Empirical Methods in Finance – Assignment 3

I. Part 1: GARCH Models

In this part, we will focus on the use of the well-known Generalized Autoregressive Conditional Heteroskedasticity model (or GARCH model). It is an extension of the ARCH model elaborated by Robert Engle in 1982 where the variance of innovations (or errors terms of a time series or unexpected returns) ε_t is a function of the previous innovations constructed as an Autoregressive model (AR). We can model assets returns with a drift part μ_t and the error term ε_t :

$$r_t = \mu_t + \varepsilon_t \quad ; \quad \varepsilon_t = \sigma_t z_t \quad ; \quad z_t \sim iid\,(0,1) \quad ; \quad \mu_t = E[r_t \parallel \mathcal{I}_{t-1}] \quad ; \quad \sigma_t^2 = E[(r_t - \mu_t)^2 \parallel \mathcal{I}_{t-1}]$$

$$ARMA(p,q) \Rightarrow \mu_t = \mu + \sum_{i=1}^{p} \phi_i r_{t-i} - \sum_{j=1}^{q} \theta_j \varepsilon_{t-j}$$

Unexpected returns are serially uncorrelated but dependent. Hence, the ARCH model can be represented below with the conditional volatility of r_t as σ_t^2 known at $t-1$, more relevant as a risk measure than the unconditional volatility σ^2 :

$$\sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 + \cdots + \alpha_p \varepsilon_{t-p}^2 = \omega + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2 \quad ; \quad \varepsilon_t^2 = \sigma_t^2 + \sigma_t^2(z_t^2 - 1) = \sigma_t^2 + \nu_t = E_{t-1}[\varepsilon_t^2] + \nu_t$$

$$\sigma^2 = E[\varepsilon_t^2] = E\big[E_{t-1}[\varepsilon_t^2]\big] = E[\sigma_t^2] = \frac{\omega}{(1 - \sum_{i=1}^{p} \alpha_i)}$$

Here, covariance-stationarity is satisfied if we have $\sum_{i=1}^{p} \alpha_i < 1$ and $\omega < +\infty$. Besides, ε_t would be stationary too in this case. If an ARMA model is used to model the errors terms, the GARCH model will be used instead. Persistence through volatility is missing in the ARCH model, so the GARCH model is used to capture this property. Also, ARCH models are more likely to over-predict volatility because they respond slowly to large, isolated shocks to the return series. The original model is model by σ_t^2 below with $\omega > 0$ and $\alpha_i, \beta_j \geq 0$ if we want a positive variance:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 \quad ; \quad \sigma^2 = \frac{\omega}{1 - \sum_{i=1}^{p} \alpha_1 - \sum_{j=1}^{q} \beta_j}$$

We obtain then a covariance-stationary process if the following conditions for the estimated coefficients are satisfied:

$$\sum_{i=1}^{p} \alpha_1 + \sum_{j=1}^{q} \beta_j < 1 \quad ; \quad \omega < +\infty$$

These volatilities models are widely used in quantitative finance. It is a very good way to model financial time series such as returns with time-varying volatility but also volatility clustering where large changes tend to be followed by large changes (inversely for small changes that tend to be followed by small changes). The GARCH model has been extended many times with for instance the EGARCH model. It is the exponential version of the original model:

$$\log \sigma_t^2 = \omega + (\alpha_1[|z_{t-1}| - \mu_z] + \gamma_1 z_{t-1}) + \beta_1 \log \sigma_{t-1}^2 \quad ; \quad \mu_z = E[|z_{t-1}|] = \sqrt{2/\pi}$$

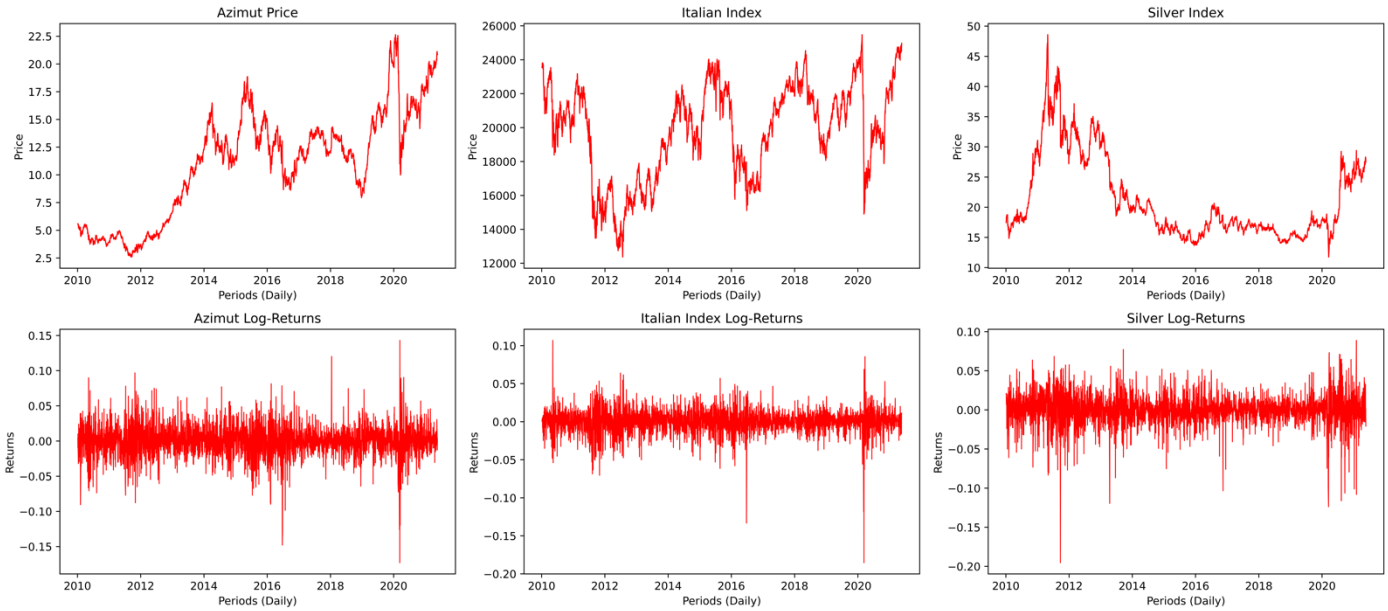
The process is covariance-stationary only if the coefficient β_1 is strictly lower than one. Furthermore, we have also the GARCH-GJR model, where σ_t^2 depends on both the size ε_{t-1}^2 and the sign of the lagged innovations ($\gamma_1 \neq \alpha_1 + \beta_1$).

$$\sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 + \gamma_1 \Pi_{t-1}^- \varepsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2 \quad ; \quad \Pi_t^- = \begin{cases} 1 \; if \; \varepsilon_t < 0 \\ 0 \; if \; \varepsilon_t \geq 0 \end{cases} \quad ; \quad E[\sigma_t^2] = \sigma^2 = \omega + \alpha_1 \sigma^2 + \beta_1 \sigma^2 + \frac{\gamma_1}{2} \sigma^2$$

The conditional variance is positive if we have $\omega > 0, \alpha_1 \geq 0,\ \alpha_1 + \gamma_1 \geq 0$ and $\beta_1 \geq 0$. Covariance-stationarity is satisfied if:

$$\alpha_1 + \frac{\gamma_1}{2} + \beta_1 < 1$$

Leverage effect is caused by the fact that negative returns have a bigger impact on future volatility than positive returns. This effect is included in the GJR-GARCH and EGARCH models. In the GJR model, if we have a negative ε_t then Π_t^- will be equal to 1. σ_t^2 will be then lower since we gave more importance on negative than positive return performance. Bad news have an impact of $\alpha_1 + \gamma_1$ on volatility and good news have an impact of α_1.
In this part, we will study the comportment of the GARCH model and of its extensions on real data. We decided to take the biggest Italian stock market index: the FTSE MIB composed of the fourthy most traded stock classes on the Italian stock exchange the "Borsa Italiana". We choose for analysis the stock of Azimut, an Italian asset management company based in Milan, Italy. Concerning the commodity index, we decided to choose the evolution of silver price.



We have the statistical properties of each returns time series including the annualized mean and annualized volatility:

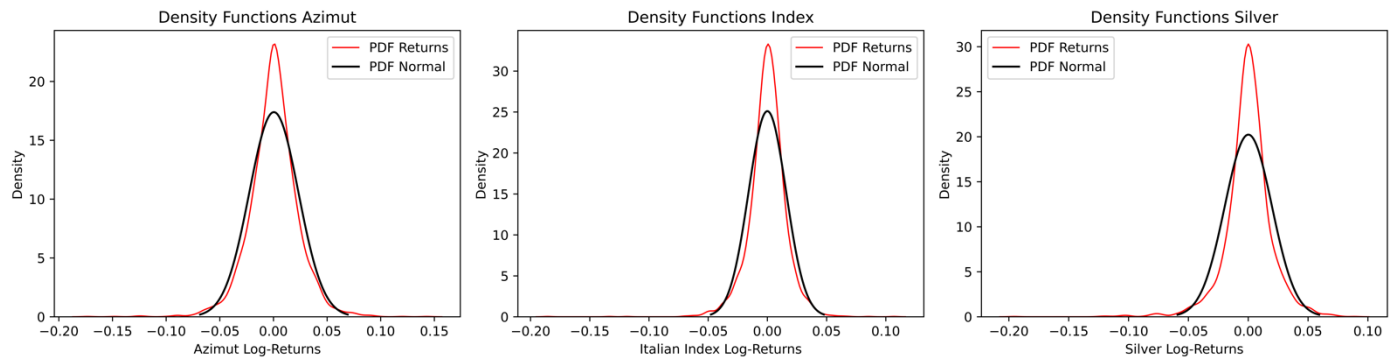
$$\hat{\mu}_i = \frac{1}{T}\sum_{t=1}^{T} r_{i,t} \quad ; \quad \hat{\sigma}_i^2 = \frac{1}{T-1}\sum_{t=1}^{T}\left(r_{i,t} - \hat{\mu}_i\right)^2$$

$$\hat{S}_i = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{r_{i,t} - \hat{\mu}_i}{\hat{\sigma}_i}\right)^3 \quad ; \quad \hat{K}_i = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{r_{i,t} - \hat{\mu}_i}{\hat{\sigma}_i}\right)^4$$

$$\hat{\mu}_{Annualized} = \hat{\mu}_{Daily} * 252 \quad ; \quad \hat{\sigma}_{Annualized} = \hat{\sigma}_{Daily} * \sqrt{252}$$

	Mean $\hat{\mu}_i^{Ann}$	*Volatility* $\hat{\sigma}_i^{Ann}$	*Skewness* $\hat{S}_i$	*Kurtosis* $\hat{K}_i$	*Minimum*	*Maximum*
Azimut	0.11522	0.364	-0.25435	7.287	-0.1731	0.14299
Italian Index	0.00514	0.2522	-0.939	14.063	-0.1855	0.10684
Silver Index	0.04091	0.31295	-0.995	10.8496	-0.1955	0.0928

The empirical distribution of each asset's log returns is given below in comparison to the normal distribution function:



Globally, all our time series have a large kurtosis, by far higher than three meaning that extreme positive or negative returns are more likely to occur. Their empirical distribution function differs strongly from the normal distribution.

First, we estimated the original GARCH model by using Maximum Likelihood maximization method. The log-likelihood is given below with the vector of parameters θ and $m = \max(p, q)$. We assume that residuals follow a normal distribution, so we use the normal pdf in the formula:

$$\mathcal{L}_T(\theta \parallel \varepsilon_{-m+1}, \dots, \varepsilon_0) = \prod_{t=1}^{T} \frac{1}{\sqrt{2\pi\sigma_t^2}} \exp\left[-\frac{\varepsilon_t^2}{2\sigma_t^2}\right] \quad ; \quad \log \mathcal{L}_T(\theta) = \sum_{t=1}^{T} -\frac{1}{2}\left[\log 2\pi + \log \sigma_t^2 + \frac{\varepsilon_t^2}{\sigma_t^2}\right] = \sum_{t=1}^{T} \log l_t(\theta)$$

The first order derivative is given by the gradient (1) and the second order derivative by the Hessian (2):

$$(1) : g(\theta) = \frac{\partial \log \mathcal{L}_T(\theta)}{\partial \theta} = \sum_{t=1}^{T} \frac{\partial \log l_t(\theta)}{\partial \theta} = \sum_{t=1}^{T} g_t(\theta) = \sum_{t=1}^{T} \left[\frac{\varepsilon_t^2}{2\sigma_t^4}\frac{\partial \sigma_t^2}{\partial \theta} - \frac{1}{2\sigma_t^2}\frac{\partial \sigma_t^2}{\partial \theta}\right] = \sum_{t=1}^{T} \left[\frac{1}{2\sigma_t^2}\left(\frac{\varepsilon_t^2}{\sigma_t^2} - 1\right)\frac{\partial \sigma_t^2}{\partial \theta}\right]$$

$$(2) : H(\theta) = \frac{\partial^2 \log \mathcal{L}_T(\theta)}{\partial \theta \partial \theta'} = \sum_{t=1}^{T} \frac{\partial^2 \log l_t(\theta)}{\partial \theta \partial \theta'} = \sum_{t=1}^{T} \left(-\left[\frac{1}{2\sigma_t^2}\frac{\partial \sigma_t^2}{\partial \theta}\right]\left[\frac{\partial \sigma_t^2}{\partial \theta'}\frac{\varepsilon_t^2}{\sigma_t^4}\right] + \left(\frac{\varepsilon_t^2}{\sigma_t^2} - 1\right)\frac{\partial}{\partial \theta'}\left(\frac{1}{2\sigma_t^2}\frac{\partial \sigma_t^2}{\partial \theta}\right)\right)$$

Then, we can estimate the Information matrix by computing the expectation of the Hessian. We also define the vector:

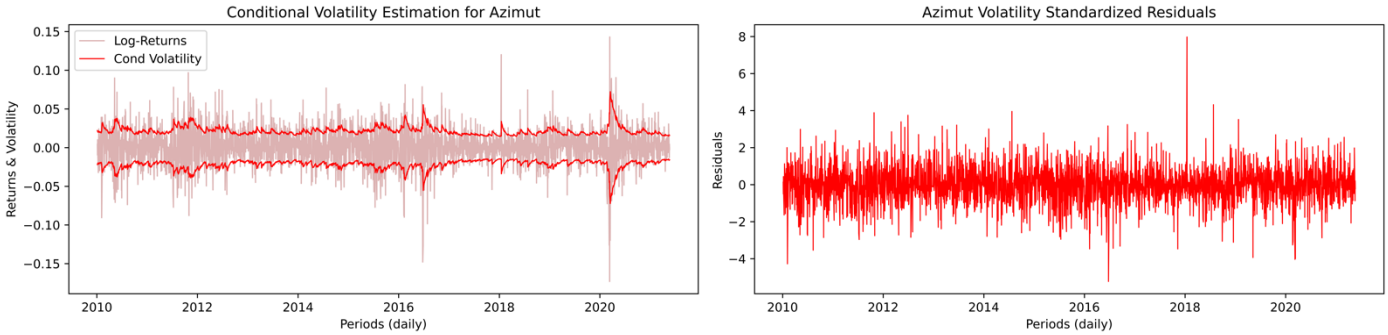
$$\xi_t = \left[1, \varepsilon_{t-1}^2, \dots, \varepsilon_{t-p}^2, \sigma_{t-1}^2, \dots, \sigma_{t-q}^2\right]'$$

$$I(\theta) = -E_0\left[\frac{\partial^2 \log l_t(\theta)}{\partial \theta \partial \theta'}\right] = -\frac{1}{T} E\left[\frac{\partial^2 \log \mathcal{L}_T(\theta)}{\partial \theta \partial \theta'}\right] \quad ; \quad \hat{I}(\theta) = \frac{1}{2T}\sum_{t=1}^{T} \frac{\hat{\xi}_t \hat{\xi}_t'}{\hat{\sigma}_t^4}$$

$$\hat{\xi}_t \hat{\xi}_t' = \begin{bmatrix} 1 & \hat{\varepsilon}_{t-1}^2 & \cdots & \hat{\varepsilon}_{t-p}^2 & \hat{\sigma}_{t-1}^2 & \cdots & \hat{\sigma}_{t-p}^2 \\ \hat{\varepsilon}_{t-1}^2 & \hat{\varepsilon}_{t-1}^4 & \cdots & \hat{\varepsilon}_{t-1}^2\hat{\varepsilon}_{t-p}^2 & \hat{\varepsilon}_{t-1}^2\hat{\sigma}_{t-1}^2 & \cdots & \hat{\varepsilon}_{t-1}^2\hat{\sigma}_{t-p}^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \hat{\varepsilon}_{t-p}^2 & \hat{\varepsilon}_{t-1}^2\hat{\varepsilon}_{t-p}^2 & \cdots & \hat{\varepsilon}_{t-p}^4 & \hat{\varepsilon}_{t-p}^2\hat{\sigma}_{t-1}^2 & \cdots & \hat{\varepsilon}_{t-p}^2\hat{\sigma}_{t-p}^2 \\ \hat{\sigma}_{t-1}^2 & \hat{\varepsilon}_{t-1}^2\hat{\sigma}_{t-1}^2 & \cdots & \hat{\varepsilon}_{t-p}^2\hat{\sigma}_{t-1}^2 & \hat{\sigma}_{t-1}^4 & \cdots & \hat{\sigma}_{t-1}^2\hat{\sigma}_{t-p}^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \hat{\sigma}_{t-p}^2 & \hat{\varepsilon}_{t-1}^2\hat{\sigma}_{t-p}^2 & \cdots & \hat{\varepsilon}_{t-p}^2\hat{\sigma}_{t-p}^2 & \hat{\sigma}_{t-1}^2\hat{\sigma}_{t-p}^2 & \cdots & \hat{\sigma}_{t-p}^4 \end{bmatrix}$$

By using this method, the estimates for the coefficients corresponding to Azimut log-returns are given below:

		Estimates	*Standard Dev.*	*t-stats*
Azimut	μ	0.1065	0.00375	2.844
	ω	0.1573	0.077	2.040
	α_1	0.0657	0.0146	4.499
	β_1	0.9026	0.0253	35.557



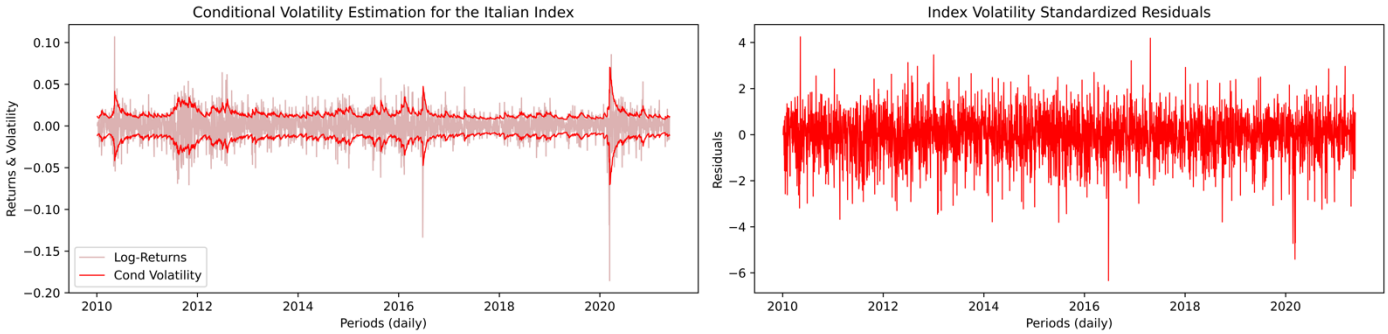
Now, we have to test if our estimates are statistically significant or not. By doing this, we test if the process is homoscedastic (our null hypothesis H_0) against the alternative that the conditional volatility is a GARCH (1,1) model:

$$H_0 : \alpha_1 = \beta_1 = 0 \; ; \; H_1 : \alpha_1 \geq 0, \beta_1 \geq 0$$

We assume that the t-stats (in the table above) are the result of the inverse Hessian Matrix. The critical values of this test are 1.96 and -1.96 for the threshold of 5%. A t-stat between these two values would mean no reject. We can notice that the t-stats for all estimates are strictly higher than 1.96, meaning that our estimates are statistically significant.

