24.08791	-77.12737
Quant5	-1.86763	-4.31082	-7.33044	-27.27289
Quant25	-0.50972	-1.20804	-1.96919	-7.8009
Median	0.03825	0.3065	1.0609	10.90821
Quant75	0.60457	1.51384	3.50115	19.4
Quant95	1.67346	3.55273	6.44758	27.14699
Max	9.33398	12.90571	18.64553	37.3794
Jarque.Bera.stat	32548.3091	9633.39994	246.25255	24.68861
Jarque.Bera.pvalue.X100	0.0	0.0	0.0	0.00044
Lillie.test.stat	0.0819	0.0689	0.07827	0.16719
Lillie.test.pvalue.X100	0.1	0.1	0.1	3.21879
N.obs	7926.0	1629.0	373.0	30.0

Table 1: Summary statistics and distribution tests for BEL20 log-returns from "adjusted closing" prices from 04-10-1991 to 31-12-2022. Mean, st. deviation, min, max, quantiles are multiplied by 100, so their order of magnitude is that of percentage returns. P-values for the Jarque-Bera, Lilliefors, Kolmogorov-Smirnoff and Anderson-darling tests are also multiplied by 100, so to reject the null hypothesis H_0 at a confidence level of 5% they must be smaller than 5.

2- Stylized Facts analysis:

Stylized Fact 1: BEL20 Prices are non-stationary.

To assess the (weak) stationarity of prices, it seems useful to examine **Figure 1**. As we can see, both daily prices and log-prices (from adjusted closing prices) exhibit long run increasing trends of non-linear characteristics. Moreover, it seems nondeterministic as the index experiences prolonged downfalls during economic crisis such as 2008 or 2020, for example.

Another evidence of the non-stationarity of BEL20 prices is the plot of autocorrelation of prices (**Figure 2**), highlighting a slow decaying ACF (starting with large values near 1), hinting towards a long memory process.

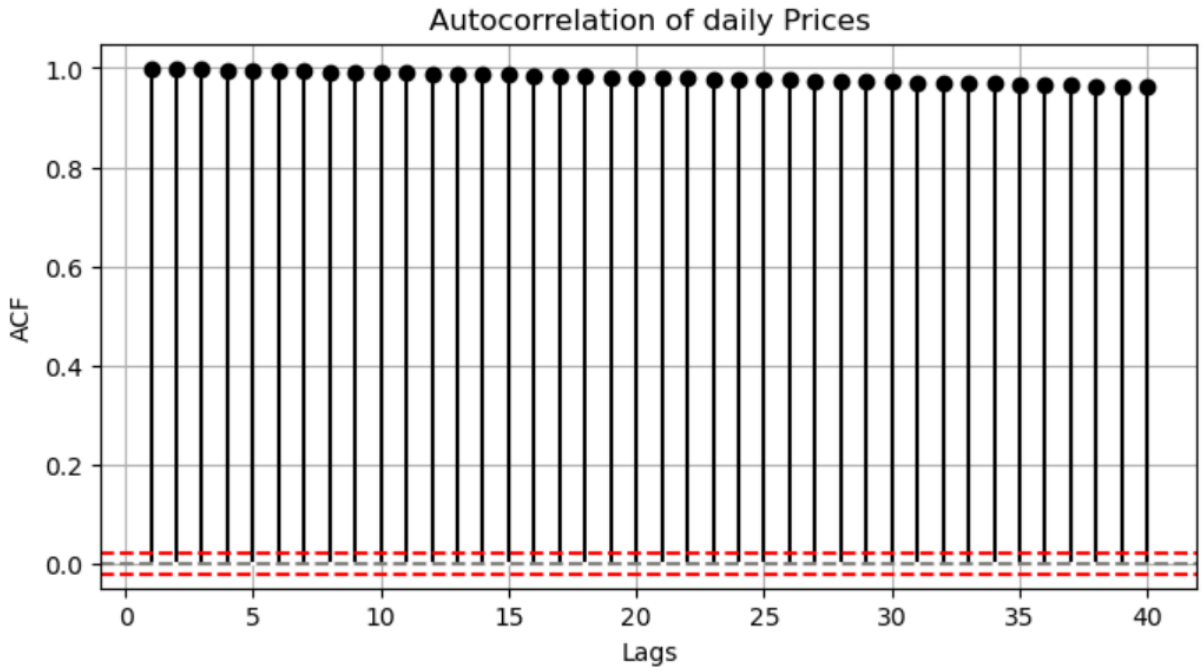

Figure 2: BEL20 daily prices autocorrelation for sample from 1991-10-04 to 2022-12-31.

Stylized fact 2: BEL20 returns are stationary.

Differently from prices, BEL20 returns typically fluctuate around a constant level, suggesting a constant mean over time, as we can see in the last part of **Figure 1**, with the daily log-returns from "adjusted closing" prices for sample from 1991-10-04 to 2022-12-31.

Stylized Fact 3: Returns are asymmetric.

If the returns were symmetric, we would expect the third central moment (Skewness) to be equal to zero. To estimate the skewness of the returns we must compute the skewness estimator which is given by this formula:

$$\hat{S} = \hat{S}(r_T) := \frac{\frac{1}{T}\sum_{t=1}^{T}(r_t - \bar{r})^3}{\left[\frac{1}{T}\sum_{t=1}^{T}(r_t - \bar{r})^2\right]^{3/2}}$$

While we assume that $\bar{r}$ is the sample mean.
A fundamental result in probability theory establishes that $\hat{S}$ is a consistently weak estimator of the theoretical moment. **Table 1** reveals that log-returns exhibit negative skewness across all frequencies. Notably, the skewness tends to increase as the sampling frequency decreases. This asymmetry is further evident when examining the histogram of returns and contrasting it with the normal distribution, as illustrated in **Figure 3**. The visual representation underscores the greater contribution of the fatter left tail in the distribution.

Stylized Fact 4: Returns have heavy tails.

To assess the tail heaviness of the log-returns distribution, the estimation of the 4th central moment (kurtosis) is necessary. If the estimated kurtosis of the data exceeds 3 (the kurtosis value for a normal distribution), it indicates that the data exhibits excess kurtosis or "fat tails". The empirical kurtosis is estimated using the sample estimator, calculated as:

$$\hat{K} = \hat{K}(r_T) := \frac{\frac{1}{T}\sum_{t=1}^{T}(r_t - \bar{r})^4}{\left[\frac{1}{T}\sum_{t=1}^{T}(r_t - \bar{r})^2\right]^2}$$

A standard result in probability theory indicates that $\hat{K}$ is a (weakly) consistent estimator of the theoretical 4th central moment. As observed in **Table 1**, log-returns exhibit excess kurtosis at all frequencies. Returns of BEL20 are leptokurtic (they have a positive excess kurtosis). Notably, the excess kurtosis tends to decrease with a decrease in sampling frequency. This observation aligns with established facts for stocks, emphasizing that stocks typically feature fat tails, and specifically, the left tail tends to be significantly fatter due to market crashes and pronounced market downturns.