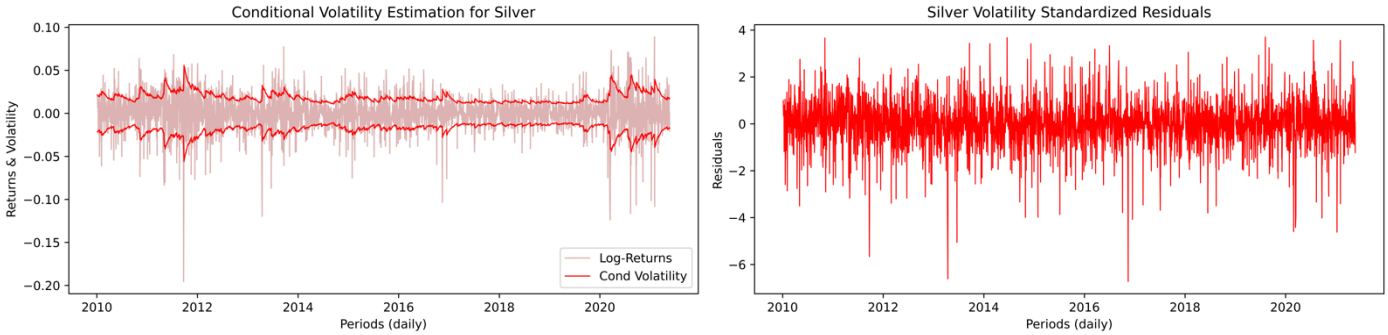
Besides, we have $\hat{\alpha}_1 + \hat{\beta}_1 = 0.9683$ very close to one and ω very close to zero. These values are close to the integrated, non-stationary bounds on the estimators. The sum of α_1 and β_1 is a measure of movements persistence in the variance for the GARCH model. So, a sum very close to one means that changes in our estimated conditional volatility is indeed persistent. When the interval from $\pm\sigma_t^2$ is narrow, then the conditional volatility is small. There are no leverage effects for now since in the classical GARCH model, positive and negative returns have the same impact on future volatility.
For the Italian index log-returns, we obtain the following estimation & graphical representations of the GARCH model:

		Estimates	*Standard Dev.*	*t-stats*
Index	μ	0.0471	0.02384	1.977
	ω	0.0544	0.01693	3.212
	α_1	0.1033	0.02054	5.03
	β_1	0.8772	0.02129	41.2



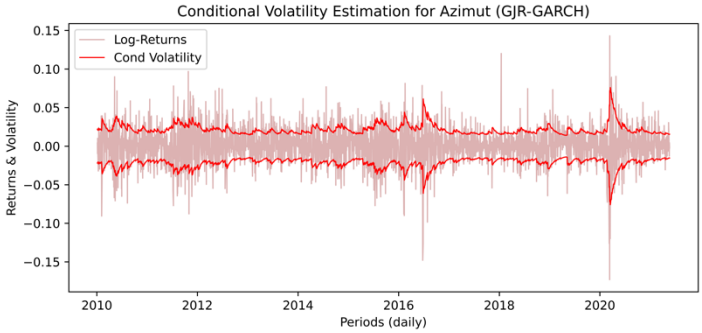
Our estimates are also very statistically significant if we compare our t-stats with the 5% threshold critical value 1.96. Again, if we compute $\hat{\alpha}_1 + \hat{\beta}_1$, we obtain $0.98 \approx 1$. The changes in conditional volatility for the Italian stock index are so persistent. Still, there are no leverage effects since we are estimating a GARCH model.
And finally, for the commodity index of the silver price evolution, the corresponding estimates are given by:

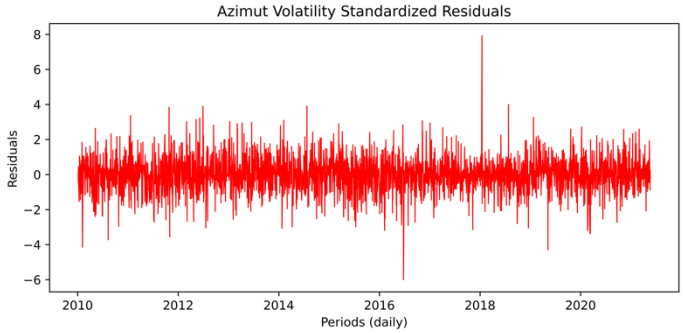
		Estimates	*Standard Dev.*	*t-stats*
Silver	μ	-0.00831	0.0309	-0.269
	ω	0.0406	0.0273	1.486
	α_1	0.0524	0.01969	2.664
	β_1	0.9379	0.02442	38.408



In this case, only the estimate of μ and ω are not statistically significant at the threshold of 5% because their test-statistics are between -1.96 and 1.96, so the null hypothesis is not rejected. If we compute the sum $\hat{\alpha}_1 + \hat{\beta}_1$, we obtain 0.99, meaning that, again, changes in conditional volatility are persistent since the sum is very large.
Now, we can then estimate the GARCH-GJR model. We obtained the following estimates for Azimut log-returns:

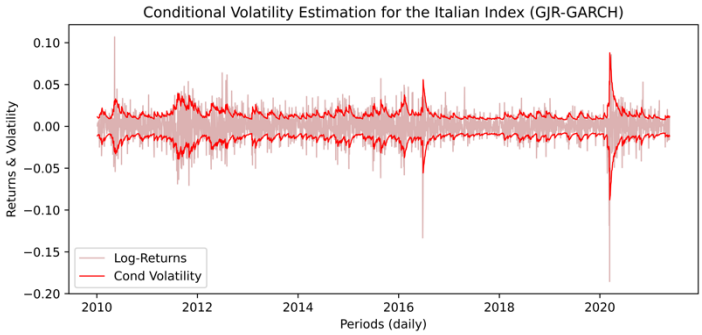
		Estimates	*Standard Dev.*	*t-stats*
Azimut	μ	0.0640	0.037	1.728
	ω	0.1274	0.04362	2.921
	α_1	0	0.01073	0
	γ_1	0.0872	0.01633	5.344
	β_1	0.9298	0.01975	47.07

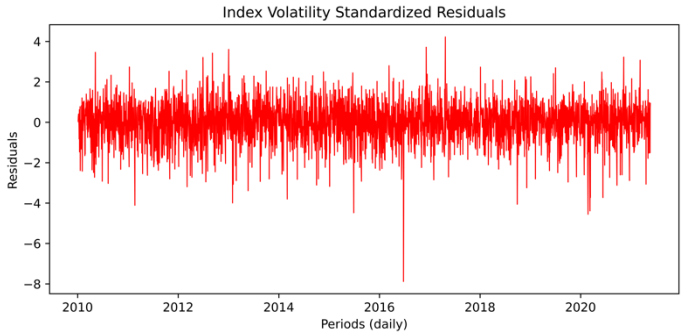




To test for statistical significance, we have to take into account that there is one more coefficient in comparison to the original GARCH model which is γ_1. So, we include in the hypothesis $H_0 : \gamma_1 = 0$.
Only $\hat{\alpha}_1$ is not statistically significant at the threshold of 5% and its estimator is equal to 0. This would mean that the moving average part of the conditional volatility (based on innovations ε) does not have a significant effect. If we compute $\alpha_1 + 0.5\gamma_1 + \beta_1$ we obtain $0.97 \approx 1$. Using this model, we found that there is persistency in conditional volatility for Azimut log-returns. Furthermore, the condition for leverage in the GJR is $\hat{\gamma}_1 > 0$ (meaning that Π_t^- is equal to 1 and ε_t is negative) since bad news produces more volatility than good news. We have $\hat{\gamma}_1 = 0.08$, so there is leverage effect.

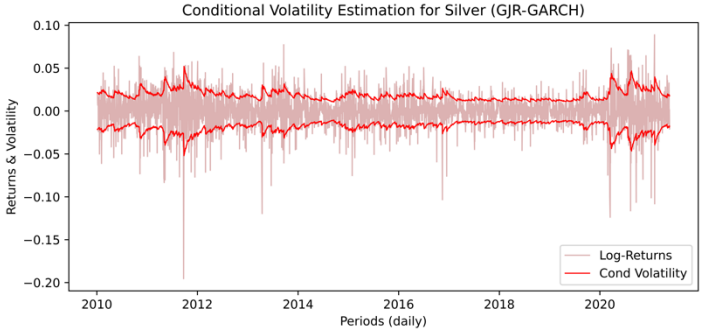
		Estimates	*Standard Dev.*	*t-stats*
Index	μ	0.0048	0.0232	0.207
	ω	0.0575	0.0177	3.247
	α_1	0.0051	0.01528	0.334
	γ_1	0.1549	0.02925	5.296
	β_1	0.892	0.01989	44.84

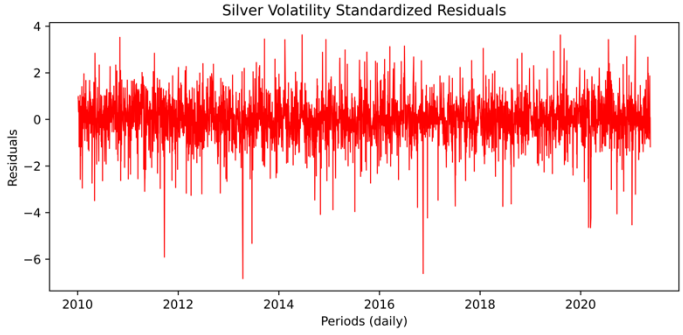




For the Italian stock Index, again the estimators of μ and α_1 are not statistically significant at the 5% threshold. Furthermore, $\alpha_1 + 0.5\gamma_1 + \beta_1 = 0.98$ so there is persistence in conditional volatility. Also, since $\hat{\gamma}_1 = 0.15$ and then higher than zero, we can say that there is a leverage effect in the Italian Index conditional volatility. Finally, for the silver index:

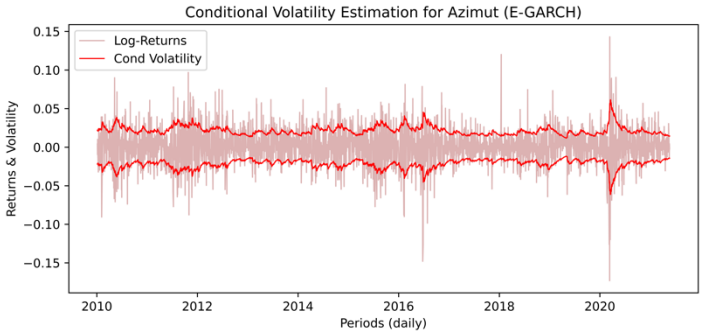
		Estimates	*Standard Dev.*	*t-stats*
Silver	μ	0.0044864	0.03101	0.145
	ω	0.0444	0.02715	1.634
	α_1	0.0711	0.02379	2.988
	γ_1	-0.0269	0.01696	-1.588
	β_1	0.9329	0.02386	39.102

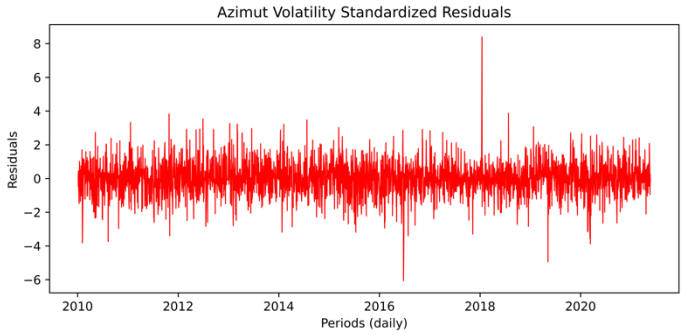




Our estimates for γ_1, μ and ω are not statistically significant at the 5% threshold since their t-statistics are between the upper and lower bound critical values. If we compute the sum of the coefficients, we obtain $\alpha_1 + 0.5\gamma_1 + \beta_1 = 0.99$ again, meaning that there is also persistency in volatility. Also, our estimation for γ_1 is lower than zero, meaning that there is not presence of leverage effect.
From now we used the E-GARCH Model to obtain the estimation of the conditional volatility for each time series:

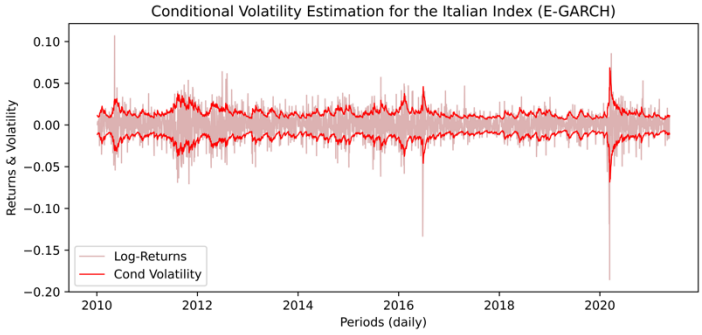
		Estimates	Standard Dev.	t-stats
Azimut	μ	0.0478	0.03913	1.222
	ω	0.0388	0.01159	3.345
	α_1	0.0798	0.02648	3.015
	γ_1	-0.0657	0.0115	-5.714
	β_1	0.9768	0.0071	137.398

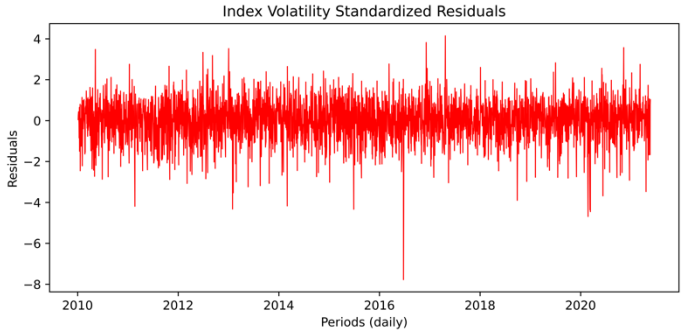




For Azimut, only the estimator of μ is not significant at the threshold of 5%. The condition for volatility persistency is the β coefficient close to 1. While looking at β_1, we have $0.9768 \approx 1$ so the process is covariance stationary and there is persistency in volatility changes. Besides, our estimates of γ_1 is equal to -0.066, which is negative. In the EGARCH model, there is a leverage effect if $\hat{\gamma}_1 < 0$. This is the case for Azimut here. Concerning the Italian stock Index, we have:

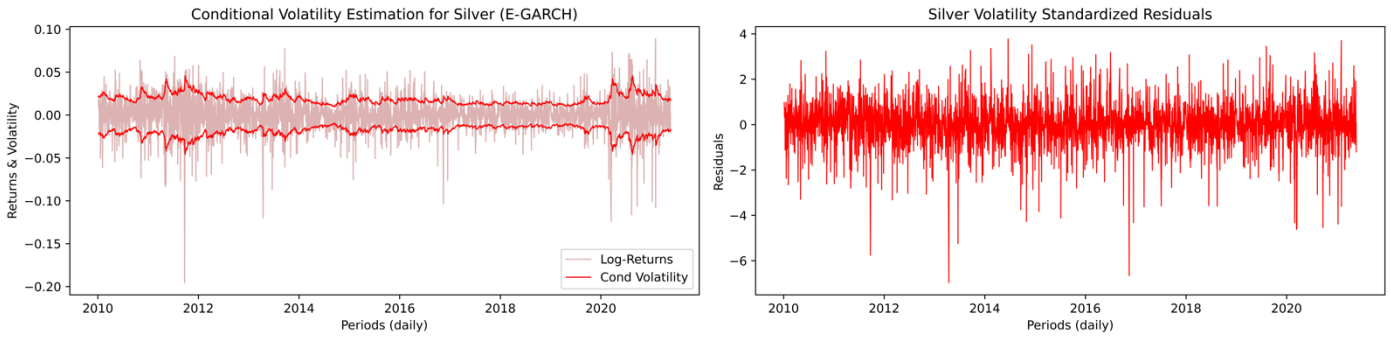
		Estimates	Standard Dev.	t-stats
Index	μ	-0.0000774	0.02257	-0.343
	ω	0.0229	0.002693	3.360
	α_1	0.1305	0.02693	4.846
	γ_1	-0.1289	0.0176	-7.339
	β_1	0.9717	0.00727	133.587





Again, for the Italian Stock Index, only the estimator of μ is not significant since the null hypothesis is rejected at the threshold of 5%. We have a coefficient β_1 =0.9717 $\approx$ 1, there is also persistency in volatility changes and the process is covariance stationary since the coefficient is not strictly equal to one. Our estimate of γ_1 is again negative, meaning that the leverage effect holds for the Italian Index log-returns. And finally, for the silver index:

		Estimates	Standard Dev.	t-stats
Silver	μ	-0.0131	0.001359	-9.648
	ω	0.0294	0.01349	2.181
	α_1	0.1367	0.0374	3.655
	γ_1	0.012	0.01382	0.868
	β_1	0.9848	0.008587	114.68



From these results, we obtained that our estimations for μ, γ_1 and ω are not statistically significant at the 5% threshold. We have a coefficient $\beta_1 = 0.9848 \approx 1$. Again, the process is stationary and there is a persistency effect in our estimated conditional volatility. Besides, $\hat{\gamma}_1$ is equal to 0.012 > 0, meaning that there is not the presence of a leverage effect.

- *Models Comparison:*

Now, we want to check which model is appropriate for our log returns time series. But first, we have to see which models are nested or not, in order to select the optimal test to conduct on our time series. In the EGARCH and GJR-GARCH models, we assume that there is another coefficient γ_1 in comparison to the original GARCH model. However, in the EGARCH the lagged volatilities are expressed and transformed in logs. So, it is not the same equation with the GARCH. The latter is in consequence not nested in the EGARCH and it is not possible to conduct Likelihood ratio tests with these two models, since this test is optimal for nested models only. But the GARCH is nested in the GJR extended model, since the only difference between the two is the presence of the coefficient γ_1. We can then assume that the GARCH model is the constrained model (with a coefficient $\gamma_1 = 0$), the GJR-GARCH is the unconstrained model. The Likelihood Ratio Test is defined with the null hypothesis H_0 that the real model is the constrained one. The Test statistic, with the likelihood of the unconstrained model U and of the constrained one R, is given by (with the number of parameters to estimate K). Under the null hypothesis, the test follows a chi-squared distribution at the α % threshold:

$$LR = 2(\log \mathcal{L}_U - \log \mathcal{L}_R) > 0 \xrightarrow{d} \chi^2(J) \quad ; \quad J = K_U - K_R = 4 - 3 = 1$$

In our case, $J = 1$ because we made one restriction on the constrained model. We reject H_0 if we have $LR \geq \chi^2_{1-\alpha}(1)$.