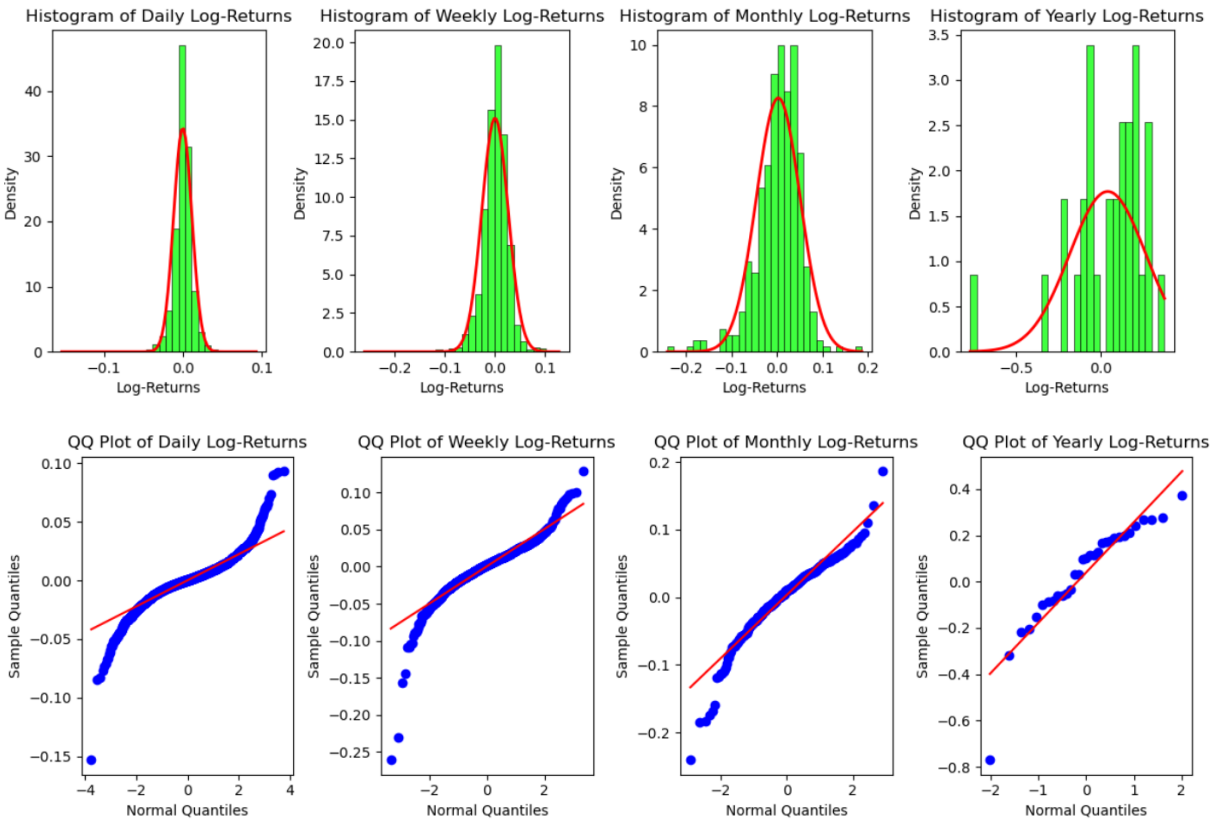

Figure 3: Log returns $r_t = p_t - p_{t-1}$ histograms of daily, weekly, monthly, and yearly "adjusted closing" of BEL20. Sample: 1991-10-04 to 2022-12-31. QQ plot against quantiles of normal distribution with same mean and variance as the empirical distribution of returns.

Visual confirmation of this phenomenon can be obtained through QQ-plot, **Figure 3**. The QQ-plot compares empirical quantiles with the quantiles of a normal distribution having the same mean and standard deviation as the empirical distribution of log-returns. In the QQ-plot, the distribution of returns displays fatter tails than a normal distribution, evident by points on the left side significantly below the y = x line, while those on the right side are above. These results hold for daily, weekly,

monthly, and annual returns, although there appears to be less deviation from the y = x line as the sampling frequency decreases, consistent with stylized fact number 5.

Stylized Fact 5: High Frequency non-Gaussianity

Empirical findings indicating the non-Gaussian nature of returns are evident through negative skewness, substantial kurtosis, and fat-tailed patterns in the empirical distribution. To assess the deviation from Gaussianity, Jack-Bera tests (for skewness and kurtosis) and goodness-of-fit tests such as Kolmogorov-Smirnov, Lilliefors, and Anderson-Darling were conducted.

The outcomes in **Table 1** reveal that the Jack-Bera test consistently highlights significant disparities in both kurtosis and skewness from normal distribution expectations across all frequencies. Notably, there is weaker support for the null hypothesis, particularly at the annual frequency.

Interestingly, **Figure 4** indicates that the quantiles of the distribution seem to align well with a student-t distribution with degrees of freedom equal to 3, 4, 5 or 6 at a daily level which confirms the non-Gaussianity.

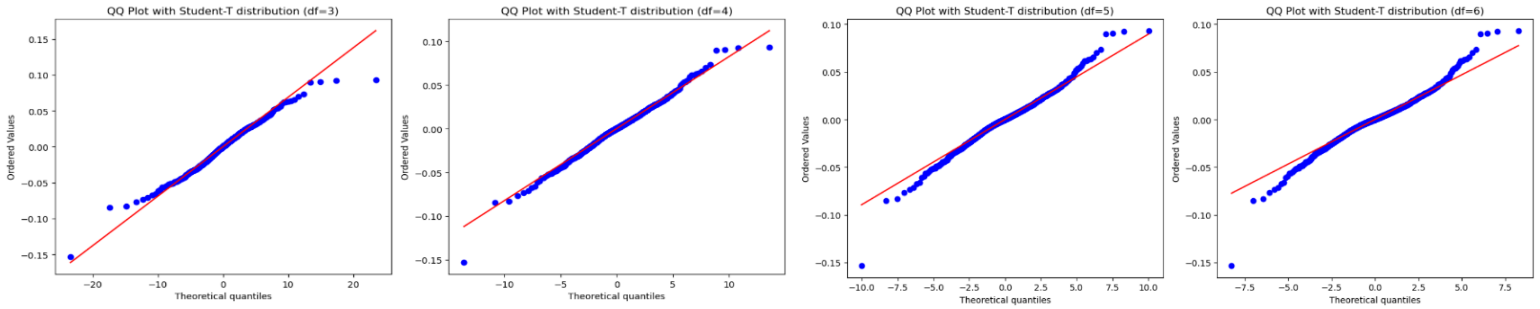

Figure 4: Log returns $rt := p_t - pt{-}1$: daily "adjusted closing" of BEL20. Sample: 04-10-1991 to 31-12-2022. QQ plot of Sample standardized quantiles (0 mean and unit variance) of daily log-returns against quantiles of standardized (0 mean and unit variance) Student-t distributions with ν= 3, 4, 5 and 6 degrees of freedom

Also, from **Table 1** and **Figure 5** Lilliefors test rejects normality assumption at 5% (|observed value|< 0,1542) for all frequency except at annual level.

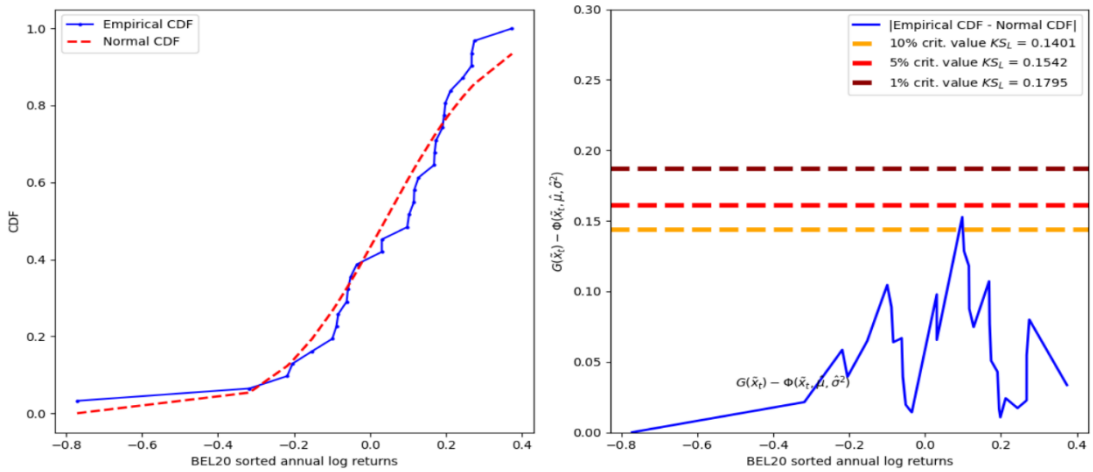

Figure 5: Log returns $rt := p_t - pt-1$: annual "adjusted closing" of BEL20. Sample: 04-10-1991 to 31-12-2022 (T=30). Left panel: empirical and Normal cdf's for the standardized annual returns of the BEL20. Right panel: values $|G_T(\tilde{r}t) - \Phi(\tilde{r}t,\hat{\mu}, \hat{\sigma}^2)|$ (blue line) and critical values for the Lilliefors test for the three significance levels 10%, 5% and 1%.

Stylized Fact 6: Returns are not autocorrelated.

To examine the autocorrelation of returns, three distinct methods were employed. Initially, individual asymptotic tests were conducted on the distribution of each lag-k correlogram ($\hat{\rho}_k$). The outcomes are visually presented in **Figure 6.**

On a daily and weekly basis, autocorrelation functions (ACFs) exhibit statistical significance but remain very small in absolute value. Conversely, for monthly returns, ACFs display very few statistical significances, suggesting little discernible departure from zero indicating a notable absence of autocorrelation in BEL20 returns. In summary, the findings collectively indicate limited or no evidence of autocorrelation among BEL20 returns.

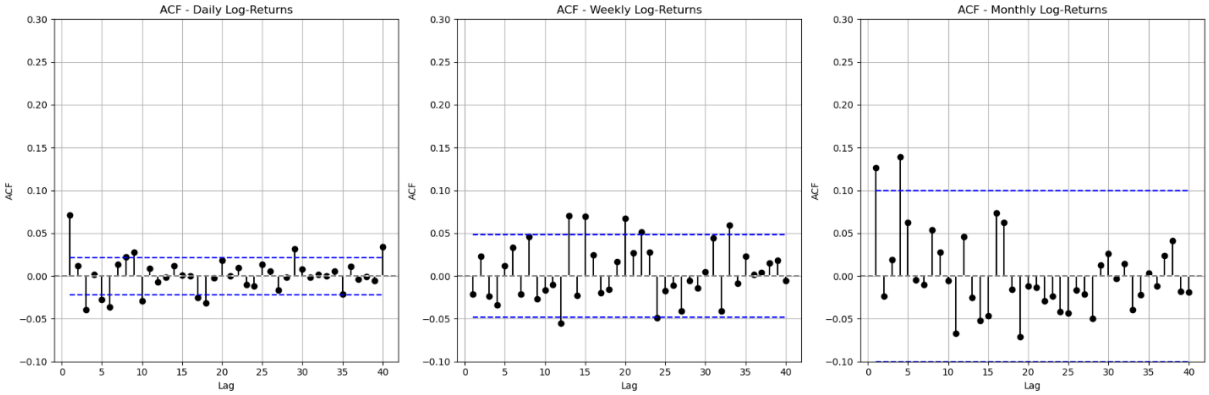

Figure 6: Empirical Autocorrelation (ACF) of *rt,* the daily, weekly and monthly log-return from the "adjusted closing" of BEL20. The autocorrelation of order 0 is not reported to have a better graphical representation of the smaller autocorrelations of order k. Sample: 04-10-1991 to 31-12-2022.

The final two methods are the standard Ljung-Box and Box-pierce tests of the distribution of the sum of the autocorrelations $\hat{\rho}_1+\hat{\rho}_2+\ldots+\hat{\rho}_p$. The two tests rely on the same hypothesis: $H_0 : \hat{\rho}_1 = \hat{\rho}_2 = \ldots = \hat{\rho}_p = 0$; $H_1 : \exists j : \rho_j \neq 0$ for $J = 1,\ldots,p$ and their test statistics are both distributed $\sim X^2_p$ (asymptotically) but their values are different as

$$Q_{BP} := T\sum_{j=1}^{p}\hat{\rho}_j^{\,2} \qquad Q_{LB} := T(T+2)\sum_{j=1}^{p}\frac{\hat{\rho}_j^{\,2}}{T+j}$$

As can be seen from **table 4** the daily returns do show weak signs of autocorrelation. While monthly data shows very weak signs of autocorrelation confirming at these frequencies the stylised fact.

lag	acf	acf diam.	acf test	**Box Pierce stat**	B-P pval	**L-B stat**	L-B pval	crit
1	0.13	0.101	2.513	6.314	0.012	6.364	0.012	3.841
2	-0.02	0.101	-0.395	6.469	0.039	6.522	0.038	5.991
3	0.02	0.101	0.381	6.614	0.085	6.669	0.083	7.815
4	0.145	0.101	2.813	14.529	0.006	14.712	0.005	9.488
5	0.063	0.101	1.214	16.002	0.007	16.213	0.006	11.07
6	-0.006	0.101	-0.123	16.018	0.014	16.229	0.013	12.592
7	-0.006	0.101	-0.111	16.03	0.025	16.241	0.023	14.067
8	0.055	0.101	1.056	17.144	0.029	17.386	0.026	15.507
9	0.022	0.101	0.424	17.324	0.044	17.571	0.04	16.919
10	-0.0	0.101	-0.008	17.324	0.068	17.571	0.063	18.307
11	-0.068	0.101	-1.313	19.048	0.06	19.357	0.055	19.675
12	0.041	0.101	0.801	19.689	0.073	20.023	0.067	21.026
13	-0.033	0.101	-0.635	20.092	0.093	20.443	0.085	22.362
14	-0.05	0.101	-0.97	21.034	0.101	21.426	0.091	23.685
15	-0.047	0.101	-0.913	21.867	0.111	22.298	0.1	24.996

Table 4: Empirical Autocorrelation (ACF), ACF "diameter" ($1.96\times\sqrt{\frac{1}{T}}$), Box-Pierce (BP) test and Ljung-Box test (LB): statistics and p-values. Data: r_t, the daily log-return from the "adjusted closing" of BEL20. Sample: 04-10-1991 to 31-12-2022.

lag	ACF	ACF diam.	ACF test	**Box-Pierce stat**	BP pval	**LB stat**	LB pval	crit
1	0.071	0.022	6.453	41.64	0.0	41.655	0.0	3.841
2	0.012	0.022	1.086	42.82	0.0	42.836	0.0	5.991
3	-0.04	0.022	-3.58	55.637	0.0	55.661	0.0	7.815
4	0.002	0.022	0.136	55.655	0.0	55.679	0.0	9.488
5	-0.027	0.022	-2.454	61.68	0.0	61.709	0.0	11.07
6	-0.037	0.022	-3.297	72.551	0.0	72.591	0.0	12.592
7	0.014	0.022	1.246	74.105	0.0	74.146	0.0	14.067
8	0.022	0.022	2.007	78.132	0.0	78.178	0.0	15.507
9	0.028	0.022	2.517	84.468	0.0	84.523	0.0	16.919
10	-0.029	0.022	-2.598	91.215	0.0	91.28	0.0	18.307
11	0.009	0.022	0.781	91.825	0.0	91.89	0.0	19.675
12	-0.007	0.022	-0.603	92.188	0.0	92.255	0.0	21.026
13	-0.001	0.022	-0.118	92.202	0.0	92.269	0.0	22.362
14	0.012	0.022	1.076	93.359	0.0	93.428	0.0	23.685
15	0.001	0.022	0.064	93.363	0.0	93.432	0.0	24.996

Table 5: Empirical Autocorrelation (ACF), ACF "diameter" ($1.96\times\sqrt{\frac{1}{T}}$), Box-Pierce (BP) test and Ljung-Box test (LB): statistics and p-values. Data: r_t, the monthly log-return from the "adjusted closing" of BEL20. Sample: 04-10-1991 to 31-12-2022.