	Threshold at 1%	*Threshold at 5%*	*Threshold at 10%*
Critical Values $\chi^2_{1-\alpha}(1)$	6.635	3.841	2.706

However, there exists some tests that permits to compare non-nested models like the Vuong test for instance, which imply the null hypothesis of both models tested are relatively closed following test-statistic:

$$Z = \frac{\sum_{i=1}^{N} z_i}{\sqrt{N}\sigma_z} \sim \mathcal{N}(0,1) \quad ; \quad z_i^{1,2} = \log \mathcal{L}_1(x_i \parallel \theta_1) - \log \mathcal{L}(x_i \parallel \theta_2)$$

With two models 1 and 2. To estimate the volatility of z_i, we use some robust estimator such as Newey West. The null hypothesis is rejected if the test Z is not between the critical values $z_{1-\alpha}$ and $-z_{1-\alpha}$ with the α % confidence threshold from the Normal distribution. The critical values are 1.645, 1.96 and 2.576 for the levels 10, 5 and 1% respectively. If the test-statistic is above the positive bound, then the model 1 is preferred. If it is lower than the negative bound, then it is the model 2 that is preferred. First, we began with Azimut models comparison. The log-likelihood corresponding to each model (GARCH, EGARCH and GJR-GARCH) are given below:

	GARCH	*GJR-GARCH*	*EGARCH*
Log-likelihood Azimut	-6287.95	-6255.21	-6261.18

If we compare the GARCH model with the GJR-GARCH, we found a LR statistic equal to 65.5, which is by far higher than any of our critical values. We reject the null hypothesis that the right model for Azimut log-returns is the GARCH. If now we compare the GARCH model with the EGARCH, we used the Vuong test and we found a test-statistic equal to 2.25, meaning that both models are not exactly equivalent at the 5% threshold. The E-GARCH is then more adapted.

If finally, we compare the EGARCH and the GJR-GARCH, we obtained a test-statistic of -0.757. This value is very small and the null hypothesis that both models are equivalent is not rejected. We conclude that using either the EGARCH or the GJR-GARCH is better than using the GARCH model to estimate the conditional volatility.
Then, for the Italian Stock Index log returns, the following values for the log-likelihood for each model were obtained:

	GARCH	*GJR-GARCH*	*EGARCH*
Log-likelihood Italian	-5073.66	-5025.4	-5005.74

We compared the GARCH model with the GJR-GARCH. The corresponding LR statistic is 96.5 and it is higher than the critical values for all thresholds 1, 5 and 10%. We conclude that the most adapted model between the two in this test is the GJR-GARCH model. If we compare this time the GARCH model with the EGARCH, the test statistic of the Vuong test is 4.1 so the null hypothesis of equivalence between both model is rejected and the EGARCH model is the most adapted. Finally, a test statistic of 2.87 is obtained from the Vuong test if we compare the EGARCH and the GJR-GARCH. The null hypothesis is then rejected and using the E-GARCH model is preferable to estimate the conditional volatility for the Italian stock Index log returns.

Finally, we have the log-likelihood values for each models concerning the Silver Index log returns in the following table:

	GARCH	*GJR-GARCH*	*EGARCH*
Log-likelihood Silver	-5679.67	-5676.048	-5673.85

We began to compare the GARCH and GJR-GARCH together. A LR test-statistic is computed and equal to 7.25. H_0 is then rejected by very small margin and the most adapted model is the GJR-GARCH (so the unconstrained model). If we compare the GARCH and EGARCH, we obtained a Vuong test equal to 0.77 so the null hypothesis is again rejected, and the two models are equivalent. Finally, we compared the EGARCH and the GJR-GARCH. The corresponding test-statistic using the Vuong test is 0.38 so again there is equivalence between both models estimated. The results seem to indicate that the three models are equivalent if we want to estimate the conditional volatility of the Silver Index.

- *Test for normality of the residuals:*

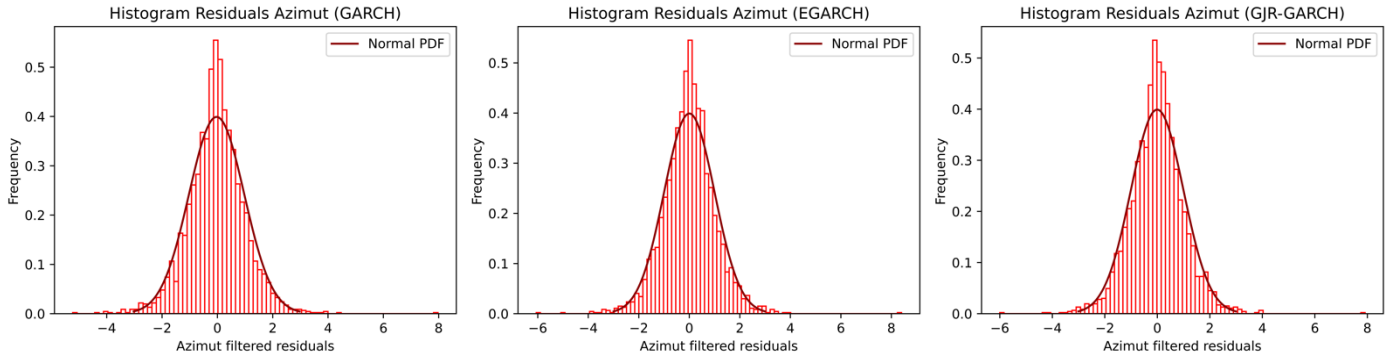
From now, we will check if the residuals of each estimation are close to the Normal distribution or not. First, we must compute the filtered residuals by applying the following formula and we assumed that the distribution is normal:

$$z_t = \frac{r_t - \mu}{\sigma_t} = \frac{\varepsilon_t}{\sigma_t}$$

One way to test for normality the data is the well-known Jarque-Bera Test. This is a type of test that permits to determine if the data distribution follows a Normal Distribution by using the standardized Skewness and Kurtosis. The T corresponds to the sample size. We run the test with a 5% threshold, meaning that the test is rejected if we have:

$$JB\ Test = T * \left[\frac{\hat{S}^2}{6} + \frac{(\hat{K} - 3)^2}{24}\right] \geq \chi^2_{0.95}(2) = 5.991$$

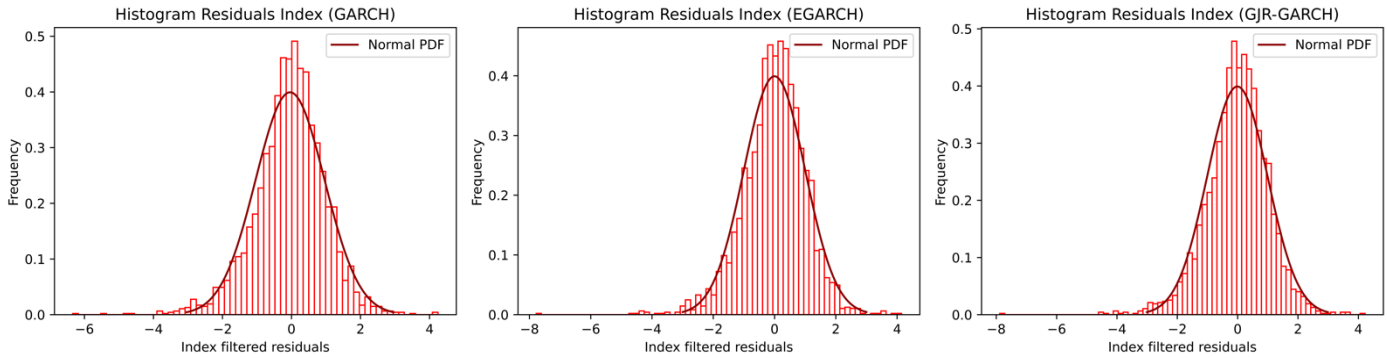
At first, let's have a look on the residuals from the models' estimations of the Azimut log-returns. Their histogram and density functions are given below with a comparison to the normal distribution:



As we can see, the distribution of the residuals of each model doesn't seem to fit the normal distribution. We can guess that the distribution has a strong kurtosis with large tails. If we apply the Jarque-Bera test, we obtained a test-statistic equal to 837.6 for the GARCH, 1092.2 for the EGARCH and 809.9 for the GJR-GARCH, meaning that the null hypothesis of normality is rejected. We were expecting this conclusion with the previous graphs but also by looking at the Skewness and the kurtosis of each filtered residual's series:

	GARCH	*GJR-GARCH*	*EGARCH*
Skewness	0.097	0.1	0.084
Kurtosis	5.6	5.59	6

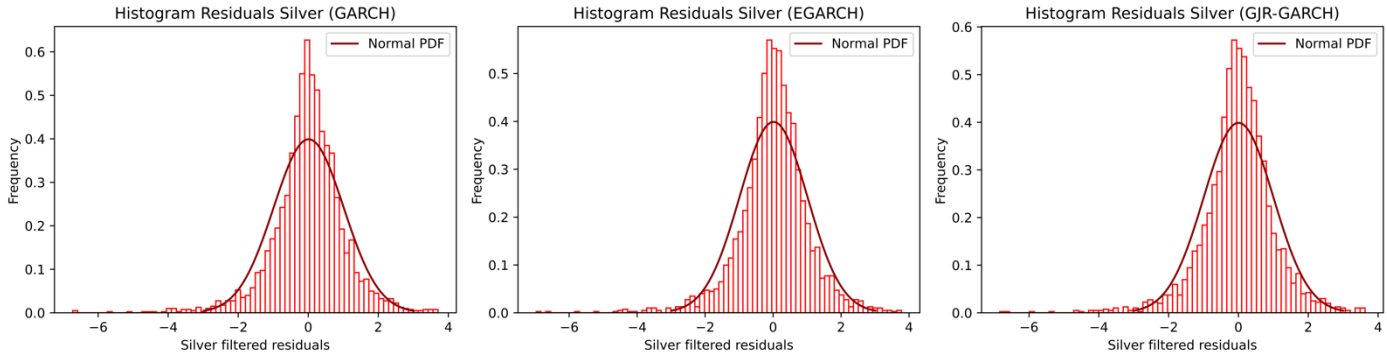
Concerning the Italian Stock Index, the histogram of residuals from the different estimation's models are given below:



Again, the distribution of each residual is far away from a normality distribution characteristic with very fat tails. The Jarque-Bera leads to the test statistic of 504.7, 845 and 830.5 for the GARCH, EGARCH and GJR-GARCH models respectively. So, the null hypothesis is rejected for each model. The values of the skewness and the kurtosis are below:

	GARCH	*GJR-GARCH*	*EGARCH*
Skewness	-0.45	-0.51	-0.53
Kurtosis	4.84	5.42	5.4

And finally, for the silver index the histogram of each residual's estimates is represented as the following, and we can see again that their distribution does not fit the normal distribution since we can notice the presence of fat tails:



If we run the Jarque-Bera test, the result obtained is equal to 1896.4 for the GARCH, 2098.7 for the EGARCH and 2045.5 for the GJR-GARCH. H_0 is in consequence rejected again for each model and the distribution of the filtered residuals is not a normal one. We can also notice that by looking at the following table:

	GARCH	*GJR-GARCH*	*EGARCH*
Skewness	-0.61	-0.63	-0.64
Kurtosis	6.8	6.96	7

We can conclude that for all our time series, the estimations made by using several GARCH models lead to non-normal residuals. We will have to find an adapted distribution that could fit them. Indeed, using the wrong distribution to estimates the residuals could leads to wrong estimations so it would be primordial to find the best fitting distribution.

- *Empirical distributions fitting:*

From now, we will try to fit three different non-gaussian distributions to our residuals. We need to find the best conditional distribution such as $z_t \sim g(z_t \parallel \eta)$ where η represents the parameters describing the distribution. We will try to fit the Skewed-t distribution, the generalized hyperbolic distribution and the gaussian mixture. By doing so, we need to estimate parameters by using the Maximum Likelihood maximization. After these steps, we need to compute a Kolmogorov and Smirnov Test for each series in order to see if the selected distribution with the estimated parameters fits the residuals data. It is one of the simplest measures of the difference between two Cumulative Density Functions. To compute the test, we must follow different steps, explicated below. We have the assumed theoretical CDF $F^*(x\,;\theta)$ compared with the empirical CDF $G_T(x)$, where θ is the vector of parameters.

1)	*Sort data in increasing order with* $r_1^* \leq \cdots \leq r_T^*$
2)	*Denote the new sample* $\{r_t^*\}_{t=1}^T$
3)	*By construction, we obtained* $G_T(r_t^*) = t/T$
4)	*Evaluate the assumed CDF* $F^*(r_t^*\,;\theta)$ *for all values*
5)	*Finally, compute the following test statistic:*

$$KS = \max_{t=1,\dots,T} |G_T(x) - F^*(x;\theta)| = \max_{t=1,\dots,T} \left| F^*(r_t^*) - \frac{t}{T} \right|$$

The null hypothesis that the distribution follows the theoretical distribution is rejected at the 5% threshold if we have test statistic higher than $1.36/\sqrt{T}$. First, let's try with the skewed-t distribution. It captures the fat tails of financial returns and the asymmetric properties of the distribution. By assuming that parameters are dependent on past realizations, moments can be time varying. The skewed student distribution is characterized by the following density function:

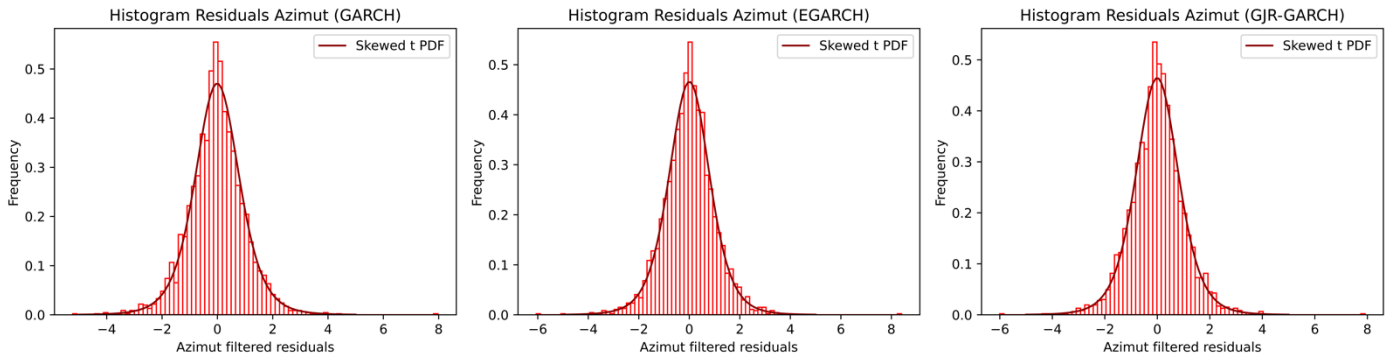
$$g(z_t \parallel \eta) = b * c * \left[1 + \frac{\xi_t^2}{\nu - 2}\right]^{-\frac{\nu+1}{2}}$$

$$\xi_t = \begin{cases} \frac{bz_t + a}{1-\lambda} & if z < -\frac{a}{b} \\ \frac{bz_t + a}{1+\lambda} & if\ z \geq -\frac{a}{b} \end{cases} \quad ; \quad a = 4\lambda * c * \frac{\nu - 2}{\nu - 1} \quad ; \quad b^2 = 1 + 3\lambda^2 - a^2 \quad ; \quad c = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\pi(\nu-2)} * \Gamma\left(\frac{\nu}{2}\right)} \quad ; \quad \eta = \begin{bmatrix} \nu \\ \lambda \end{bmatrix}$$

$$Gamma \Rightarrow \Gamma(x) = \int_0^\infty t^{x-1} e^{-t}\, dt \quad ; \quad E[Z_t] = 0 \ ; \ Var(Z_t) = 1$$

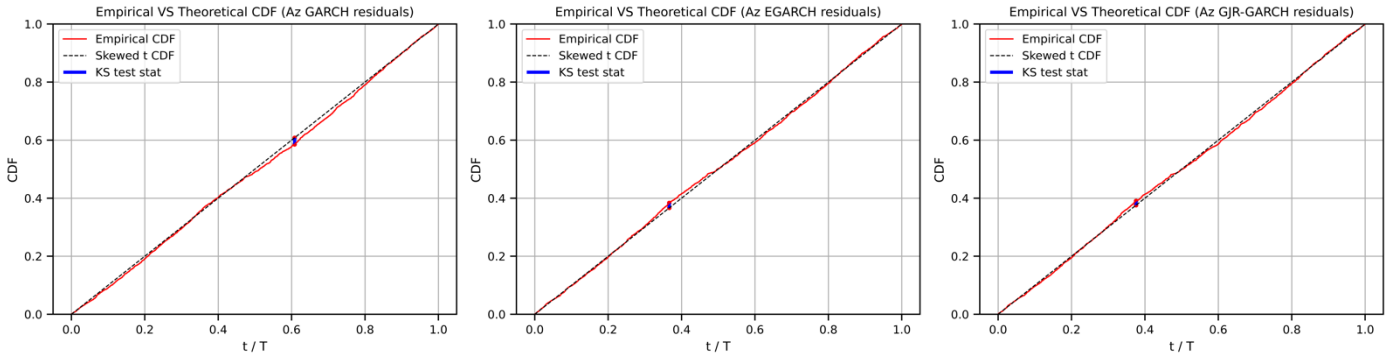
With the degree-of-freedom parameter $2 < \nu < \infty$ and asymmetry parameter $-1 < \lambda < 1$. If we have $\lambda = 0$, it is a student distribution. If $\lambda = 0$ and $\nu = \infty$, we obtain a normal distribution. We tried to fit this distribution with our residuals, and we estimated the different parameters by maximum likelihood for at first Azimut standardized residuals:

	GARCH	*GJR-GARCH*	*EGARCH*
Estimation λ	0.00446	0.00166	-0.00435
Estimation ν	5.89	6.2875	6.18



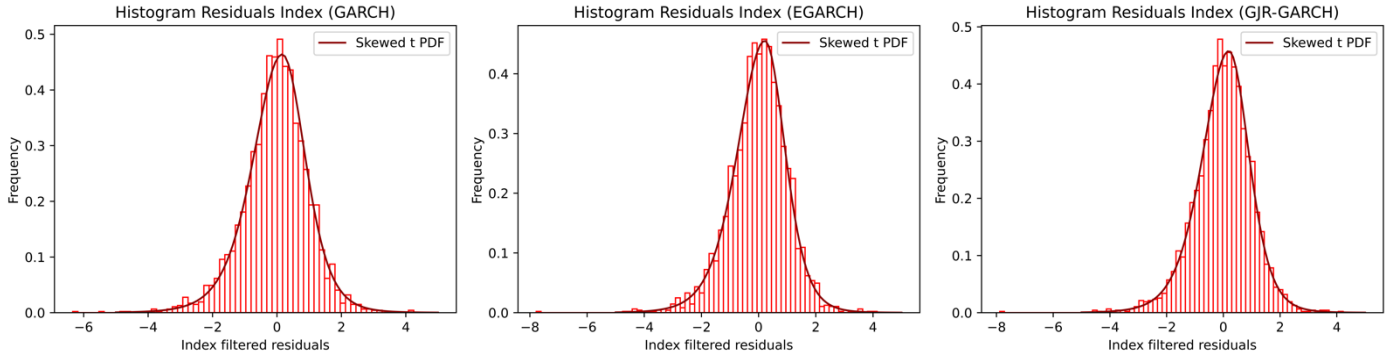
As we can see, the skewed t distribution seems to fit better the data than the normal distribution for all our residuals.

So, in our case the theoretical distribution $F^*(x\,;\theta)$ is the skewed student distribution with the vector of estimated parameters η. If we run the Kolmogorov Smirnov test, we obtain the results of 0.023 for the GARCH residuals, 0.018 for the EGARCH residuals and 0.015 for the GJR-GARCH residuals. The critical value at the 5% threshold is 0.025 so the null hypothesis is not rejected for all our residuals. The skewed student distribution fit the Azimut residuals.