Stylized fact 7: Returns feature volatility clustering.

The outcomes from autocorrelation tests on the absolute values of returns, as depicted in **Figure 7**, validate the stylized observation of volatility clustering. Volatility clustering denotes the occurrence of significant clusters of large price changes. This phenomenon implies substantial autocorrelation in squared returns (ARCH effect) or absolute returns. The evidence presented in **Figure 7** affirms this observation. Specifically, daily squared returns exhibit persistent correlation that decays slowly to zero as the lag (k) increases. This suggests long-memory properties in higher moments of returns. Additionally, it reinforces the corollary observation that volatility clustering diminishes at lower frequencies, as both monthly and annual data exhibit no pronounced signs of strong autocorrelation.

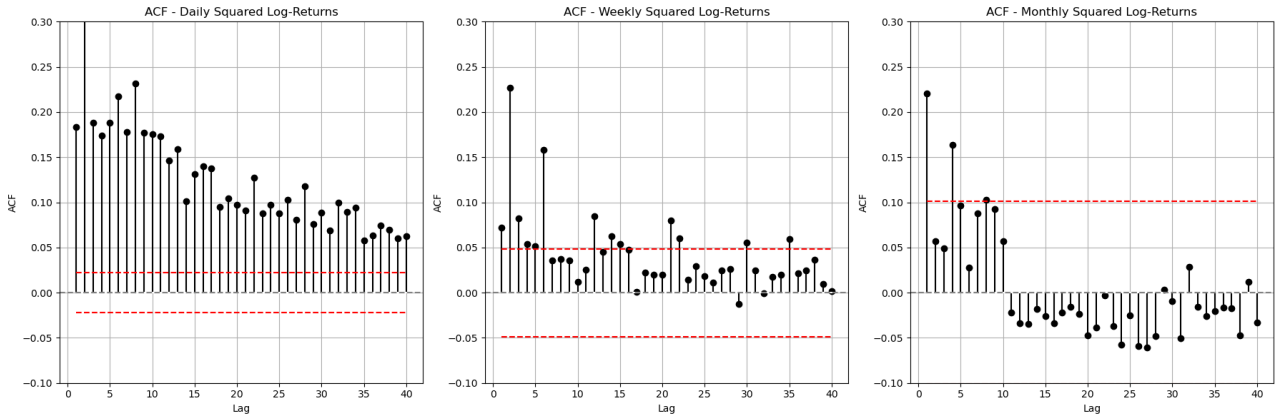

Figure 7: Empirical Autocorrelation of the squared value of daily, weekly and monthly log-return from the "adjusted closing" of BEL20. Sample: 04-10-1991 to 31-12-2022.

Stylized fact 8: BEL20 Returns show leverage effect.

The leverage effect is characterized by a negative autocorrelation between asset returns and the changes in their volatilities. Specifically, it anticipates an increase in asset-return volatility following price declines, and the magnitude of volatility spikes is expected to be positively correlated with the magnitude of the decline.

In essence, we anticipate a negative correlation, as indicated by $corr(r_{t-j}, r^2_t) < 0$ for $j > 0$. This expectation is affirmed by the results observed in the BEL20 analysis, as depicted in **Figure 8**. The left-hand side (LHS) of the figure illustrates that the cross-correlation with past values at a daily level is both negative and statistically significant.

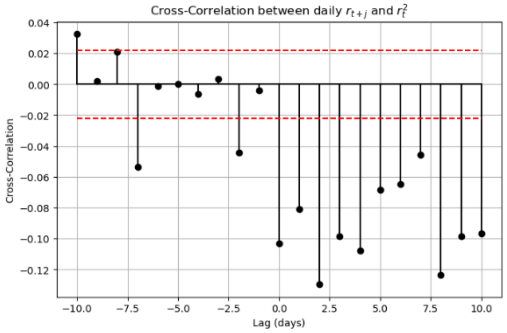

Figure 8: Empirical cross-correlation of daily lagged BEL20 log-returns, with squared returns: corr(r_{t-j} ; r^2_t) for j = −10;...;+10. Daily log-return from the "adjusted closing" of BEL20. Sample: 04-10-1991 to 31-12-2022. Ashed red lines are the (asymptotic) bounds for the rejection region a significance test of each cross-correlation. A line rising above or below the blue dashed lines represent a significant cross-correlation.

BONUS

Volatility prediction using GARCH and ARCH model.

According to Figure 1 and Stylized Fact 7 it has been established that return volatility is clustered. This suggests that the volatility at time 't' would depend on the volatility at times 't-1,' 't-2,' and so on. This is what we will attempt to demonstrate using ARCH and GARCH models.

An ARCH (autoregressive conditionally heteroscedastic) model is a model for the variance of a time series. ARCH models are used to describe a changing, possibly volatile variance. It is most often used in situations in which there may be short periods of increased variation. The formula for an ARCH(M) model (Autoregressive Conditional Heteroskedasticity with 'M' lags) is expressed as follows:

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{M} \alpha_i r_{t-i}^2$$

Assuming that σ^2_t is the conditional variance of the return at time 't.'
$\propto_0$ is a constant term representing the intercept of the model.
$\propto_i$ are parameters to be estimated, capturing the autoregressive relationship with the squared returns at lags t-i.
r^2_{t-i} represents the squared return at time t-i.
M is the number of lags considered in the model.

The GARCH (m, m) model (Generalized Autoregressive Conditional Heteroskedasticity with equal autoregressive and moving average orders 'm') is expressed as follows:

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{m} \alpha_i r_{t-i}^2 + \sum_{j=1}^{m} \beta_j \sigma_{t-j}^2$$

Assuming that β_j are parameters to be estimated, capturing the moving average relationship with the lagged conditional variances at lags t-j.
σ^2_{t-j} represents the lagged conditional variance at time t-j.
m is the number of lags considered in both the autoregressive and moving average components of the model.

The GARCH (m, m) model extends the ARCH model by incorporating lagged conditional variances, allowing for both autoregressive and moving average components in modelling the conditional variance of financial returns. Like the ARCH model, the parameters ($\propto_1, \propto_2, \ldots, \propto_n$, $\beta_1, \beta_2, \ldots, \beta_n$) are estimated from the data, typically using maximum likelihood estimation (MLE).

Estimation of the number of lags for the model

To estimate the number of lags that will fit our data from BEL20 index we can use the PACF function on squared returns. Each significant spike that we will observe after the first one could be considered as one more lags to take into account for our model.

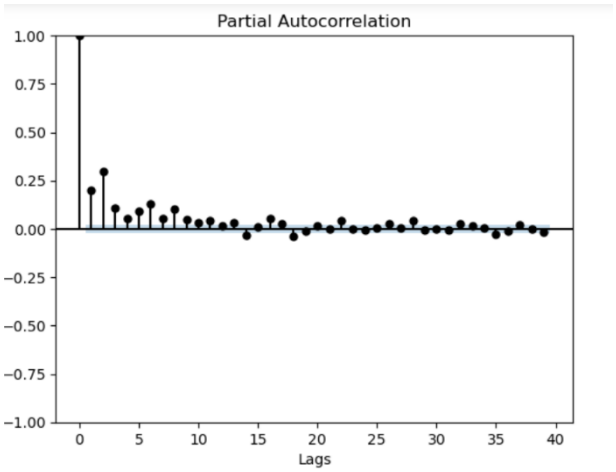

Figure 9: Empirical Partial Autocorrelation of the squared returns of daily returns from the "adjusted closing" of BEL20. Sample: 04-10-1991 to 31-12-2022.

As can be seen from **Figure 9** we observe that the two first lags are way above the critical values but then the partial autocorrelation values drop a lot after the first two lags which suggests that an appropriate model should be a GARCH (2,2) or an ARCH(2) model.

Testing the Adequacy of the Model in Capturing Observed Patterns

As can be seen from **Table 6** only alpha 1 is significant when we simulate a GARCH (2,2) model to predict the future variance which is odd since we would expect from **Figure 9** to have two parameters to explain the volatility. This might be due to the fact that we could be overfitting the model which mean that we are trying to explain our volatility with a model that is to complex for our data set.

This can be confirmed by the results of **Table 7** which reveals that both parameters are significant to explain the variance of our data. We would now expect to have a good prediction of future volatility using a ARCH(2) model.

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	0.0242	2.465e-02	0.983	0.326	[-2.409e-02,7.254e-02]
alpha[1]	0.1367	2.662e-02	5.133	2.848e-07	[8.448e-02, 0.189]
alpha[2]	0.0000	0.151	0.000	1.000	[-0.295, 0.295]
beta[1]	0.5698	1.124	0.507	0.612	[-1.632, 2.772]
beta[2]	0.2766	0.977	0.283	0.777	[-1.637, 2.191]

Table 6: Statistics summary of the fitness of a GARCH (2,2) model: statistics and p-values. Data: $r_{t,}$ the daily return from the "adjusted closing" of BEL20. Sample: 04-10-1991 to 31-12-2022.

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	0.5956	2.753e-02	21.637	8.070e-104	[0.542, 0.650]
alpha[1]	0.3062	3.029e-02	10.109	5.063e-24	[0.247, 0.366]
alpha[2]	0.2809	2.961e-02	9.486	2.409e-21	[0.223, 0.339]

Table 7: Statistics summary of the fitness of a ARCH (2) model: statistics and p-values. Data: r_t, the daily return from the "adjusted closing" of BEL20. Sample: 04-10-1991 to 31-12-2022.

Volatility Prediction

In order to try to predict the future volatility of our index we can try to do a rolling forecast of the volatility. A rolling forecast is a dynamic approach to predicting future volatility by continuously updating the forecast based on the latest available data.

From **figure 10** we can clearly see that the ARCH(2) model is a good model to predict future volatility because when the returns start to get more volatile the predicted volatility in orange have a similar reaction.

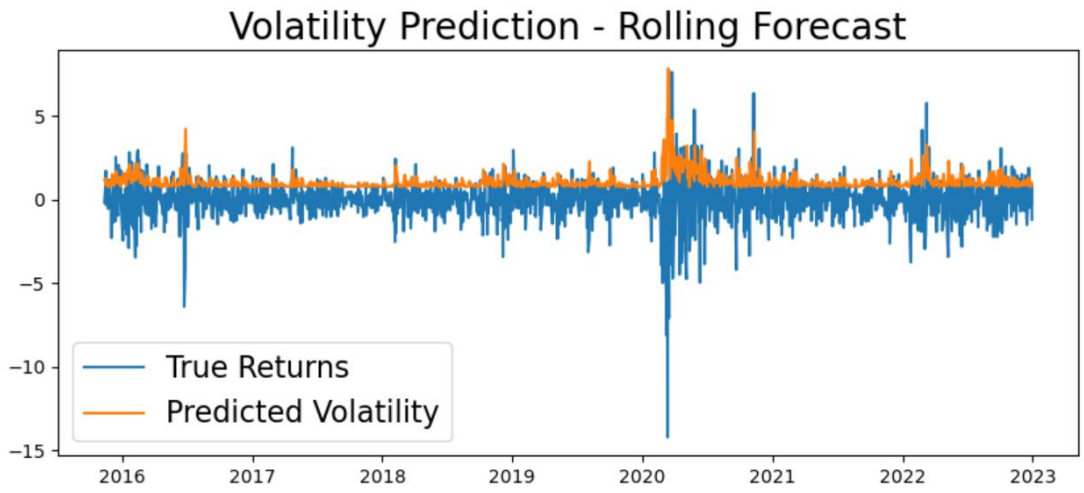

Figure 10: Volatility Prediction using rolling forecast over a period of 7 years, the blue line is the actual return of BEL20 for that period and the orange line is the predicted volatility for that period.

We have now a model that can predict the future volatility of the index which is very useful for taking decisions on whether we should invest or not in the index. Note that the accuracy of those volatilities' predictions declines with the increase of the time forecasting.

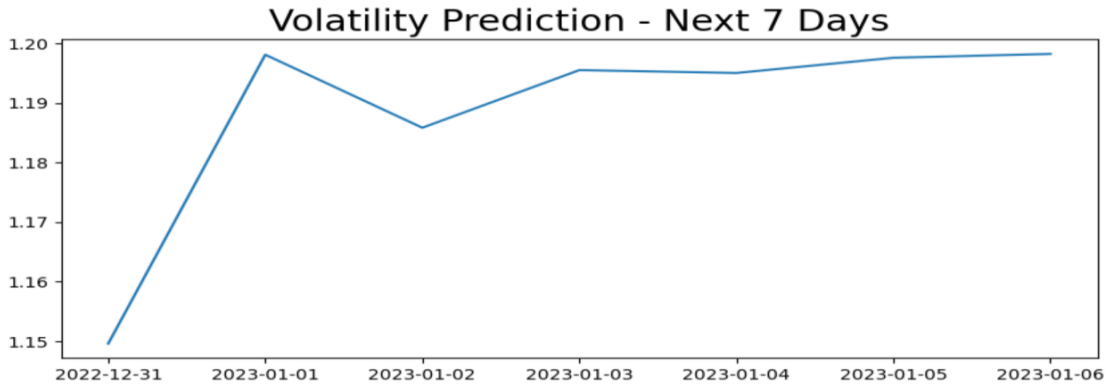

Figure 10: Volatility Prediction using our ARCH(2) model over a period of 7 days, the blue line is the predicted volatility of BEL20 returns