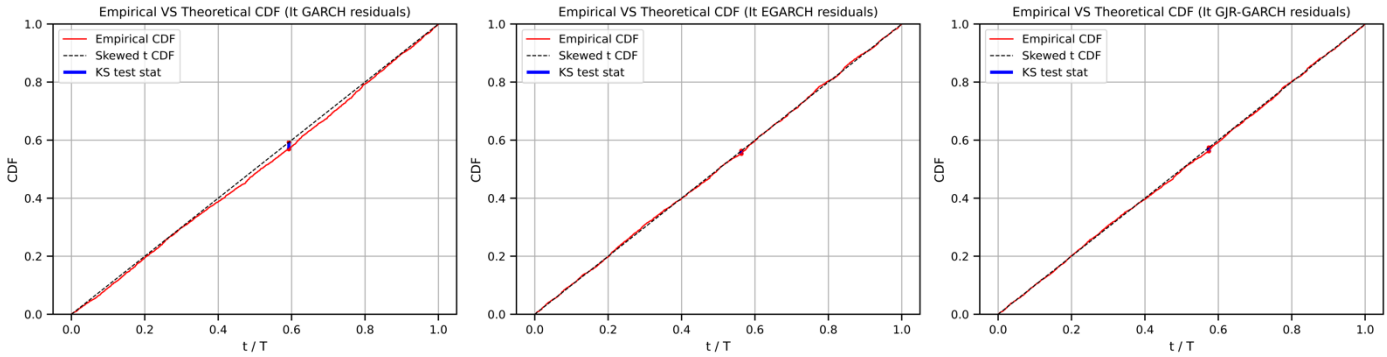


For the Italian Stock Index standardized residuals, we obtained the following estimates:

	GARCH	*GJR-GARCH*	*EGARCH*
Estimation λ	-0.11	-0.132	-0.145
Estimation ν	6.46	7.04	7.35

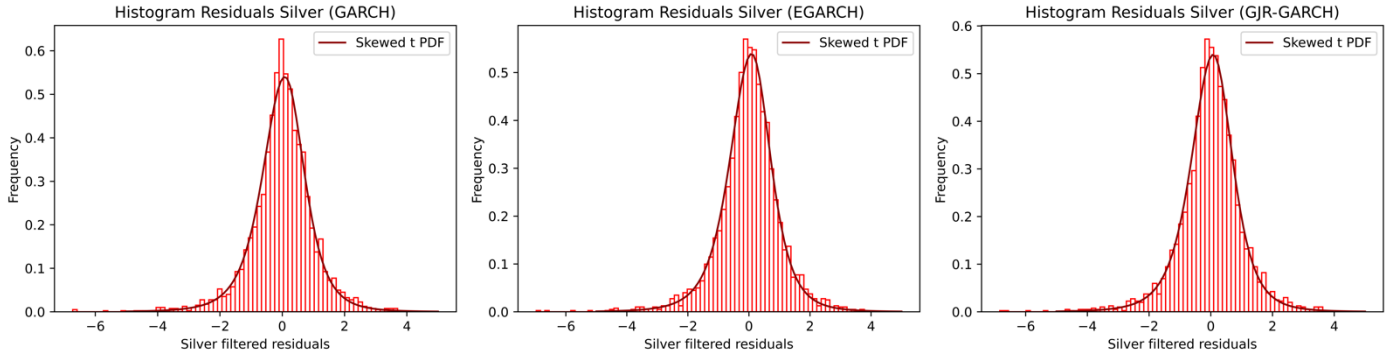


Again, we can see that the distribution also fits the data better than the normal distribution in this case. The results of the KS test are given by 0.023, 0.0097 and 0.0122 for the GARCH, EGARCH and GJR-GARCH residuals respectively. The null hypothesis is then not rejected (because our critical value is stile 0.025), and the true distribution is the theoretical one. We conclude that each residual of the Italian Index does fit with the skewed student. We can illustrate this statement by having a look on the representation of the empirical Cumulative distribution function:

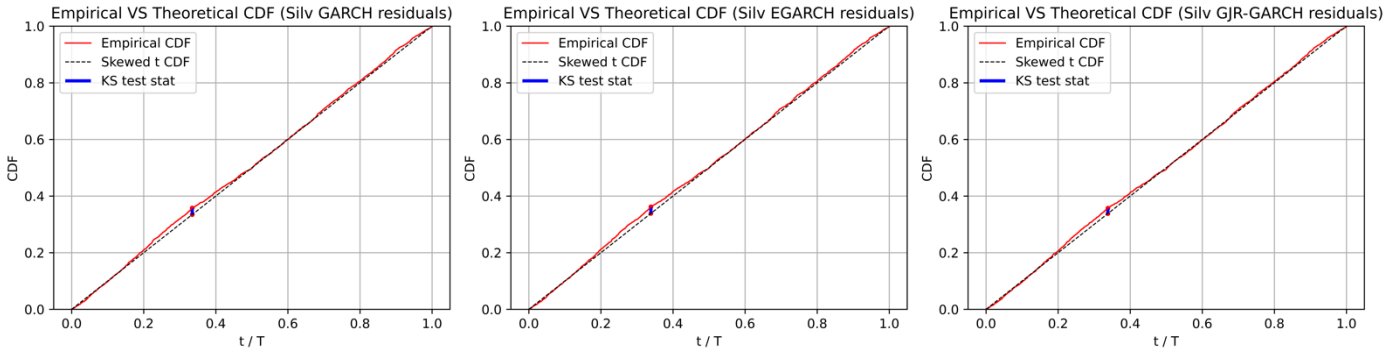


And finally, for the silver index, we used again the maximum likelihood method to estimate the following parameters:

	GARCH	*GJR-GARCH*	*EGARCH*
Estimation λ	-0.061	-0.057	-0.065
Estimation ν	3.85	3.86	3.87



Again, the distribution seems to fit very well the data. If we run the Kolmogorov Smirnov test, we obtained a test-statistic equal to 0.024 for the GARCH residuals, 0.023 for the EGARCH residuals and 0.0194 for GJR-GARCH residuals, meaning that the null hypothesis is not rejected for our models.

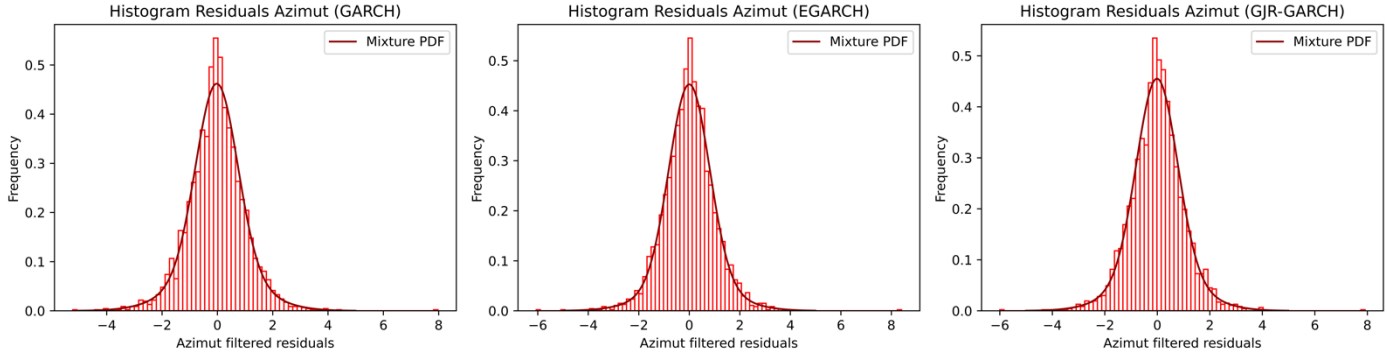


As we just saw previously, the skewed student distribution fit perfectly our data (for all GARCH models also). Using this distribution seems to be correct to estimate financial models based on returns. From now, we will try to fit another non-gaussian distribution: the Gaussian Mixture distribution. Its density function can be represented as follow with the gaussian density function of the first distribution and the density function of the second distribution:

$$g(z_t \parallel \phi, \mu_1, \sigma_1, \mu_2, \sigma_2) = \phi * g_1(z_t \parallel \mu_1, \sigma_1) + (1 - \phi) * g_2(z_t \parallel \mu_2, \sigma_2)$$

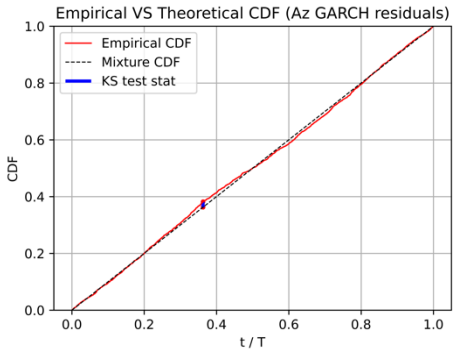
We tried again to fit this distribution with our residuals, and we estimated the different parameters by maximum likelihood for at first Azimut standardized residuals:

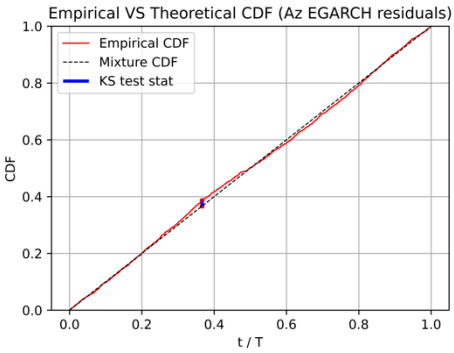
	GARCH	*GJR-GARCH*	*EGARCH*
Estimation ϕ	0.73	0.246	0.799
Estimation μ_1	-0.02	0.0197	0.00055
Estimation σ_1	0.749	1.494	0.794
Estimation μ_2	-0.038	-0.012	0.0111
Estimation σ_2	1.48	0.773	1.57

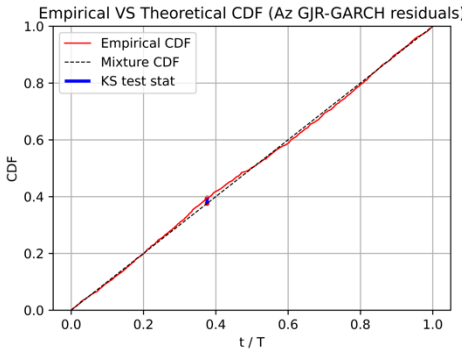


The above density function is then the result of the mixture between two different gaussian densities distribution. It seems that the result fits the data. To verify it, we will perform a Kolmogorov Smirnov test.

After performing it, we obtained the result concerning the KS test-statistic of 0.0195 for the GARCH residuals, 0.0205 for the EGARCH residuals and 0.019 for the GJR-GARCH residuals. Our values are lower than the critical value, so the null hypothesis is not rejected, and the empirical distribution is equivalent to the Gaussian Mixture distribution.

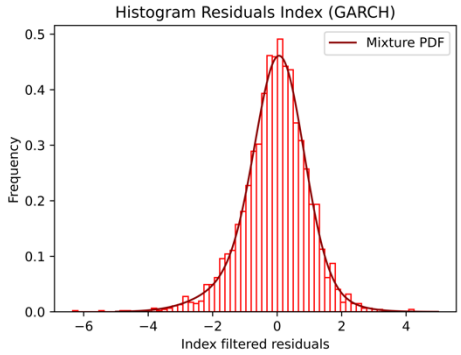


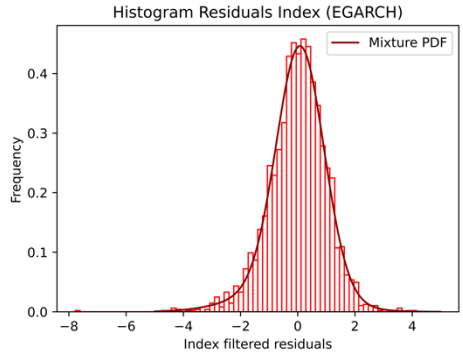


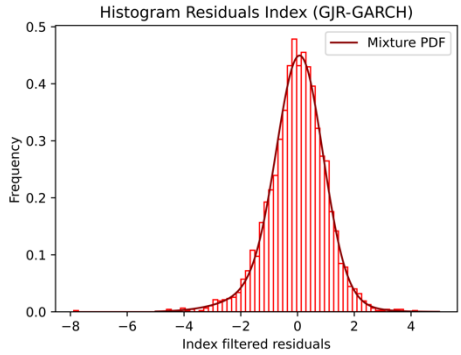


Concerning the residuals of the Italian Stock Index, the following coefficients' estimates were obtained:

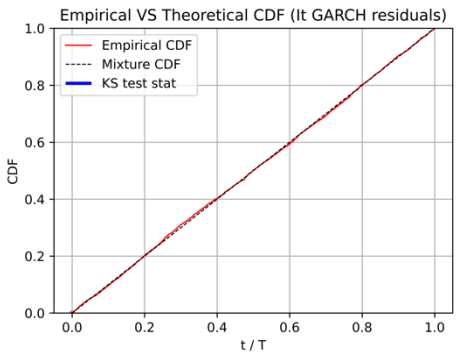
	GARCH	*GJR-GARCH*	*EGARCH*
Estimation ϕ	0.286	0.826	0.856
Estimation μ_1	-0.35	0.088	0.089
Estimation σ_1	1.4	0.81	0.828
Estimation μ_2	0.084	-0.46	-0.525
Estimation σ_2	0.74	1.542	1.59

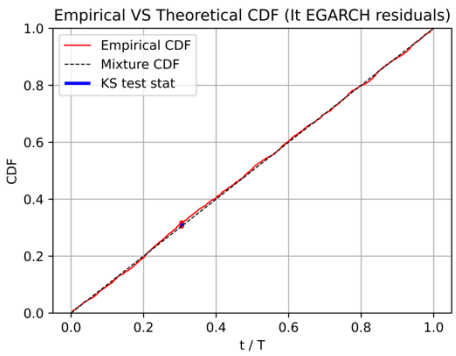


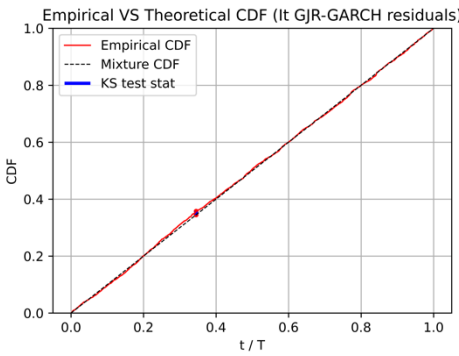




In this case also, the distribution seems to fit the data and if we run the KS test, we obtained value of 0.011 for the GARCH residuals, 0.0125 for the EGARCH residuals and 0.01297 for the GJR-GARCH residuals. Again, we do not reject the null hypothesis that the empirical and theoretical distribution are equivalent. For the Italian Stock Index, the Gaussian mixture fits very well the data also. We can illustrate this fact by having a look on the following graphs that represents the comparison between the empirical and theoretical Cumulative Distribution function.

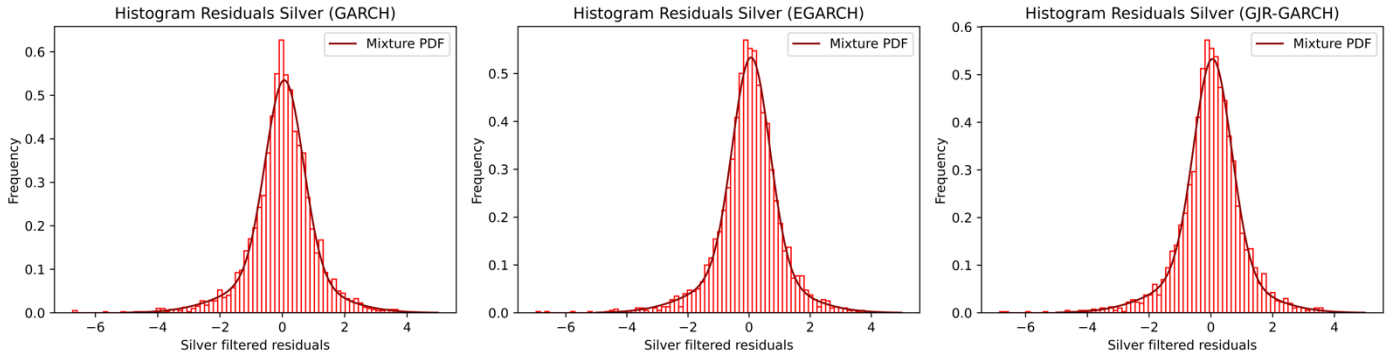




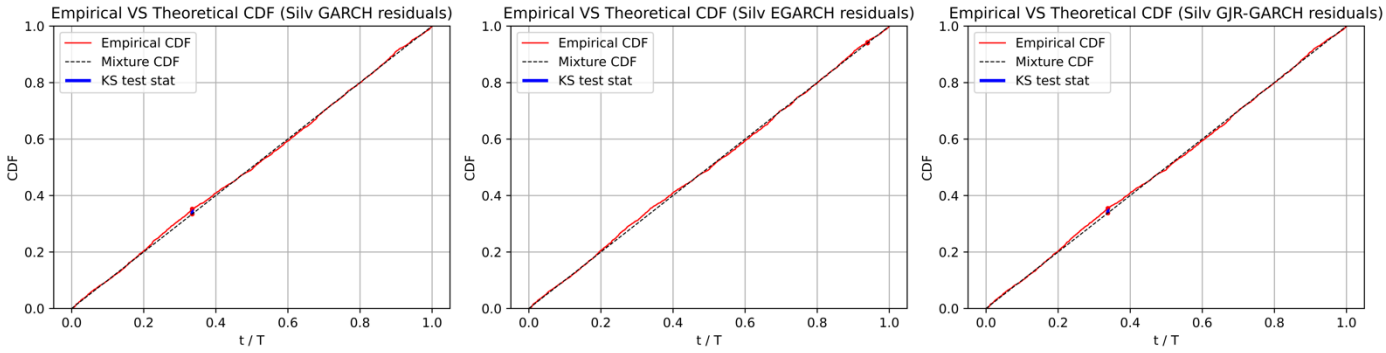


And finally, we obtained for the Silver Index residuals the following estimates while using Maximum Likelihood estimation with the density function:

	GARCH	GJR-GARCH	EGARCH
Estimation ϕ	0.2996	0.703	0.298
Estimation μ_1	-0.11	0.0574	-0.1142
Estimation σ_1	1.57	0.613	1.565
Estimation μ_2	0.065	-0.119	0.0688
Estimation σ_2	0.609	1.569	0.611



Again, it seems that the estimated distribution with our parameters fits the residuals of each time series. After running the Kolmogorov test, we obtain a KS test of 0.0179 for GARCH residuals, 0.023 for EGARCH residuals and 0.0168 for GJR-GARCH residuals. So, the null hypothesis that the empirical and theoretical distribution are equivalent is not rejected. Then, the distribution of the silver index residuals can be estimated using a Gaussian mixture (or Bi-modal distribution). Indeed, very satisfying results are obtained if we look below:

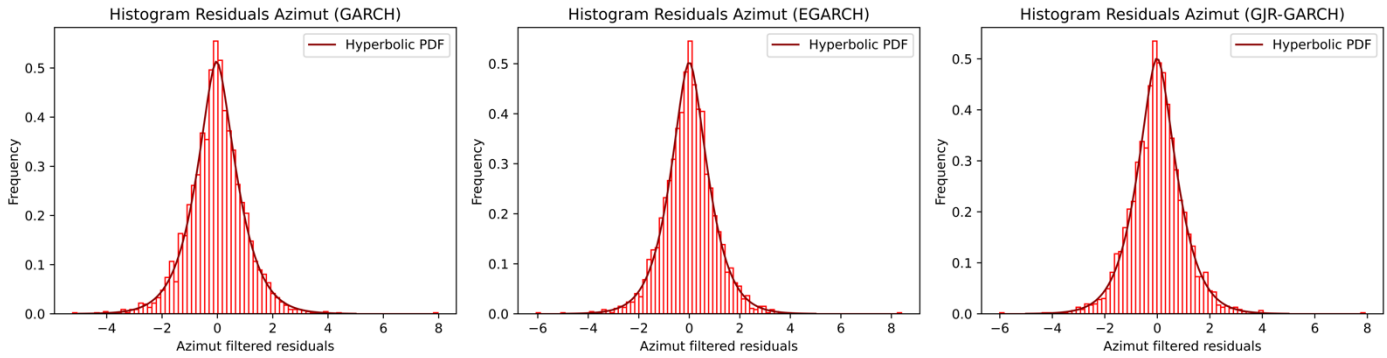


So, in conclusion, using the Gaussian Mixture could lead to very satisfying results in order to fit the empirical data. Finally, to finish this analysis we will try to fit a last distribution to our residuals: the Generalized Hyperbolic distribution. The density function can be represented as the following:

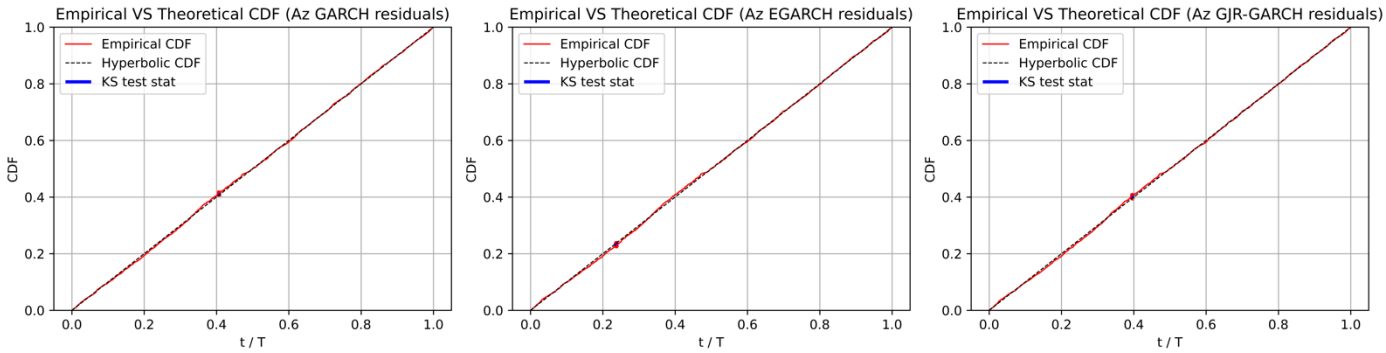
$$g(z_t \parallel \lambda, \alpha, \beta, \delta, \mu) = \frac{\left(\sqrt{\alpha^2-\beta^2}/\delta\right)^{\lambda}}{\sqrt{2\pi}K_{\lambda}\left(\delta\sqrt{\alpha^2-\beta^2}\right)} e^{\beta(x-\mu)} \frac{K_{\lambda-0.5}\left(\alpha\sqrt{\delta^2+(x-\mu)^2}\right)}{\left(\sqrt{\delta^2+(x-\mu)^2}/\alpha\right)^{0.5-\lambda}}$$

With the modified Bessel function $K_{\nu}(x)$ of the third kind. We have also $\delta > 0$ and $\alpha > |\beta| > 0$. It was proved in several studies that this distribution led to some very satisfying results when it comes to fit return time series. It encompasses fat tails (not too fat) and various degrees of freedom.
We begin to estimate the parameters for the Azimut standardized residuals. We obtained the following estimates:

	GARCH	GJR-GARCH	EGARCH
Estimation λ	1.84	1.99	1.98
Estimation α	1.922	2.004	1.9989
Estimation β	0.01	0.01	0.01
Estimation δ	0.01	0.01	0.01
Estimation μ	-0.031	-0.0116	-0.0034

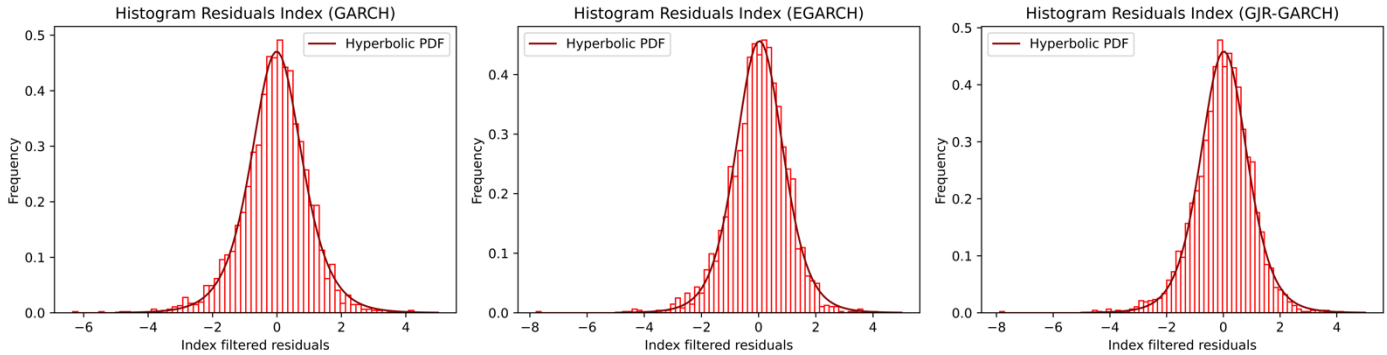


So, applying this distribution by using the previously estimated coefficients seems to fit very well all our data. We run the Kolmogorov Smirnov test and the corresponding value of the KS test for the GARCH, EGARCH and GJR-GARCH residuals are 0.0092, 0.0099 and 0.0105. So, H_0 is not rejected since these values are lower than the critical value at the 5% threshold. We can then say that the Hyperbolic distribution fits very well the Azimut data. The difference between the empirical and theoretical distribution can be seen below:

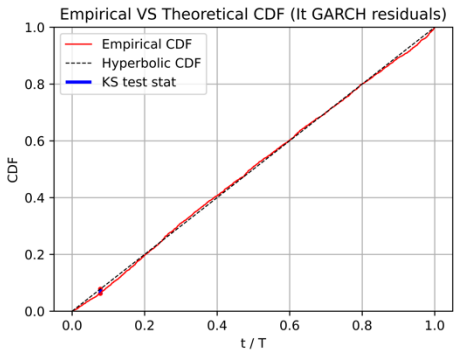


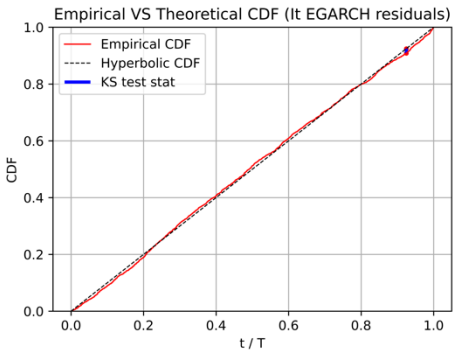
For the residuals of the Italian Stock Index, the following estimates were obtained while running the Maximum likelihood maximization method:

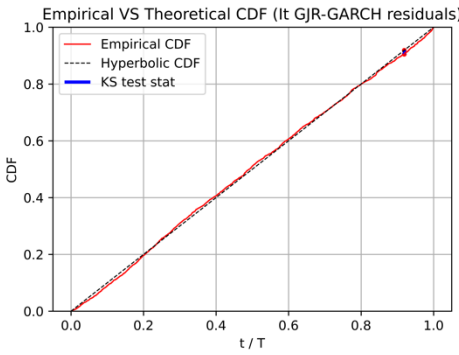
	GARCH	*GJR-GARCH*	*EGARCH*
Estimation λ	-0.997	-3.41	-3.539
Estimation α	1.19	0.02	0.02
Estimation β	0.01	0.01	0.01
Estimation δ	1.488	2.189	2.245
Estimation μ	-0.0143	0.015	0.0236



Again, the distribution seems to fit the data as well for the Italian Stock Index. While running the KS test, we obtain a test-statistic of 0.0156 for the GARCH residuals, 0.0165 for the EGARCH residuals and 0.014 for the GJR-GARCH residuals. Again, none of these values are greater than the critical value at the 5% threshold so the Hyperbolic distribution fit as well the Italian Stock Index residuals. We can check it by looking at the following difference between the observed distribution and the theoretical one corresponding to the Hyperbolic distribution.

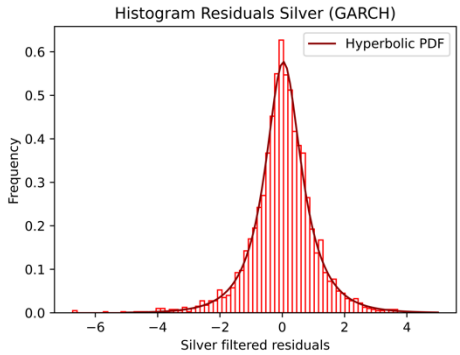


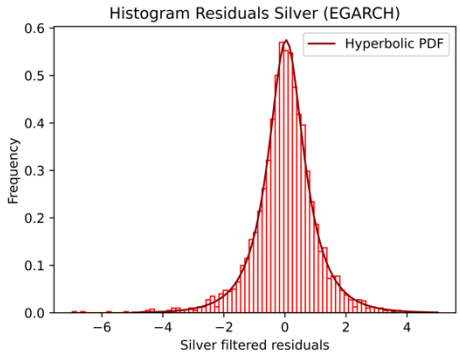


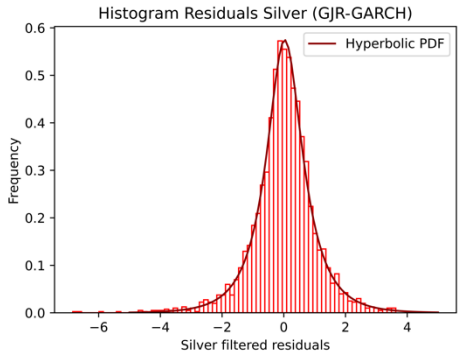


And finally, for the silver index residuals, the maximum likelihood maximization leads to the following estimates:

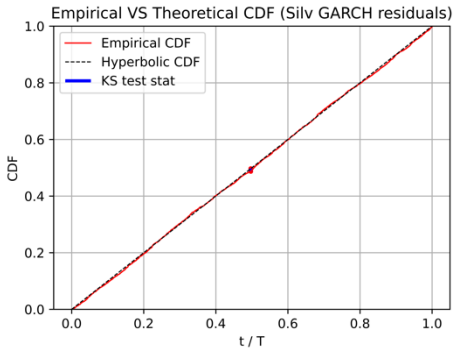
	GARCH	*GJR-GARCH*	*EGARCH*
Estimation λ	-0.00546	-0.0466	-0.0425
Estimation α	1.028	1.016	1.019
Estimation β	0.01	0.01	0.01
Estimation δ	0.635	0.6557	0.6546
Estimation μ	0.0348	0.0274	0.0374

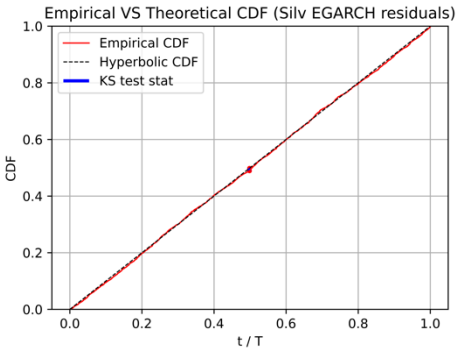


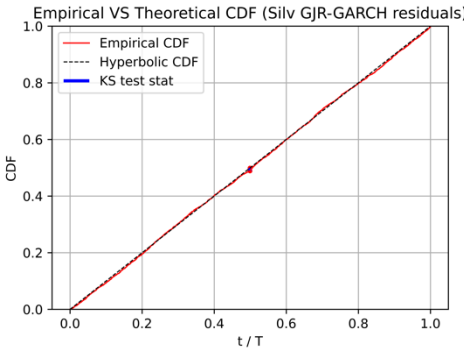




Estimating this distribution with our parameters from the Maximum Likelihood optimization leads to a fit with the residuals. We run the KS test and we obtain for our test-statistics the values 0.00988, 0.00938 and 0.011 for the GARCH, EGARCH and GJR-GARCH residuals respectively. One more time, the hyperbolic distribution fits perfectly our residuals, the null hypothesis is not rejected. The hyperbolic distribution is a very good way to model the residuals from the GARCH model implying the Silver Index.

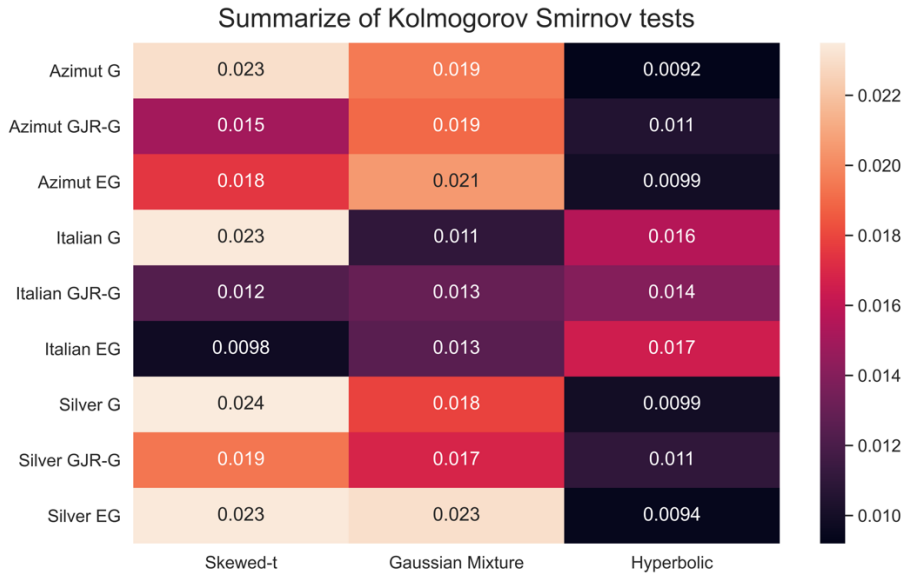






Previously, we just analyzed three different distributions in order to fit them with the residuals data from our GARCH models. We just saw that they all perfectly fit the observations. However, we have to check which one is the best to use for each time series. To do so, we will compare the different Kolmogorov Smirnov test-statistic and will check which one is the minimum value. The latter will give us the conclusion that the corresponding distribution will be the best for the time series.

At first, we can summaries the KS test-statistics previously obtained in the following table:



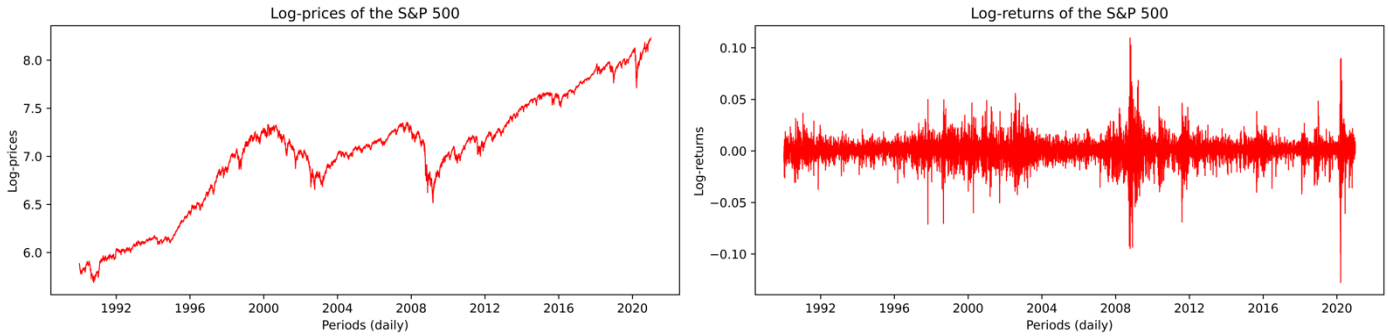
The smallest KS test-statistics are represented with the darkest colors in the table such as black for instance. The highest values are in this case represented by the most luminous colors. We can see then that the Hyperbolic distribution is the most adapted to model the statistical properties of the residuals for Azimut and the Silver Index. However, for the Italian Stock Index residuals, it is a different case. Both the skewed-t and the Gaussian Mixture present very small KS tests values compared to the Hyperbolic. Then, using these two distributions could be equivalent since they lead to very similar results. Besides, we saw previously that asymmetry in the density function is observed for the Italian Stock Index residuals so using the skewed-t distribution can be more adapted since it captures the most the asymmetric properties compared to the other distributions.

- *Conclusion GARCH models:*

We previously tried to fit several non-gaussian distributions to our standardized residuals from the estimated GARCH models and its respective extensions (GJR and EGARCH models). In the case where these residuals don't follow a normal distribution, using a gaussian distribution would lead to wrong estimations and predictions concerning conditional volatility. Most of the studies showed that normality is largely rejected, and this is also the case in our own study. Then, finding the best fitting distribution is the correct way to do if we want to perform precise forecasts of the volatility process. It would be interesting in the future to find other non-gaussian distribution for our residuals to study their fitting properties. Besides, performing volatility forecasts could be a very interesting method to do in further analysis.

II. Part 2: Empirical Option Pricing model

From now, we will use the daily price return of the S&P 500 to compute and estimate implied volatility with Option prices, for the period going from the 31st of December 1989 to 31st December 2020. The characteristics of the index, including the price and log-return evolution, is given below:



	Mean $\hat{\mu}_i^{Ann}$	Volatility $\hat{\sigma}_i^{Ann}$	Skewness $\hat{S}_i$	Kurtosis $\hat{K}_i$	Minimum	Maximum
S&P log-returns	0.0757	0.1825	-0.41	14.37	-0.1277	0.1096

- *Heston and Nandi model:*

This model describes the evolution of the volatility of an underlying asset such as a stock. It assumes that volatility is stochastic, meaning that follows a random process. It is so not constant, nor deterministic over time. The model can be described as the following:

$$\log S_t = \log S_{t-1} + r + \lambda h_t + \sqrt{h_t}\,\varepsilon_t \quad ; \quad \varepsilon_t \sim N(0,1)$$

$$R_t = \log S_t - \log S_{t-1} = r + \lambda h_t + \sqrt{h_t}\,\varepsilon_t$$

S_t	Underlying asset price at time t.
R_t	Log-return of the asset price at time t.
r	Daily continuously compounded risk-free rate.
ε_t	Innovations that follow standard normal random variable.
h_t	Conditional variance of the log-returns between $t-1$ and t.