Appendix : Python code for Question 1 and Bonus.

```
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import acf
from scipy.stats import gaussian_kde, norm, iqr, skew, kurtosis, jarque_bera, kstest, anderson
from statsmodels.stats.diagnostic import lilliefors
import scipy.signal as ss
import pylab
from statsmodels.graphics.tsaplots import plot_acf # import this function from this submodule
import statsmodels.api as sm
import scipy.stats as stats



BEL20 = yf.download("^BFX",start="1991-10-04",end="2022-12-31")
Price_adjusted = BEL20["Adj Close"]
Price_adjusted = Price_adjusted.rename("Price")
Price_adjusted.index = pd.to_datetime(Price_adjusted.index)

#Log return computation
Logprice = np.log(Price_adjusted)
Logprice=Logprice.rename("Logprice")
#Logprice.head()

Logprice_W = Logprice.resample("W").last()
Logprice_M = Logprice.resample("M").last()
Logprice_Y = Logprice.resample("Y").last()

Logprice_W = Logprice_W.rename("Logprice_W")
Logprice_M = Logprice_M.rename("Logprice_M")
Logprice_Y = Logprice_Y.rename("Logprice_Y")

#Returns Log
```

```
LReturn_d = Logprice.diff()


#Remove NA


LReturn_D =Logprice.diff().dropna()
LReturn_W =Logprice_W.diff().dropna()
LReturn_M =Logprice_M.diff().dropna()
LReturn_Y =Logprice_Y.diff().dropna()


LReturn_D=LReturn_D.rename("LReturn_D")
LReturn_W=LReturn_W.rename("LReturn_W")
LReturn_M=LReturn_M.rename("LReturn_M")
LReturn_Y=LReturn_Y.rename("LReturn_Y")


#Interval considered
start_date = pd.to_datetime("1991-10-04")
end_date=pd.to_datetime("2023-11-10")
P_D = Price_adjusted.loc[start_date : end_date]
logp_D = Logprice.loc[start_date : end_date]
logp_W = Logprice_W.loc[start_date : end_date]
logp_M = Logprice_M.loc[start_date : end_date]
logp_Y = Logprice_Y.loc[start_date : end_date]
Logr_D = LReturn_D.loc[start_date : end_date]
Logr_W = LReturn_W.loc[start_date : end_date]
Logr_M = LReturn_M.loc[start_date : end_date]
Logr_Y = LReturn_Y.loc[start_date : end_date]


#first 3 graphs
fig, axs = plt.subplots(3, 1, figsize=(15, 5))
#Daily price
axs[0].plot(Price_adjusted.index,Price_adjusted, color = "black")
axs[0].set_title("Price : $P_t$")
axs[0].grid(True)
```

```
axs[1].plot(Logprice.index,Logprice, color = "orange")
axs[1].set_title("Log-Price : $p_t$")
axs[1].grid(True)

axs[2].plot(LReturn_D.index,LReturn_D, color = "red")
axs[2].set_title("Return : $r_t$")
axs[2].grid(True)
plt.tight_layout()

#auto correl

lags = 40
acf_values = acf(Logprice, nlags=lags)
Bartlett_Int = 1.96/np.sqrt(len(Logprice))
plt.figure(figsize=(8,4))
plt.stem(np.arange(1,lags+1), acf_values[1:], linefmt = "k-", markerfmt = "ko", basefmt = "w-")
plt.axhline(y=0, color="gray", linestyle = "--")
plt.axhline(y=Bartlett_Int, color="red", linestyle = "--")
plt.axhline(y=-Bartlett_Int, color="red", linestyle= "--")
plt.title("Autocorrelation of daily Prices")
plt.xlabel("Lags")
plt.ylabel("ACF")
plt.grid(True)

#Plting Stats summary
Dict = {
   "Daily" : LReturn_D,
   "Weekly" : LReturn_W,
   "Monthly" : LReturn_M,
   "Yearly" : LReturn_Y
}

def multi_fun(x):
   stat_tab = {
      'Mean': round(np.mean(x) * 100,5),
```

```
        'St.Deviation': round(np.std(x) * 100,5),
        'Diameter.C.I.Mean': round(1.96 * np.sqrt(np.var(x) / len(x)) * 100,5),
        'Skewness': round(skew(x),5),
        'Kurtosis': round(kurtosis(x),5),
        'Excess.Kurtosis': round(kurtosis(x) - 3,5),
        'Min': round(np.min(x) * 100,5),
        'Quant5': round(np.quantile(x, 0.05) * 100,5),
        'Quant25': round(np.quantile(x, 0.25) * 100,5),
        'Median': round(np.quantile(x, 0.50) * 100,5),
        'Quant75': round(np.quantile(x, 0.75) * 100,5),
        'Quant95': round(np.quantile(x, 0.95) * 100,5),
        'Max': round(np.max(x) * 100,5),
        'Jarque.Bera.stat': round(jarque_bera(x)[0],5),
        'Jarque.Bera.pvalue.X100': round(jarque_bera(x)[1] *100,5),
        'Lillie.test.stat': round(lilliefors(x)[0],5),
        'Lillie.test.pvalue.X100': round(lilliefors(x)[1] * 100,5),
        'N.obs': len(x)
    }
    return stat_tab

Statistic_Dict = {
    key: multi_fun(data.iloc[1:])
    for key, data in Dict.items()
}
Statistics_df = pd.DataFrame(Statistic_Dict)
print(Statistics_df)
Statistics_df.to_csv('statistics2.csv', index_label='Statistics')

# Set up the subplots
fig, axs = plt.subplots(2, 2, figsize=(12, 10))

# Probability Plot for Daily Returns
stats.probplot(LReturn_D, dist="norm", plot=axs[0, 0])
axs[0, 0].set_title("QQ Plot for Daily Returns")
axs[0, 0].grid(True)
```

```
# Probability Plot for Weekly Returns
stats.probplot(LReturn_W, dist="norm", plot=axs[0, 1])
axs[0, 1].set_title("QQ Plot for Weekly Returns")
axs[0, 1].grid(True)

# Probability Plot for Monthly Returns
stats.probplot(LReturn_M, dist="norm", plot=axs[1, 0])
axs[1, 0].set_title("QQ Plot for Monthly Returns")
axs[1, 0].grid(True)

# Probability Plot for Yearly Returns
stats.probplot(LReturn_Y, dist="norm", plot=axs[1, 1])
axs[1, 1].set_title("QQ Plot for Yearly Returns")
axs[1, 1].grid(True)

# Adjust layout and display the plot
plt.tight_layout()
plt.show()
fig, axs = plt.subplots(2, 2, figsize=(12, 10))
axs = axs.flatten()

df_values = [3, 4, 5, 6]

for i, df in enumerate(df_values):
    # QQ plot avec une distribution de Student-T
    stats.probplot(LReturn_D, dist="t", sparams=(df,), plot=axs[i])
    axs[i].set_title(f"QQ Plot with Student-T distribution (df={df})")

plt.tight_layout()
plt.show()
from arch import arch_model
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
returns = 100*BEL20.Close.pct_change().dropna()
plot_pacf(returns**2, color='black',vlines_kwargs={"colors": 'black'})
```

```
plt.xlabel('Lags', fontsize=10)
plt.show()
#model = arch_model(returns,p=2,q=2)
#model_fit = model.fit()
#model_fit.summary()
model2= arch_model(returns,p=2,q=0)
model2_fit = model2.fit()
model2_fit.summary()
model = arch_model(returns,p=2,q=0)
model_fit = model.fit()
model_fit.summary()
rolling_predictions = []
test_size = 365*5

for i in range(test_size):
    train = returns[:-(test_size-i)]
    model = arch_model(train, p=2, q=0)
    model_fit = model.fit(disp='off')
    pred = model_fit.forecast(horizon=1)
    rolling_predictions.append(np.sqrt(pred.variance.values[-1,:][0]))
rolling_predictions = pd.Series(rolling_predictions, index=returns.index[-365*5:])
plt.figure(figsize=(10,4))
true, = plt.plot(returns[-365*5:])
preds, = plt.plot(rolling_predictions)
plt.title('Volatility Prediction - Rolling Forecast', fontsize=20)
plt.legend(['True Returns', 'Predicted Volatility'], fontsize=16)
train = returns
model = arch_model(train, p=2, q=0)
model_fit = model.fit(disp='off')
pred = model_fit.forecast(horizon=7)
future_dates = [returns.index[-1] + timedelta(days=i) for i in range(1,8)]
pred = pd.Series(np.sqrt(pred.variance.values[-1,:]), index=future_dates)
plt.figure(figsize=(10,4))
plt.plot(pred)
plt.title('Volatility Prediction - Next 7 Days', fontsize=20)
```

```
# Compute the empirical ACF
lags = 40
acf_values_daily = acf(LReturn_D, nlags=lags)

# Compute the bartlet intervals
confint = 1.96 / np.sqrt(len(LReturn_D))
confint_upper = np.full(lags, confint)
confint_lower = -np.full(lags, confint)

# Set the layout
fig, axs = plt.subplots(1, 3, figsize=(18, 6))

# ACF of daily log-returns
axs[0].stem(np.arange(1, lags + 1), acf_values_daily[1:], linefmt='k-', markerfmt='ko', basefmt='w-')
axs[0].axhline(y=0, color='gray', linestyle='--')
axs[0].plot(np.arange(1, lags + 1), confint_upper, color='blue', linestyle='dashed')
axs[0].plot(np.arange(1, lags + 1), confint_lower, color='blue', linestyle='dashed')
axs[0].set_ylim(-0.1, 0.3)
axs[0].set_title('ACF - Daily Log-Returns')
axs[0].set_xlabel('Lag')
axs[0].set_ylabel('ACF')
axs[0].grid(True)

# ACF of weekly log-returns
acf_values_weekly = acf(LReturn_W, nlags=lags)
confint_weekly = 1.96 / np.sqrt(len(LReturn_W))
confint_weekly_upper = np.full(lags, confint_weekly)
confint_weekly_lower = -np.full(lags, confint_weekly)
axs[1].stem(np.arange(1, lags + 1), acf_values_weekly[1:], linefmt='k-', markerfmt='ko', basefmt='w-')
axs[1].axhline(y=0, color='gray', linestyle='--')
axs[1].plot(np.arange(1, lags + 1), confint_weekly_upper, color='blue', linestyle='dashed')
axs[1].plot(np.arange(1, lags + 1), confint_weekly_lower, color='blue', linestyle='dashed')
axs[1].set_ylim(-0.1, 0.3)
axs[1].set_title('ACF - Weekly Log-Returns')
axs[1].set_xlabel('Lag')
```

```
axs[1].set_ylabel('ACF')
axs[1].grid(True)

# ACF of monthly log-returns
acf_values_monthly = acf(LReturn_M, nlags=lags)
confint_monthly = 1.96 / np.sqrt(len(LReturn_M))
confint_monthly_upper = np.full(lags, confint_monthly)
confint_monthly_lower = -np.full(lags, confint_monthly)

axs[2].stem(np.arange(1, lags + 1), acf_values_monthly[1:], linefmt='k-', markerfmt='ko', basefmt='w-')
axs[2].axhline(y=0, color='gray', linestyle='--')
axs[2].plot(np.arange(1, lags + 1), confint_monthly_upper, color='blue', linestyle='dashed')
axs[2].plot(np.arange(1, lags + 1), confint_monthly_lower, color='blue', linestyle='dashed')
axs[2].set_ylim(-0.1, 0.3)
axs[2].set_title('ACF - Monthly Log-Returns')
axs[2].set_xlabel('Lag')
axs[2].set_ylabel('ACF')
axs[2].grid(True)


plt.tight_layout()

plt.show()
my_max_lag = 15
lags_all = np.arange(1, my_max_lag + 1)
my_acf = sm.tsa.acf(LReturn_D, nlags=my_max_lag)
my_acf_diameter = 1.96 / np.sqrt(len(LReturn_D))
my_acf_tstat_0 = (my_acf[1:] - 0) / np.sqrt(1 / len(LReturn_D))
my_LjungBox = sm.stats.diagnostic.acorr_ljungbox(LReturn_D, lags=lags_all, boxpierce=False)
my_BoxPierce = sm.stats.diagnostic.acorr_ljungbox(LReturn_D, lags=lags_all, boxpierce=True)
crit_value_5_BP = stats.chi2.ppf(0.95,lags_all)
my_table = np.column_stack((
    lags_all,
    my_acf[1:],
    np.full(my_max_lag, my_acf_diameter),
```

```
    my_acf_tstat_0,
    my_BoxPierce['bp_stat'],
    my_BoxPierce['bp_pvalue'],
    my_LjungBox['lb_stat'],
    my_LjungBox['lb_pvalue'],
    np.full(my_max_lag, crit_value_5_BP)
))
column_names = ["lag", "acf", "acf diam.", "acf test", "B-P stat", "B-P pval", "L-B stat", "L-B pval", "crit"]
my_table_df = pd.DataFrame(data=my_table, columns=column_names)

# Print the rounded table
my_table_df = my_table_df.round(3)
my_table_df



# Csv
#my_table_df.to_csv('ACFmy_table_df.csv')
my_max_lag = 15
lags_all = np.arange(1, my_max_lag + 1)
my_acf = sm.tsa.acf(LReturn_M, nlags=my_max_lag)
my_acf_diameter = 1.96 / np.sqrt(len(LReturn_M))
my_acf_tstat_0 = (my_acf[1:] - 0) / np.sqrt(1 / len(LReturn_M))
my_LjungBox = sm.stats.diagnostic.acorr_ljungbox(LReturn_M, lags=lags_all, boxpierce=False)
my_BoxPierce = sm.stats.diagnostic.acorr_ljungbox(LReturn_M, lags=lags_all, boxpierce=True)
crit_value_5_BP = stats.chi2.ppf(0.95,lags_all)

my_table = np.column_stack((
    lags_all,
    my_acf[1:],
    np.full(my_max_lag, my_acf_diameter),
    my_acf_tstat_0,
    my_BoxPierce['bp_stat'],
    my_BoxPierce['bp_pvalue'],
    my_LjungBox['lb_stat'],
    my_LjungBox['lb_pvalue'],
```

```
    np.full(my_max_lag, crit_value_5_BP)
))
column_names = ["lag", "acf", "acf diam.", "acf test", "B-P stat", "B-P pval", "L-B stat", "L-B pval", "crit"]
my_table_df = pd.DataFrame(data=my_table, columns=column_names)

# Print the rounded table
my_table_df = my_table_df.round(3)
my_table_df

my_table_df.to_csv('ACFmy_table_df21.csv')
# Compute the empirical ACF (squared)
lags = 40
acf_values_daily = acf(LReturn_D**2, nlags=lags)

# Compute the bartlet intervals
confint = 1.96 / np.sqrt(len(LReturn_D))
confint_upper = np.full(lags, confint)
confint_lower = -np.full(lags, confint)

# Set the layout
fig, axs = plt.subplots(1, 3, figsize=(18, 6))

# ACF of daily log-returns
axs[0].stem(np.arange(1, lags + 1), acf_values_daily[1:], linefmt='k-', markerfmt='ko', basefmt='w-')
axs[0].axhline(y=0, color='gray', linestyle='--')
axs[0].plot(np.arange(1, lags + 1), confint_upper, color='red', linestyle='dashed')
axs[0].plot(np.arange(1, lags + 1), confint_lower, color='red', linestyle='dashed')
axs[0].set_ylim(-0.1, 0.3)
axs[0].set_title('ACF - Daily Squared Log-Returns')
axs[0].set_xlabel('Lag')
axs[0].set_ylabel('ACF')
axs[0].grid(True)

# ACF of squared weekly log-returns
acf_values_weekly = acf(LReturn_W**2, nlags=lags)
```

```
confint_weekly = 1.96 / np.sqrt(len(LReturn_W))
confint_weekly_upper = np.full(lags, confint_weekly)
confint_weekly_lower = -np.full(lags, confint_weekly)

axs[1].stem(np.arange(1, lags + 1), acf_values_weekly[1:], linefmt='k-', markerfmt='ko', basefmt='w-')
axs[1].axhline(y=0, color='gray', linestyle='--')
axs[1].plot(np.arange(1, lags + 1), confint_weekly_upper, color='red', linestyle='dashed')
axs[1].plot(np.arange(1, lags + 1), confint_weekly_lower, color='red', linestyle='dashed')
axs[1].set_ylim(-0.1, 0.3)
axs[1].set_title('ACF - Weekly Squared Log-Returns')
axs[1].set_xlabel('Lag')
axs[1].set_ylabel('ACF')
axs[1].grid(True)

# ACF of squared monthly log-returns
acf_values_monthly = acf(LReturn_M**2, nlags=lags)
confint_monthly = 1.96 / np.sqrt(len(LReturn_M))
confint_monthly_upper = np.full(lags, confint_monthly)
confint_monthly_lower = -np.full(lags, confint_monthly)

axs[2].stem(np.arange(1, lags + 1), acf_values_monthly[1:], linefmt='k-', markerfmt='ko', basefmt='w-')