With λh_t as the risk premium. The dynamics of the conditional variance can be represented as a GARCH (1,1) model:

$$h_t = \omega + \beta h_{t-1} + \alpha\left(\varepsilon_{t-1} - \gamma\sqrt{h_{t-1}}\right)^2$$

To perform an estimation of the parameters [$\lambda\ \omega\ \beta\ \alpha\ \gamma$], we are going to perform a Maximum Likelihood estimation using the time series of S&P 500 historical daily index returns (7811 daily observations). Before doing that, we apply the change in variable theorem explained as follow:

$$R_t = r + \lambda h_t + \sqrt{h_t}\,\varepsilon_t \Leftrightarrow \frac{R_t - r - \lambda h_t}{\sqrt{h_t}} = \varepsilon_t$$

Which implies that the Likelihood function with a standard normal pdf is given by (1), and the log-likelihood by (2):

$$(1): \mathcal{L} = \prod_1^T \frac{1}{\sqrt{2\pi h_t}} \exp\left[-\frac{(R_t - r - \lambda h_t)^2}{2h_t}\right] \quad ; \quad (2): \log\mathcal{L} = \sum_1^T -\frac{1}{2}\left(\log(2\pi h_t) + \frac{(R_t - r - \lambda h_t)^2}{h_t}\right)$$

Before doing our MLE, we need a starting value for the variance h_t, we used the Esperance of it as starting value:

$$h_t = \omega + \beta h_{t-1} + \alpha\left(\varepsilon_{t-1} - \gamma\sqrt{h_{t-1}}\right)^2$$

$$h_t = \omega + \beta h_{t-1} + \alpha\, {\varepsilon_{t-1}}^2 - \alpha\gamma\varepsilon_{t-1}\sqrt{h_{t-1}} + \alpha\gamma^2 h_{t-1}$$

$$E[h_t] = \omega + \beta E[h_{t-1}] + \alpha E[\,{\varepsilon_{t-1}}^2] - \alpha\gamma E[\,\varepsilon_{t-1}]E\left[\sqrt{h_{t-1}}\right] + \alpha\gamma^2 E[h_{t-1}]$$

Knowing that $E[\,{\varepsilon_{t-1}}^2] = 1$ and $E[\,\varepsilon_{t-1}] = 0$, and by assuming stationarity with $E[h_t] = E[h_{t-1}] = E[h]$, we would obtain:

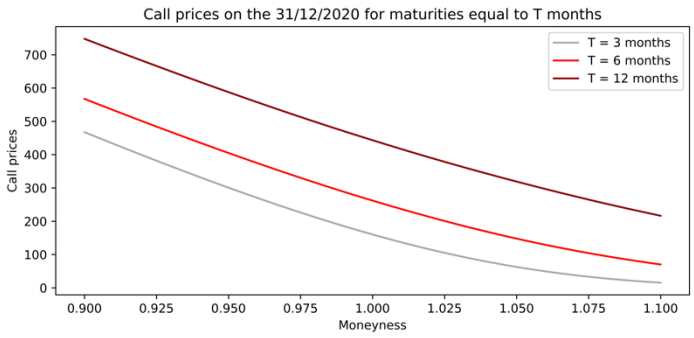
$$E[h] = \omega + \beta E[h] + \alpha + \alpha\gamma^2 E[h]] = E[h](1 - \beta - \alpha\gamma^2) = \omega + \alpha \Leftrightarrow E[h] = \frac{(\omega + \alpha)}{1 - \beta - \alpha\gamma^2}$$

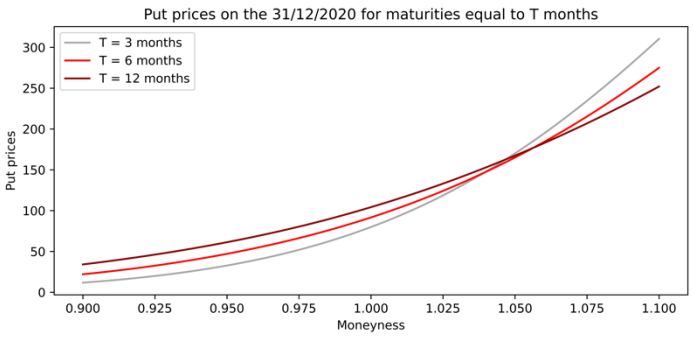
	Estimate	*Std Error*	*t-stats*
Coefficient λ	2.774	1.0026	2.77
Coefficient ω	-0.000001	0	-5.72
Coefficient β	0.836	0.0077	108.54
Coefficient α	0.000005	0	22.25
Coefficient γ	158.34	7.189	22.024

The null hypothesis for each of our parameters is H_0 : Parameter = 0. We can reject this null hypothesis whenever the absolute value of our T-stat is above a certain critical value. All our T-stats for all our parameters are quite big (higher than 2.56 that corresponds to the 1% significance level) thus we can strongly reject the null hypothesis for all of them and say they are all very significant.

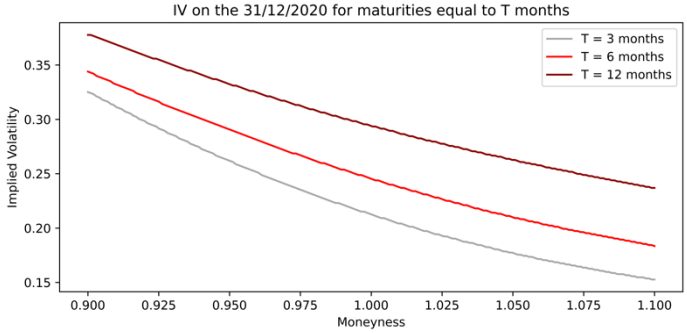
- *Option Prices simulation:*

After estimating these parameters, we will use them to simulate options prices using the Heston-Nandi process. We want to simulate the variable R_t. We can deduce with this the value of the underlying asset price $S_t = S_0 e^{R_t}$ with the initial index price at time 0 (on the 31st of December 2020). We used the Monte-Carlo simulation 10000 times with a risk-free rate equal to 0.25%. We simulated for maturities equal to 3, 6 and 12 months, the moneyness parameter is set between 0.9 and 1.1. We obtained the following results:





From our simulated option prices, we were able to recover the implied volatilities for each term and strike prices:



We observe that for a given time to maturity, implied volatilities are a convex function of the moneyness and bigger for a longer maturity. As the Implied Volatility (IV) is a forward-looking measure, a longer maturity let more time to the asset to potentially vary and thus to potentially have a bigger implied volatility.

Let S_0 being the price of the S&P500 on the 31st of December 2020, with a strike price $K = 0.9S_0$. The Call payoff is defined as:

$$C_T = \max(0,\ S_T - K) = \max(0,\ S_T - 0.9 * S_0)$$

$$\Leftrightarrow C_T = \max(0,\ S_0 e^{R_T} - 0.9S_0) = \max(0, e^{R_T} - 0.9)$$

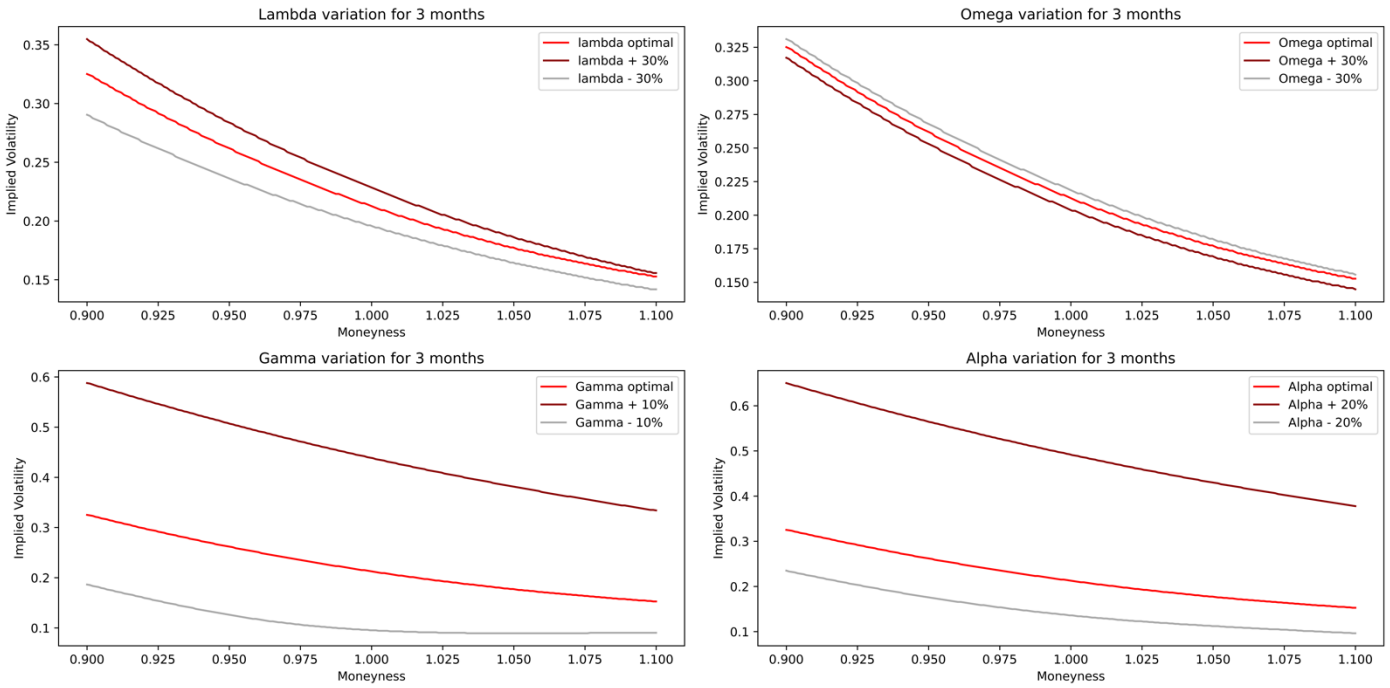
If R_T is bigger than $\ln(0.9)$ then the call option will be exercised. Therefore, if $R_T \in [-0.105, \infty]$ the call option will be exercised, which we can infer that on the 31st of December 2020 the option is already in the money. If the market is ready to call options which are already in the money, it might be that the market suspect that the underlying is going to have big variations, which is reflected in the implied volatility.

- *Implied Volatility variations:*

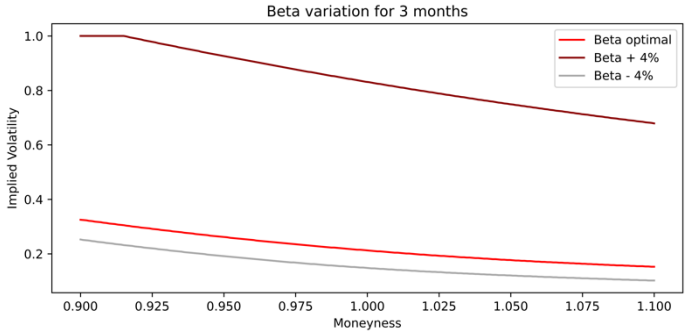
We will now look at how implied volatility is affected when our parameters from the Heston-Nandi process changes. To do so we made the parameters vary one by one by x % to the optimal value found by the maximum likelihood. First let's look at the variation of omega, we clearly see that for big variation the implied volatility doesn't change much. The omega coefficient is constant in the GARCH (1,1) variance, the value is a relatively small and variation in it seems to not affect that much the implied volatility. λh_t is the risk premium, therefore a variation of lambda impacts the risk premium. We can observe on the second figure that if lambda is reduced, which imply that the risk premia is reduced, the implied volatility decreases. Mechanically a decrease of lambda, decrease the log-return which imply that the price variation is weaker, and this is reflected in the implied volatility. If an asset provides less risk premia, investors might be less attracted by this asset and therefore trade it less, which imply that the asset suffer from less variation. To look at alpha and gamma variation, lets first rearrange the dynamic of the variance:

$$h_t = \omega + \beta h_{t-1} + \alpha\left(\varepsilon_{t-1} - \gamma\sqrt{h_{t-1}}\right)^2 \Leftrightarrow h_t = \omega + \beta h_{t-1} + \alpha\,{\varepsilon_{t-1}}^2 - \alpha\gamma\varepsilon_{t-1}\sqrt{h_{t-1}} + \alpha\gamma^2 h_{t-1}$$

We see that if alpha or gamma increases, it will mechanically increase the variance. These increases will thereafter affect the log-returns, which will finally be reflected in a higher implied volatility as prices variations are bigger.

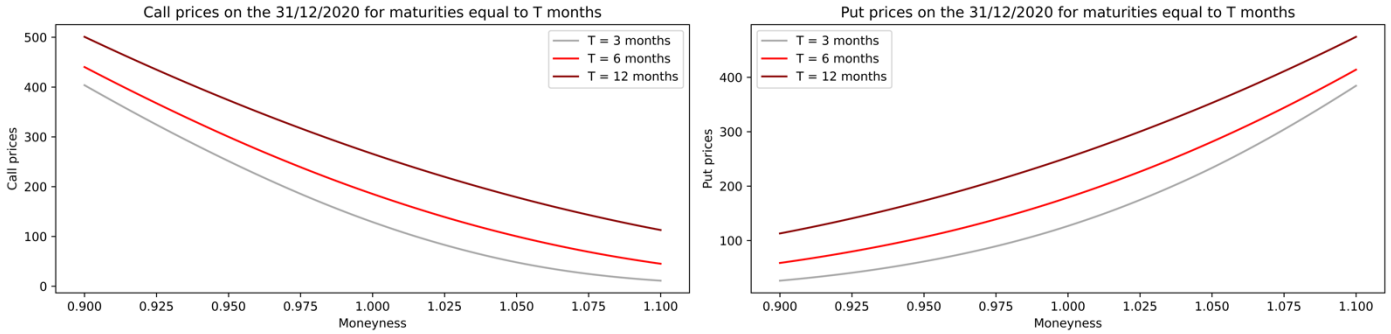


Our last parameter to look at is β, we found that our optimal beta equal to 0.84. This imply that the conditional variance on time t depend by 84% of his value on time t-1. We were able to only vary beta by small percentages, because bigger variations were making the process explosive and therefore the simulation was not possible. We can observe on the figure that only with a small increase of beta the implied volatility increases a lot. A higher beta imply that conditional variance depends more on his past value, de facto this will increase the variance and we arrive at the same conclusions than for gamma and alpha, that higher prices variations will be reflected with a higher implied volatility.

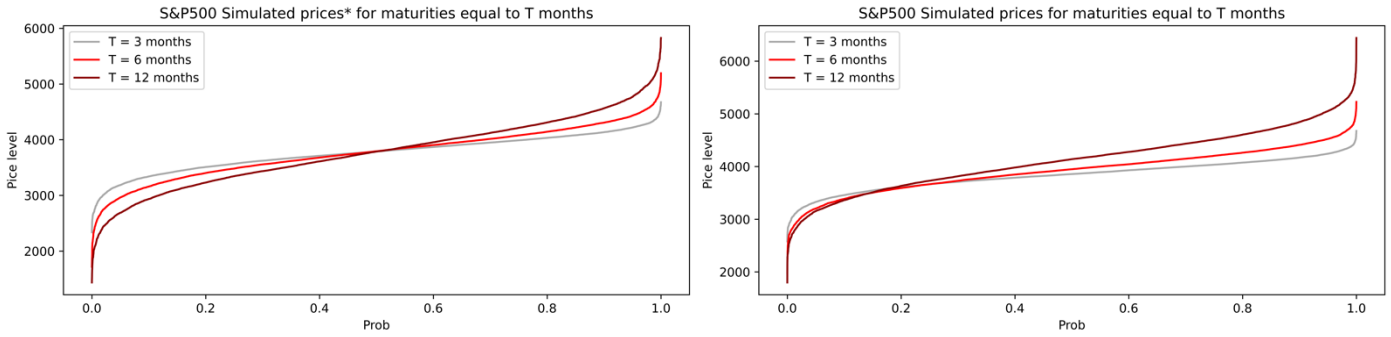


Finally, we look at options prices under risk neutral distribution with a Heston-Nandi process as follows:

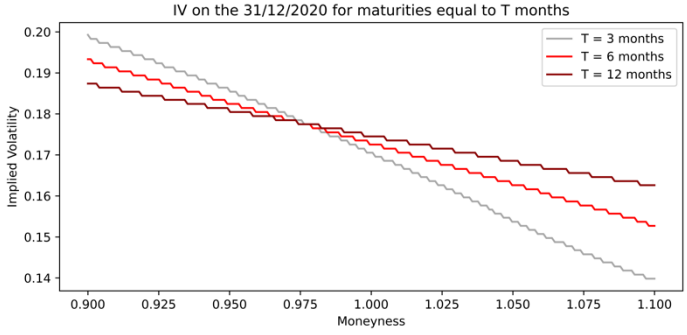
$$R_t = r - 0.5h_t + \sqrt{h_t}\,\varepsilon_t \quad ; \quad h_t = \omega + \beta h_{t-1} + \alpha\left(\varepsilon_{t-1} - \gamma^*\sqrt{h_{t-1}}\right)^2 \quad ; \quad \gamma^* = \gamma + \lambda + 0.5$$



We observe that under risk neutral distributions the simulated prices for the call options are smaller than our first simulation and that put prices are bigger. This due to the Simulated prices distribution, on left we have the simulated prices under risk neutral distribution compared to the simulated prices. We see that on left we have symmetric distribution, 50 % of the prices generated were higher than $S_0 = 3756$ and 50% smaller. In the first simulation, nearly 80 % of prices generated were bigger than S_0. This explains why the changes in Call prices and Put prices.



Finally, we see that implied volatility is much smaller in the risk neutral framework, compared to what we found previously. This is due to the distribution of simulated prices. In the risk neutral world, the Heston-Nandi process predict higher and lower prices compared to S_o with equal probability. On the non-risk neutral framework, the Heston-Nandi process predict more often higher prices than the price on 31 December 2020, which is coherent as the risk premia is much higher. We can conclude that implied volatility and risk premium are positively correlated.



III. Conclusion:

The main subject that we covered was essentially focused on volatility and its empirical estimation with Option pricing. At first, we estimated the conditional volatility of several time series with GARCH models, and we tried to find the best fitting distribution that could be used to model volatility and perform forecasts. Then, these concepts were used to be implemented in the Option pricing model of Heston and Nandi. Very satisfying results were obtained with highly significant estimates. It would be very interesting to focus on other models to estimate the price of options and the different parameters such as the Implied volatility.

IV. Appendix:

The code that we used in this assignment is given below:

- *Part 1:*

```
@author : The Quants Managers : Alessandro Loury, Endrit Kastrati, Sinan Guven and Gregory Porchet
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
from scipy.stats import skew
from scipy.stats import kurtosis
from scipy.stats import norm
import seaborn as sns
from numpy import size, log, pi, sum, array, zeros, diag, mat, asarray, sqrt, copy
from scipy.optimize import fmin_slsqp
from arch import arch_model
from scipy.stats.distributions import chi2
from scipy.optimize import minimize
import scipy.integrate as integrate
from scipy.stats import t
from scipy.special import kv



#==================================================
#Downloading the data
#==================================================



Azimut_Price = yf.download(tickers = 'AZM.MI', start = '2010-01-01', end = '2021-05-22')
Azimut_Price = Azimut_Price.iloc[:, 4]

Italian_Index = yf.download(tickers = 'FTSEMIB.MI', start = '2010-01-01', end = '2021-05-22')
Italian_Index = Italian_Index.iloc[:, 4]

Commodity_Index = yf.download(tickers = 'SI=F', start = '2010-01-01', end = '2021-05-22')
Commodity_Index = Commodity_Index.iloc[1:, 4]



#======================================================
#Computation of the Log Returns
#======================================================



Azimut_Price_log = np.log(Azimut_Price)
Azimut_Price_log = np.array(Azimut_Price_log)

Italian_Index_log = np.log(Italian_Index)
Italian_Index_log = np.array(Italian_Index_log)

Commodity_Index_log = np.log(Commodity_Index)
Commodity_Index_log = np.array(Commodity_Index_log)


Log_Returns_Azimut = np.subtract(Azimut_Price_log[1:len(Azimut_Price)], Azimut_Price_log[0:(len(Azimut_Price) - 1)])

Log_Returns_Index = np.subtract(Italian_Index_log[1:len(Italian_Index)], Italian_Index_log[0:(len(Italian_Index) - 1)])

Log_Returns_Silver = np.subtract(Commodity_Index_log[1:len(Commodity_Index)], Commodity_Index_log[0:(len(Commodity_Index) - 1)])



#================================================
#Graphical representation
#================================================



plt.figure(dpi = 1000, figsize=(18, 8))
plt.subplot(231)
```

```
plt.plot(Azimut_Price, color = 'red', linewidth = 1)
plt.ylabel('Price')
plt.xlabel('Periods (Daily)')
plt.title('Azimut Price')

plt.subplot(232)
plt.plot(Italian_Index, color = 'red', linewidth = 1)
plt.ylabel('Price')
plt.xlabel('Periods (Daily)')
plt.title('Italian Index')

plt.subplot(233)
plt.plot(Commodity Index, color = 'red', linewidth = 1)
plt.ylabel('Price')
plt.xlabel('Periods (Daily)')
plt.title('Silver Index')

plt.subplot(234)
plt.plot(Azimut_Price.index[1:], Log_Returns_Azimut, color = 'red', linewidth = 0.8)
plt.ylabel('Returns')
plt.xlabel('Periods (Daily)')
plt.title('Azimut Log-Returns')

plt.subplot(235)
plt.plot(Italian_Index.index[1:], Log_Returns_Index, color = 'red', linewidth = 0.8)
plt.ylabel('Returns')
plt.xlabel('Periods (Daily)')
plt.title('Italian Index Log-Returns')

plt.subplot(236)
plt.plot(Commodity_Index.index[1:], Log_Returns_Silver, color = 'red', linewidth = 0.8)
plt.ylabel('Returns')
plt.xlabel('Periods (Daily)')
plt.title('Silver Log-Returns')
plt.tight_layout()



#=====================================
#Statistical properties
#=====================================



def Stats(Data):
    Mean = np.mean(Data, 0) * 252
    Std = np.std(Data, 0) * np.power(252, 0.5)
    Skewness = skew(Data)
    Kurtosis = kurtosis(Data, fisher = False)
    Min = min(Data)
    Max = max(Data)
    print('Mean is:', Mean)
    print('Std is:', Std)
    print('Skewness is:',  Skewness)
    print('Kurtosis is:', Kurtosis)
    print('Min is:', Min)
    print('Max is:', Max)


Stats(Log_Returns_Azimut)
Stats(Log_Returns_Index)
Stats(Log_Returns_Silver)



#=================================================================
#Density function
#=================================================================



plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
sns.distplot(Log_Returns_Azimut, hist=False, kde=True, bins=int(180/5), color = 'red', kde_kws={'linewidth': 1})
x = np.linspace(np.mean(Log_Returns_Azimut) - 3*np.std(Log_Returns_Azimut), np.mean(Log_Returns_Azimut) +
3*np.std(Log_Returns_Azimut), 100)
plt.plot(x, norm.pdf(x, np.mean(Log_Returns_Azimut), np.std(Log_Returns_Azimut)), color = 'black')
plt.legend(['PDF Returns', 'PDF Normal'])
plt.title('Density Functions Azimut')
plt.xlabel('Azimut Log-Returns')

plt.subplot(132)
sns.distplot(Log_Returns_Index, hist=False, kde=True, bins=int(180/5), color = 'red', kde_kws={'linewidth': 1})
x = np.linspace(np.mean(Log_Returns_Index) - 3*np.std(Log_Returns_Index), np.mean(Log_Returns_Index) + 3*np.std(Log_Returns_Index),
100)
plt.plot(x, norm.pdf(x, np.mean(Log_Returns_Index), np.std(Log_Returns_Index)), color = 'black')
plt.legend(['PDF Returns', 'PDF Normal'])
plt.title('Density Functions Index')
plt.xlabel('Italian Index Log-Returns')

plt.subplot(133)
sns.distplot(Log_Returns_Silver, hist=False, kde=True, bins=int(180/5), color = 'red', kde_kws={'linewidth': 1})
x = np.linspace(np.mean(Log_Returns_Silver) - 3*np.std(Log_Returns_Silver), np.mean(Log_Returns_Silver) +
3*np.std(Log_Returns_Silver), 100)
plt.plot(x, norm.pdf(x, np.mean(Log_Returns_Silver), np.std(Log_Returns_Silver)), color = 'black')
plt.legend(['PDF Returns', 'PDF Normal'])
plt.title('Density Functions Silver')
plt.xlabel('Silver Log-Returns')
plt.tight_layout()
plt.show()
```

```
#================================================
#GARCH Model Implementation
#================================================



#-----------
#GARCH Model
#-----------



#Azimut
GM_Az = arch_model(Log_Returns_Azimut, p = 1, q = 1, mean = 'constant', vol = 'GARCH', dist = 'normal', rescale = True)
GM_result = GM_Az.fit(update_freq = 4)
print(GM_result.summary())
standardized_residuals = GM_result.resid / GM_result.conditional_volatility

plt.figure(dpi = 1000, figsize=(16, 4))
plt.subplot(121)
plt.plot(Azimut_Price.index[1:], Log_Returns_Azimut, linewidth = 1, color = 'darkred', alpha = 0.3)
plt.plot(Azimut_Price.index[1:], 0.01 * -GM_result.conditional_volatility, linewidth = 0.9, color = 'red')
plt.plot(Azimut_Price.index[1:], 0.01 * GM_result.conditional_volatility, linewidth = 0.9, color = 'red')
plt.xlabel('Periods (daily)')
plt.ylabel('Returns & Volatility')
plt.title('Conditional Volatility Estimation for Azimut')
plt.legend(['Log-Returns', 'Cond Volatility'])

plt.subplot(122)
plt.plot(Azimut_Price.index[1:], standardized_residuals, linewidth = 0.8, color = 'red')
plt.xlabel('Periods (daily)')
plt.ylabel('Residuals')
plt.title('Azimut Volatility Standardized Residuals')
plt.tight_layout()



#Index
GM_It = arch_model(Log_Returns_Index, p = 1, q = 1, mean = 'constant', vol = 'GARCH', dist = 'normal', rescale = True)
GM_result_It = GM_It.fit(update_freq = 4)
print(GM_result_It.summary())
standardized_residuals2 = GM_result_It.resid / GM_result_It.conditional_volatility

plt.figure(dpi = 1000, figsize=(16, 4))
plt.subplot(121)
plt.plot(Italian_Index.index[1:], Log_Returns_Index, linewidth = 1, color = 'darkred', alpha = 0.3)
plt.plot(Italian_Index.index[1:], 0.01 * -GM_result_It.conditional_volatility, linewidth = 0.9, color = 'red')
plt.plot(Italian_Index.index[1:], 0.01 * GM_result_It.conditional_volatility, linewidth = 0.9, color = 'red')
plt.xlabel('Periods (daily)')
plt.ylabel('Returns & Volatility')
plt.title('Conditional Volatility Estimation for the Italian Index')
plt.legend(['Log-Returns', 'Cond Volatility'])

plt.subplot(122)
plt.plot(Italian_Index.index[1:], standardized_residuals2, linewidth = 0.8, color = 'red')
plt.xlabel('Periods (daily)')
plt.ylabel('Residuals')
plt.title('Index Volatility Standardized Residuals')
plt.tight_layout()



#Silver
GM_Silver = arch_model(Log_Returns_Silver, p = 1, q = 1, mean = 'constant', vol = 'GARCH', dist = 'normal', rescale = True)
GM_result_Silver = GM_Silver.fit(update_freq = 4)
print(GM_result_Silver.summary())
standardized_residuals3 = GM_result_Silver.resid / GM_result_Silver.conditional_volatility

plt.figure(dpi = 1000, figsize=(16, 4))
plt.subplot(121)
plt.plot(Commodity_Index.index[1:], Log_Returns_Silver, linewidth = 1, color = 'darkred', alpha = 0.3)
plt.plot(Commodity_Index.index[1:], 0.01 * -GM_result_Silver.conditional_volatility, linewidth = 0.9, color = 'red')
plt.plot(Commodity_Index.index[1:], 0.01 * GM_result_Silver.conditional_volatility, linewidth = 0.9, color = 'red')
plt.xlabel('Periods (daily)')
plt.ylabel('Returns & Volatility')
plt.title('Conditional Volatility Estimation for Silver')
plt.legend(['Log-Returns', 'Cond Volatility'])

plt.subplot(122)
plt.plot(Commodity_Index.index[1:], standardized_residuals3, linewidth = 0.8, color = 'red')
plt.xlabel('Periods (daily)')
plt.ylabel('Residuals')
plt.title('Silver Volatility Standardized Residuals')
plt.tight_layout()



#---------------
#GJR-Garch Model
#---------------



#Azimut
Az = arch_model(Log_Returns_Azimut, p=1, o=1, q=1, mean = 'constant', vol = 'GARCH', dist = 'normal', rescale = True)
res = Az.fit(update_freq=4)
print(res.summary())
standardized_residuals4 = res.resid / res.conditional_volatility
```

```
plt.figure(dpi = 1000, figsize=(16, 4))
plt.subplot(121)
plt.plot(Azimut_Price.index[1:], Log_Returns_Azimut, linewidth = 1, color = 'darkred', alpha = 0.3)
plt.plot(Azimut_Price.index[1:], 0.01 * -res.conditional_volatility, linewidth = 0.9, color = 'red')
plt.plot(Azimut_Price.index[1:], 0.01 * res.conditional_volatility, linewidth = 0.9, color = 'red')
plt.xlabel('Periods (daily)')
plt.ylabel('Returns & Volatility')
plt.title('Conditional Volatility Estimation for Azimut (GJR-GARCH)')
plt.legend(['Log-Returns', 'Cond Volatility'])

plt.subplot(122)
plt.plot(Azimut_Price.index[1:], standardized_residuals4, linewidth = 0.8, color = 'red')
plt.xlabel('Periods (daily)')
plt.ylabel('Residuals')
plt.title('Azimut Volatility Standardized Residuals')
plt.tight_layout()


#Index
It = arch_model(Log_Returns_Index, p=1, o=1, q=1, mean = 'constant', vol = 'GARCH', dist = 'normal', rescale = True)
res_It = It.fit(update_freq=4, disp="off")
print(res_It.summary())
standardized_residuals5 = res_It.resid / res_It.conditional_volatility

plt.figure(dpi = 1000, figsize=(16, 4))
plt.subplot(121)
plt.plot(Italian_Index.index[1:], Log_Returns_Index, linewidth = 1, color = 'darkred', alpha = 0.3)
plt.plot(Italian_Index.index[1:], 0.01 * -res_It.conditional_volatility, linewidth = 0.9, color = 'red')
plt.plot(Italian_Index.index[1:], 0.01 * res_It.conditional_volatility, linewidth = 0.9, color = 'red')
plt.xlabel('Periods (daily)')
plt.ylabel('Returns & Volatility')
plt.title('Conditional Volatility Estimation for the Italian Index (GJR-GARCH)')
plt.legend(['Log-Returns', 'Cond Volatility'])

plt.subplot(122)
plt.plot(Italian Index.index[1:], standardized residuals5, linewidth = 0.8, color = 'red')
plt.xlabel('Periods (daily)')
plt.ylabel('Residuals')
plt.title('Index Volatility Standardized Residuals')
plt.tight_layout()


#Silver
Silv = arch_model(Log_Returns_Silver, p=1, o=1, q=1, mean = 'constant', vol = 'GARCH', dist = 'normal', rescale = True)
res_Silver = Silv.fit(update_freq=4)
print(res_Silver.summary())
standardized_residuals6 = res_Silver.resid / res_Silver.conditional_volatility

plt.figure(dpi = 1000, figsize=(16, 4))
plt.subplot(121)
plt.plot(Commodity_Index.index[1:], Log_Returns_Silver, linewidth = 1, color = 'darkred', alpha = 0.3)
plt.plot(Commodity_Index.index[1:], 0.01 * -res_Silver.conditional_volatility, linewidth = 0.9, color = 'red')
plt.plot(Commodity_Index.index[1:], 0.01 * res_Silver.conditional_volatility, linewidth = 0.9, color = 'red')
plt.xlabel('Periods (daily)')
plt.ylabel('Returns & Volatility')
plt.title('Conditional Volatility Estimation for Silver (GJR-GARCH)')
plt.legend(['Log-Returns', 'Cond Volatility'])

plt.subplot(122)
plt.plot(Commodity_Index.index[1:], standardized_residuals6, linewidth = 0.8, color = 'red')
plt.xlabel('Periods (daily)')
plt.ylabel('Residuals')
plt.title('Silver Volatility Standardized Residuals')
plt.tight layout()



#-------------
#E-Garch Model
#-------------



#Azimut
Az2 = arch_model(Log_Returns_Azimut, p=1, o=1, q=1, mean = 'constant', vol = 'EGARCH', dist = 'normal', rescale = True)
res2 = Az2.fit(update_freq=4)
print(res2.summary())
standardized_residuals7 = res2.resid / res2.conditional_volatility

plt.figure(dpi = 1000, figsize=(16, 4))
plt.subplot(121)
plt.plot(Azimut_Price.index[1:], Log_Returns_Azimut, linewidth = 1, color = 'darkred', alpha = 0.3)
plt.plot(Azimut_Price.index[1:], 0.01 * -res2.conditional_volatility, linewidth = 0.9, color = 'red')
plt.plot(Azimut_Price.index[1:], 0.01 * res2.conditional_volatility, linewidth = 0.9, color = 'red')
plt.xlabel('Periods (daily)')
plt.ylabel('Returns & Volatility')
plt.title('Conditional Volatility Estimation for Azimut (E-GARCH)')
plt.legend(['Log-Returns', 'Cond Volatility'])

plt.subplot(122)
plt.plot(Azimut_Price.index[1:], standardized_residuals7, linewidth = 0.8, color = 'red')
plt.xlabel('Periods (daily)')
plt.ylabel('Residuals')
plt.title('Azimut Volatility Standardized Residuals')
plt.tight_layout()


#Italian Index
It2 = arch_model(Log_Returns_Index, p=1, o=1, q=1, mean = 'constant', vol = 'EGARCH', dist = 'normal', rescale = True)
res_It2 = It2.fit(update_freq=4)
print(res_It2.summary())
```

```
standardized_residuals8 = res_It2.resid / res_It2.conditional_volatility

plt.figure(dpi = 1000, figsize=(16, 4))
plt.subplot(121)
plt.plot(Italian_Index.index[1:], Log_Returns_Index, linewidth = 1, color = 'darkred', alpha = 0.3)
plt.plot(Italian_Index.index[1:], 0.01 * -res_It2.conditional_volatility, linewidth = 0.9, color = 'red')
plt.plot(Italian_Index.index[1:], 0.01 * res_It2.conditional_volatility, linewidth = 0.9, color = 'red')
plt.xlabel('Periods (daily)')
plt.ylabel('Returns & Volatility')
plt.title('Conditional Volatility Estimation for the Italian Index (E-GARCH)')
plt.legend(['Log-Returns', 'Cond Volatility'])

plt.subplot(122)
plt.plot(Italian_Index.index[1:], standardized_residuals8, linewidth = 0.8, color = 'red')
plt.xlabel('Periods (daily)')
plt.ylabel('Residuals')
plt.title('Index Volatility Standardized Residuals')
plt.tight_layout()
plt.plot(res_It2.resid)


#Silver Index
Silv2 = arch_model(Log_Returns_Silver, p=1, o=1, q=1, mean = 'constant', vol = 'EGARCH', dist = 'normal', rescale = True)
res_Silver2 = Silv2.fit(update_freq=4)
print(res_Silver2.summary())
standardized_residuals9 = res_Silver2.resid / res_Silver2.conditional_volatility

plt.figure(dpi = 1000, figsize=(16, 4))
plt.subplot(121)
plt.plot(Commodity_Index.index[1:], Log_Returns_Silver, linewidth = 1, color = 'darkred', alpha = 0.3)
plt.plot(Commodity_Index.index[1:], 0.01 * -res_Silver2.conditional_volatility, linewidth = 0.9, color = 'red')
plt.plot(Commodity_Index.index[1:], 0.01 * res_Silver2.conditional_volatility, linewidth = 0.9, color = 'red')
plt.xlabel('Periods (daily)')
plt.ylabel('Returns & Volatility')
plt.title('Conditional Volatility Estimation for Silver (E-GARCH)')
plt.legend(['Log-Returns', 'Cond Volatility'])

plt.subplot(122)
plt.plot(Commodity_Index.index[1:], standardized_residuals9, linewidth = 0.8, color = 'red')
plt.xlabel('Periods (daily)')
plt.ylabel('Residuals')
plt.title('Silver Volatility Standardized Residuals')
plt.tight_layout()



#=========================================================
#Likelihood Ratio Test Azimut
#=========================================================



def garch_likelihood2(parameters, data, sigma2):
    mu = parameters[0]
    eps = data - mu
    logliks = -0.5 * (np.log(2 * pi) + np.log(sigma2) + np.divide(np.power(eps, 2), sigma2))
    #loglik = sum(logliks)
    return logliks#, loglik


#Log-likelihood
AZ_GARCH_loglike = GM_result.loglikelihood  #Constrained model (since we have gamma = 0 so we have 1 constraint)
AZ_GJRGARCH_loglike = res.loglikelihood     #Unconstrained model
AZ_EGARCH_loglike = res2.loglikelihood      #Unconstrained model 2


#For nested models (GARCH and GJR-GARCH)
LR_Test_AZ1 = 2 * (AZ_GJRGARCH_loglike - AZ_GARCH_loglike)
p_value = chi2.sf(LR_Test_AZ1, 3)


#Vuong test for non-nested models (first GARCH and EGARCH)
All_z_i_Garch = garch_likelihood2(GM_result.params, GM_result.scale * Log_Returns_Azimut, np.power(GM_result.conditional_volatility,
2))

All_z_i_EGarch = garch_likelihood2(res2.params, res2.scale * Log_Returns_Azimut, np.power(res2.conditional_volatility, 2))

LR_Test_AZ2 = np.sum(np.subtract(All_z_i_EGarch, All_z_i_Garch))
Test_Vuong_AZ = np.divide(LR_Test_AZ2, np.power(len(Log_Returns_Azimut), 0.5) * np.std(np.subtract(All_z_i_EGarch, All_z_i_Garch)))


#For EGARCH and GJR-GARCH
All_z_i_GJRGarch = garch_likelihood2(res.params, res.scale * Log_Returns_Azimut, np.power(res.conditional_volatility, 2))