axs[2].axhline(y=0, color='gray', linestyle='--')
axs[2].plot(np.arange(1, lags + 1), confint_monthly_upper, color='red', linestyle='dashed')
axs[2].plot(np.arange(1, lags + 1), confint_monthly_lower, color='red', linestyle='dashed')
axs[2].set_ylim(-0.1, 0.3)
axs[2].set_title('ACF - Monthly Squared Log-Returns')
axs[2].set_xlabel('Lag')
axs[2].set_ylabel('ACF')
axs[2].grid(True)

# Set the space
plt.tight_layout()

plt.show()
LReturn_D2=LReturn_D**2
```

```
def ccf(x, y, lag_max = 100):
    # compute correlation
    result = ss.correlate(y - np.mean(y), x - np.mean(x), method='direct') / (np.std(y) * np.std(x) * len(y))
    # define the length
    length = (len(result) - 1) // 2
    lo = length - lag_max
    hi = length + (lag_max + 1)

    return result[lo:hi]
lag_max = 10
cross_corr = ccf(LReturn_D,LReturn_D2,lag_max=lag_max)
lags = np.arange(-lag_max, lag_max + 1)

# ACF dei log-returns mensili con bande di confidenza
confint_daily = 1.96 / np.sqrt(len(LReturn_D))
confint_daily_upper = np.full(len(lags), confint_daily)
confint_daily_lower = -np.full(len(lags), confint_daily)

plt.figure(figsize=(8, 5))
plt.stem(lags, cross_corr, linefmt='black',markerfmt='black', basefmt='k-')
plt.plot(lags, confint_daily_upper, color='red', linestyle='dashed')
plt.plot(lags, confint_daily_lower, color='red', linestyle='dashed')
plt.xlabel('Lag (days)')
plt.ylabel('Cross-Correlation')
plt.title('Cross-Correlation between daily $r_{t+j}$ and $r_t^2$')
plt.grid(True)

plt.show()

# Sort the data
my_data_ordered = np.sort(LReturn_Y)

# Calculate mean and standard deviation of the data
mean_data = np.mean(LReturn_Y)
sd_data = np.std(LReturn_Y)
```

```
# Sample size
samp_size = len(LReturn_Y)

# Empirical CDF
seq_ind = np.arange(1, samp_size + 1, 1)
emp_cdf = seq_ind / samp_size
emp_cdf_2 = (seq_ind - 1) / samp_size

# Theoretical CDF (Normal distribution)
theor_cdf = norm.cdf(my_data_ordered, mean_data, sd_data)

# Plotting
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(my_data_ordered, emp_cdf, 'bo-', label='Empirical CDF', markersize=2)
plt.plot(my_data_ordered, theor_cdf, 'r--', label='Normal CDF', linewidth=2)
plt.xlabel('BEL20 sorted annual log returns')
plt.ylabel('CDF')
plt.legend()

# Kolmogorov-Smirnov test statistics
KS_L_stat1 = np.max(np.abs(emp_cdf - theor_cdf))
KS_L_stat2 = np.max(np.abs(emp_cdf_2 - theor_cdf))
KS_L_stat = max(KS_L_stat1, KS_L_stat2)

plt.subplot(1, 2, 2)
plt.plot(my_data_ordered, np.abs(emp_cdf_2 - theor_cdf), 'b-', label='|Empirical CDF - Normal CDF|', linewidth=2)
plt.axhline(0.144 , color='orange', linestyle='--', linewidth=4, label='10% crit. value $KS_L$ = 0.1401')
plt.axhline(0.161 , color='red', linestyle='--', linewidth=4, label='5% crit. value $KS_L$ = 0.1542')
plt.axhline(0.187, color='darkred', linestyle='--', linewidth=4, label='1% crit. value $KS_L$ = 0.1795')
plt.text(-0.5, 0.032, '$ G(\\tilde{x}_t)-\\Phi(\\tilde{x}_t, \\hat{\\mu}, \\hat{\\sigma}^2)$', fontsize=10)
plt.xlabel('BEL20 sorted annual log returns')
plt.ylabel('$ G(\\tilde{x}_t)-\\Phi(\\tilde{x}_t, \\hat{\\mu}, \\hat{\\sigma}^2)$')
```

```
plt.ylim(0, 0.3)
plt.legend()

# Critical values of Lilliefors test for T = length dataset
crit_values = np.array([0.805, 0.886, 1.031]) / np.sqrt(samp_size)

plt.tight_layout()
plt.show()

# Print KS_L statistics
print(KS_L_stat1, KS_L_stat2, KS_L_stat)

# Print critical values
print(crit_values)
import yfinance as yf
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats
import seaborn as sns

# Download BEL20 data for the specified period
bel20 = yf.download("^BFX", start="1951-01-01", end="2022-12-31")

# Extract daily log-returns
log_price_daily = np.log(bel20['Adj Close'])
log_returns_daily = log_price_daily.diff().dropna()

log_price_weekly = np.log(bel20['Adj Close']).resample('W').last()
log_returns_weekly = log_price_weekly.diff().dropna()

log_price_yearly = np.log(bel20['Adj Close']).resample('Y').last()
log_returns_yearly = log_price_yearly.diff().dropna()



# Calculate monthly log-returns
```

```
log_price_monthly = np.log(bel20['Adj Close']).resample('M').last()

log_returns_monthly = log_price_monthly.diff().dropna()


# Create the figure with four subplots

fig, axs = plt.subplots(2, 2, figsize=(12, 8))


# Plot histogram of daily log-returns

sns.histplot(log_returns_daily, bins=30, color='lime', edgecolor='black', kde_kws={'color': 'red'}, ax=axs[0, 0], stat='density')

axs[0, 0].plot(np.linspace(log_returns_daily.min(), log_returns_daily.max(), 100),

        stats.norm.pdf(np.linspace(log_returns_daily.min(), log_returns_daily.max(), 100),

                log_returns_daily.mean(), log_returns_daily.std()), color='red', linewidth=2)

axs[0, 0].set_title('Histogram of Daily Log-Returns')

axs[0, 0].set_xlabel('Log-Returns')

axs[0, 0].set_ylabel('Density')


# Plot histogram of weekly log-returns

sns.histplot(log_returns_weekly, bins=30, color='lime', edgecolor='black', kde_kws={'color': 'red'}, ax=axs[0, 1],
stat='density')

axs[0, 1].plot(np.linspace(log_returns_weekly.min(), log_returns_weekly.max(), 100),

        stats.norm.pdf(np.linspace(log_returns_weekly.min(), log_returns_weekly.max(), 100),

                log_returns_weekly.mean(), log_returns_weekly.std()), color='red', linewidth=2)

axs[0, 1].set_title('Histogram of Weekly Log-Returns')

axs[0, 1].set_xlabel('Log-Returns')

axs[0, 1].set_ylabel('Density')


# Plot histogram of monthly log-returns

sns.histplot(log_returns_monthly, bins=30, color='lime', edgecolor='black', kde_kws={'color': 'red'}, ax=axs[1, 0],
stat='density')

axs[1, 0].plot(np.linspace(log_returns_monthly.min(), log_returns_monthly.max(), 100),

        stats.norm.pdf(np.linspace(log_returns_monthly.min(), log_returns_monthly.max(), 100),

                log_returns_monthly.mean(), log_returns_monthly.std()), color='red', linewidth=2)

axs[1, 0].set_title('Histogram of Monthly Log-Returns')

axs[1, 0].set_xlabel('Log-Returns')

axs[1, 0].set_ylabel('Density')


# Plot histogram of yearly log-returns
```

```
sns.histplot(log_returns_yearly, bins=30, color='lime', edgecolor='black', kde_kws={'color': 'red'}, ax=axs[1, 1], stat='density')
axs[1, 1].plot(np.linspace(log_returns_yearly.min(), log_returns_yearly.max(), 100),
        stats.norm.pdf(np.linspace(log_returns_yearly.min(), log_returns_yearly.max(), 100),
                log_returns_yearly.mean(), log_returns_yearly.std()), color='red', linewidth=2)
axs[1, 1].set_title('Histogram of Daily Log-Returns')
axs[1, 1].set_xlabel('Log-Returns')
axs[1, 1].set_ylabel('Density')
```