LR_Test_AZ3 = np.sum(np.subtract(All_z_i_EGarch, All_z_i_GJRGarch))
Test_Vuong_AZ2 = np.divide(LR_Test_AZ3, np.power(len(Log_Returns_Azimut), 0.5) * np.std(np.subtract(All_z_i_EGarch,
All_z_i_GJRGarch)))



#=========================================================
#Likelihood Ratio Test Italian index
#=========================================================



It_GARCH_loglike = GM_result_It.loglikelihood
It_GJRGARCH_loglike = res_It.loglikelihood
It_EGARCH_loglike = res_It2.loglikelihood


#For nested models (GARCH and GJR-GARCH)
LR_Test_It1 = 2 * (It_GJRGARCH_loglike - It_GARCH_loglike)
```

```
p_value1It = chi2.sf(LR_Test_It1, 3)


#Vuong test for non-nested models (first GARCH and EGARCH)
All_z_i_GarchIt = garch_likelihood2(GM_result_It.params, GM_result_It.scale * Log_Returns_Index,
np.power(GM_result_It.conditional_volatility, 2))


All_z_i_EGarchIt = garch_likelihood2(res_It2.params, res_It2.scale * Log_Returns_Index, np.power(res_It2.conditional_volatility, 2))

LR_Test_It2 = np.sum(np.subtract(All_z_i_EGarchIt, All_z_i_GarchIt))
Test_Vuong_It = np.divide(LR_Test_It2, np.power(len(Log_Returns_Index), 0.5) * np.std(np.subtract(All_z_i_EGarchIt, All_z_i_GarchIt)))


#For EGARCH and GJR-GARCH
All z i GJRGarchIt = garch likelihood2(res It.params, res It.scale * Log Returns Index, np.power(res It.conditional volatility, 2))

LR_Test_It3 = np.sum(np.subtract(All_z_i_EGarchIt, All_z_i_GJRGarchIt))
Test_Vuong_It2 = np.divide(LR_Test_It3, np.power(len(Log_Returns_Index), 0.5) * np.std(np.subtract(All_z_i_EGarchIt,
All_z_i_GJRGarchIt)))



#=======================================================
#Likelihood Ratio Test Silver Index
#=======================================================



Silv_GARCH_loglike = GM_result_Silver.loglikelihood
Silv_GJRGARCH_loglike = res_Silver.loglikelihood
Silv_EGARCH_loglike = res_Silver2.loglikelihood


#For nested models (GARCH and GJR-GARCH)
LR_Test_Silv1 = 2 * (Silv_GJRGARCH_loglike - Silv_GARCH_loglike)
p_value1Silv = chi2.sf(LR_Test_Silv1, 3)


#Vuong test for non-nested models (first GARCH and EGARCH)
All z i GarchSilv = garch likelihood2(GM result Silver.params, GM result Silver.scale * Log Returns Silver,
np.power(GM_result_Silver.conditional_volatility, 2))

All_z_i_EGarchSilv = garch_likelihood2(res_Silver2.params, res_Silver2.scale * Log_Returns_Silver,
np.power(res_Silver2.conditional_volatility, 2))

LR_Test_Silv2 = np.sum(np.subtract(All_z_i_EGarchSilv, All_z_i_GarchSilv))
Test_Vuong_Silv = np.divide(LR_Test_Silv2, np.power(len(Log_Returns_Silver), 0.5) * np.std(np.subtract(All_z_i_EGarchSilv,
All_z_i_GarchSilv)))


#For EGARCH and GJR-GARCH
All_z_i_GJRGarchSilv = garch_likelihood2(res_Silver.params, res_Silver.scale * Log_Returns_Silver,
np.power(res_Silver.conditional_volatility, 2))

LR_Test_Silv3 = np.sum(np.subtract(All_z_i_EGarchSilv, All_z_i_GJRGarchSilv))
Test_Vuong_Silv2 = np.divide(LR_Test_Silv3, np.power(len(Log_Returns_Silver), 0.5) * np.std(np.subtract(All_z_i_EGarchSilv,
All_z_i_GJRGarchSilv)))



#================================================================
#Test for Normality Residuals
#================================================================



#For Azimut
AZ_GARCH_resid = GM_result.resid
AZ_GARCH_resid_std = standardized_residuals

AZ_EGARCH_resid = res2.resid
AZ_EGARCH_resid_std = standardized_residuals7

AZ_GJRGARCH_resid = res.resid
AZ_GJRGARCH_resid_std = standardized_residuals4



plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.hist(AZ_GARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
x = np.linspace(np.mean(AZ_GARCH_resid_std) - 3*np.std(AZ_GARCH_resid_std), np.mean(AZ_GARCH_resid_std) +
3*np.std(AZ_GARCH_resid_std), 100)
plt.plot(x, norm.pdf(x, np.mean(AZ_GARCH_resid_std), np.std(AZ_GARCH_resid_std)), color = 'darkred', linewidth = 1.5)
plt.legend(['Normal PDF'])
plt.title('Histogram Residuals Azimut (GARCH)')
plt.xlabel('Azimut filtered residuals')
plt.ylabel('Frequency')

plt.subplot(132)
plt.hist(AZ_EGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
x = np.linspace(np.mean(AZ_EGARCH_resid_std) - 3*np.std(AZ_EGARCH_resid_std), np.mean(AZ_EGARCH_resid_std) +
3*np.std(AZ_EGARCH_resid_std), 100)
plt.plot(x, norm.pdf(x, np.mean(AZ_EGARCH_resid_std), np.std(AZ_EGARCH_resid_std)), color = 'darkred', linewidth = 1.5)
plt.legend(['Normal PDF'])
plt.title('Histogram Residuals Azimut (EGARCH)')
plt.xlabel('Azimut filtered residuals')
plt.ylabel('Frequency')

plt.subplot(133)
```

```
plt.hist(AZ_GJRGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
x = np.linspace(np.mean(AZ_GJRGARCH_resid_std) - 3*np.std(AZ_GJRGARCH_resid_std), np.mean(AZ_GJRGARCH_resid_std) +
3*np.std(AZ_GJRGARCH_resid_std), 100)
plt.plot(x, norm.pdf(x, np.mean(AZ_GJRGARCH_resid_std), np.std(AZ_GJRGARCH_resid_std)), color = 'darkred', linewidth = 1.5)
plt.legend(['Normal PDF'])
plt.title('Histogram Residuals Azimut (GJR-GARCH)')
plt.xlabel('Azimut filtered residuals')
plt.ylabel('Frequency')
plt.tight layout()
plt.show()


def Jarque_Bera(data):
    Skewness = skew(data)
    Kurtosis = kurtosis(data, fisher = False)
    test_JB = np.size(data, 0) * (np.power(Skewness, 2)/6 + np.power(Kurtosis - 3, 2)/24)
    return test_JB


Jarque_Bera(AZ_GARCH_resid_std)
Jarque_Bera(AZ_EGARCH_resid_std)
Jarque_Bera(AZ_GJRGARCH_resid_std)

Stats(AZ_GARCH_resid_std)
Stats(AZ_EGARCH_resid_std)
Stats(AZ_GJRGARCH_resid_std)



#For Italian Index
It_GARCH_resid = GM_result_It.resid
It_GARCH_resid_std = standardized_residuals2

It_EGARCH_resid = res_It2.resid
It_EGARCH_resid_std = standardized_residuals8

It GJRGARCH resid = res It.resid
It_GJRGARCH_resid_std = standardized_residuals5



plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.hist(It_GARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
x = np.linspace(np.mean(It_GARCH_resid_std) - 3*np.std(It_GARCH_resid_std), np.mean(It_GARCH_resid_std) +
3*np.std(It_GARCH_resid_std), 100)
plt.plot(x, norm.pdf(x, np.mean(It_GARCH_resid_std), np.std(It_GARCH_resid_std)), color = 'darkred', linewidth = 1.5)
plt.legend(['Normal PDF'])
plt.title('Histogram Residuals Index (GARCH)')
plt.xlabel('Index filtered residuals')
plt.ylabel('Frequency')

plt.subplot(132)
plt.hist(It_EGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
x = np.linspace(np.mean(It_EGARCH_resid_std) - 3*np.std(It_EGARCH_resid_std), np.mean(It_EGARCH_resid_std) +
3*np.std(It_EGARCH_resid_std), 100)
plt.plot(x, norm.pdf(x, np.mean(It_EGARCH_resid_std), np.std(It_EGARCH_resid_std)), color = 'darkred', linewidth = 1.5)
plt.legend(['Normal PDF'])
plt.title('Histogram Residuals Index (EGARCH)')
plt.xlabel('Index filtered residuals')
plt.ylabel('Frequency')

plt.subplot(133)
plt.hist(It_GJRGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
x = np.linspace(np.mean(It_GJRGARCH_resid_std) - 3*np.std(It_GJRGARCH_resid_std), np.mean(It_GJRGARCH_resid_std) +
3*np.std(It_GJRGARCH_resid_std), 100)
plt.plot(x, norm.pdf(x, np.mean(It_GJRGARCH_resid_std), np.std(It_GJRGARCH_resid_std)), color = 'darkred', linewidth = 1.5)
plt.legend(['Normal PDF'])
plt.title('Histogram Residuals Index (GJR-GARCH)')
plt.xlabel('Index filtered residuals')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()


Jarque_Bera(It_GARCH_resid_std)
Jarque_Bera(It_EGARCH_resid_std)
Jarque_Bera(It_GJRGARCH_resid_std)

Stats(It_GARCH_resid_std)
Stats(It_EGARCH_resid_std)
Stats(It_GJRGARCH_resid_std)



#For Silver Index
Silv GARCH resid = GM result Silver.resid
Silv_GARCH_resid_std = standardized_residuals3

Silv_EGARCH_resid = res_Silver2.resid
Silv_EGARCH_resid_std = standardized_residuals9

Silv_GJRGARCH_resid = res_Silver.resid
Silv_GJRGARCH_resid_std = standardized_residuals6


plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
```

```
plt.hist(Silv_GARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
x = np.linspace(np.mean(Silv_GARCH_resid_std) - 3*np.std(Silv_GARCH_resid_std), np.mean(Silv_GARCH_resid_std) +
3*np.std(Silv_GARCH_resid_std), 100)
plt.plot(x, norm.pdf(x, np.mean(Silv_GARCH_resid_std), np.std(Silv_GARCH_resid_std)), color = 'darkred', linewidth = 1.5)
plt.legend(['Normal PDF'])
plt.title('Histogram Residuals Silver (GARCH)')
plt.xlabel('Silver filtered residuals')
plt.ylabel('Frequency')

plt.subplot(132)
plt.hist(Silv_EGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
x = np.linspace(np.mean(Silv_EGARCH_resid_std) - 3*np.std(Silv_EGARCH_resid_std), np.mean(Silv_EGARCH_resid_std) +
3*np.std(Silv_EGARCH_resid_std), 100)
plt.plot(x, norm.pdf(x, np.mean(Silv EGARCH resid std), np.std(Silv EGARCH resid std)), color = 'darkred', linewidth = 1.5)
plt.legend(['Normal PDF'])
plt.title('Histogram Residuals Silver (EGARCH)')
plt.xlabel('Silver filtered residuals')
plt.ylabel('Frequency')

plt.subplot(133)
plt.hist(Silv_GJRGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
x = np.linspace(np.mean(Silv_GJRGARCH_resid_std) - 3*np.std(Silv_GJRGARCH_resid_std), np.mean(Silv_GJRGARCH_resid_std) +
3*np.std(Silv_GJRGARCH_resid_std), 100)
plt.plot(x, norm.pdf(x, np.mean(Silv_GJRGARCH_resid_std), np.std(Silv_GJRGARCH_resid_std)), color = 'darkred', linewidth = 1.5)
plt.legend(['Normal PDF'])
plt.title('Histogram Residuals Silver (GJR-GARCH)')
plt.xlabel('Silver filtered residuals')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()



Jarque_Bera(Silv_GARCH_resid_std)
Jarque_Bera(Silv_EGARCH_resid_std)
Jarque_Bera(Silv_GJRGARCH_resid_std)

Stats(Silv_GARCH_resid_std)
Stats(Silv_EGARCH_resid_std)
Stats(Silv_GJRGARCH_resid_std)



#=============================================================================================
#Empirical distribution to the residuals
#=============================================================================================



#Gaussian Mixture
def ML_MN(para,X, out = 1):
    phi=para[0]
    mu1=para[1]
    sigma1=para[2]
    mu2=para[3]
    sigma2=para[4]
    likelihood=phi*norm.pdf(X,mu1,sigma1)+(1-phi)*norm.pdf(X,mu2,sigma2)
    loglik=-sum(np.log(likelihood))
    CDF = phi*norm.cdf(X,mu1,sigma1)+(1-phi)*norm.cdf(X,mu2,sigma2)
    if out is None:
        return loglik
    if out == 1:
        return CDF
    else:
        return likelihood


bounds = [(0, 1), (-10.0, 10.0), (0.0,10.0), (-10.0,10.0), (0.0,10.0)]
startingVals = array([0.5, 0.5, 0.5, 0.5, 0.5])
support=np.arange(-5, 5, 0.01).tolist()



estimates = fmin_slsqp(ML_MN, startingVals, f_ieqcons=None, bounds = bounds, args = AZ_GARCH_resid_std)

f1=norm.pdf(support, estimates[1], estimates[2])
f2=norm.pdf(support, estimates[3], estimates[4])

PDF = ML_MN(estimates, support)
plt.figure(dpi = 1000)
plt.plot(support, f1,'-b', support, f2,'-r')
plt.plot(support, PDF, color = 'green')
plt.hist(AZ_GARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')



#==========
#For Azimut
#==========



Estimat = minimize(ML_MN, startingVals, method = 'SLSQP', bounds = bounds, args = AZ_GARCH_resid_std)
Parameters_AZ_GARCH_MN = Estimat.x

Estimat2 = minimize(ML_MN, startingVals, method = 'SLSQP', bounds = bounds, args = AZ_EGARCH_resid_std)
Parameters_AZ_EGARCH_MN = Estimat2.x

Estimat3 = minimize(ML_MN, startingVals, method = 'SLSQP', bounds = bounds, args = AZ_GJRGARCH_resid_std)
```

```
Parameters_AZ_GJRGARCH_MN = Estimat3.x


plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.hist(AZ_GARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(support, ML_MN(Parameters_AZ_GARCH_MN, support), color = 'darkred', linewidth = 1.4)
plt.legend(['Mixture PDF'])
plt.title('Histogram Residuals Azimut (GARCH)')
plt.xlabel('Azimut filtered residuals')
plt.ylabel('Frequency')

plt.subplot(132)
plt.hist(AZ_EGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(support, ML_MN(Parameters_AZ_EGARCH_MN, support), color = 'darkred', linewidth = 1.4)
plt.legend(['Mixture PDF'])
plt.title('Histogram Residuals Azimut (EGARCH)')
plt.xlabel('Azimut filtered residuals')
plt.ylabel('Frequency')

plt.subplot(133)
plt.hist(AZ_GJRGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(support, ML_MN(Parameters_AZ_GJRGARCH_MN, support), color = 'darkred', linewidth = 1.4)
plt.legend(['Mixture PDF'])
plt.title('Histogram Residuals Azimut (GJR-GARCH)')
plt.xlabel('Azimut filtered residuals')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()



#=================
#For Italian Index
#=================



Estimat4 = minimize(ML_MN, startingVals, method = 'SLSQP', bounds = bounds, args = It_GARCH_resid_std)
Parameters_It_GARCH_MN = Estimat4.x

Estimat5 = minimize(ML_MN, startingVals, method = 'SLSQP', bounds = bounds, args = It_EGARCH_resid_std)
Parameters_It_EGARCH_MN = Estimat5.x

Estimat6 = minimize(ML_MN, startingVals, method = 'SLSQP', bounds = bounds, args = It_GJRGARCH_resid_std)
Parameters_It_GJRGARCH_MN = Estimat6.x


plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.hist(It_GARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(support, ML_MN(Parameters_It_GARCH_MN, support), color = 'darkred', linewidth = 1.4)
plt.legend(['Mixture PDF'])
plt.title('Histogram Residuals Index (GARCH)')
plt.xlabel('Index filtered residuals')
plt.ylabel('Frequency')

plt.subplot(132)
plt.hist(It EGARCH resid std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(support, ML_MN(Parameters_It_EGARCH_MN, support), color = 'darkred', linewidth = 1.4)
plt.legend(['Mixture PDF'])
plt.title('Histogram Residuals Index (EGARCH)')
plt.xlabel('Index filtered residuals')
plt.ylabel('Frequency')

plt.subplot(133)
plt.hist(It_GJRGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(support, ML_MN(Parameters_It_GJRGARCH_MN, support), color = 'darkred', linewidth = 1.4)
plt.legend(['Mixture PDF'])
plt.title('Histogram Residuals Index (GJR-GARCH)')
plt.xlabel('Index filtered residuals')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()



#================
#For Silver Index
#================



Estimat7 = minimize(ML MN, startingVals, method = 'SLSQP', bounds = bounds, args = Silv GARCH resid std)
Parameters_Silver_GARCH_MN = Estimat7.x

Estimat8 = minimize(ML_MN, startingVals, method = 'SLSQP', bounds = bounds, args = Silv_EGARCH_resid_std)
Parameters_Silver_EGARCH_MN = Estimat8.x

Estimat9 = minimize(ML_MN, startingVals, method = 'SLSQP', bounds = bounds, args = Silv_GJRGARCH_resid_std)
Parameters_Silver_GJRGARCH_MN = Estimat9.x


plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
```

```
plt.hist(Silv_GARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(support, ML_MN(Parameters_Silver_GARCH_MN, support), color = 'darkred', linewidth = 1.4)
plt.legend(['Mixture PDF'])
plt.title('Histogram Residuals Silver (GARCH)')
plt.xlabel('Silver filtered residuals')
plt.ylabel('Frequency')

plt.subplot(132)
plt.hist(Silv EGARCH resid std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(support, ML_MN(Parameters_Silver_EGARCH_MN, support), color = 'darkred', linewidth = 1.4)
plt.legend(['Mixture PDF'])
plt.title('Histogram Residuals Silver (EGARCH)')
plt.xlabel('Silver filtered residuals')
plt.ylabel('Frequency')

plt.subplot(133)
plt.hist(Silv_GJRGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(support, ML_MN(Parameters_Silver_GJRGARCH_MN, support), color = 'darkred', linewidth = 1.4)
plt.legend(['Mixture PDF'])
plt.title('Histogram Residuals Silver (GJR-GARCH)')
plt.xlabel('Silver filtered residuals')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()



#=============================================================================================
#Kolmogorov Smirnov Test for Gaussian Mixture
#=============================================================================================



#CDF
xY_MN = ML_MN(Parameters_AZ_GARCH_MN, np.sort(AZ_GARCH_resid_std))
xY2_MN = ML_MN(Parameters_AZ_EGARCH_MN, np.sort(AZ_EGARCH_resid_std))
xY3_MN = ML_MN(Parameters_AZ_GJRGARCH_MN, np.sort(AZ_GJRGARCH_resid_std))


#For Azimut
ecdfs_MN = np.arange(len(AZ_GARCH_resid_std), dtype=float)/len(AZ_GARCH_resid_std)
GSt_MN = []
maxi_MN = 0
index_MN = 0
for i in range(2890) :
    GSt_MN.append(abs(xY_MN[i] - (i/len(AZ_GARCH_resid_std))))
    if GSt_MN[i] > maxi_MN :
        index_MN = i
        maxi_MN = GSt_MN[i]
    else :
        continue
GSt_MN = np.array(GSt_MN)
max(GSt_MN)


ecdfs2_MN = np.arange(len(AZ_EGARCH_resid_std), dtype=float)/len(AZ_EGARCH_resid_std)
GSt2_MN = []
maxi2_MN = 0
index2_MN = 0
for i in range(len(AZ EGARCH resid std)) :
    GSt2_MN.append(abs(xY2_MN[i] - (i/len(AZ_EGARCH_resid_std))))
    if GSt2_MN[i] > maxi2_MN :
        index2_MN = i
        maxi2_MN = GSt2_MN[i]
    else :
        continue
GSt2_MN = np.array(GSt2_MN)
max(GSt2_MN)



ecdfs3_MN = np.arange(len(AZ_GJRGARCH_resid_std), dtype=float)/len(AZ_GJRGARCH_resid_std)
GSt3_MN = []
maxi3_MN = 0
index3_MN = 0
for i in range(len(AZ_GJRGARCH_resid_std)) :
    GSt3_MN.append(abs(xY3_MN[i] - (i/len(AZ_GJRGARCH_resid_std))))
    if GSt3_MN[i] > maxi3_MN :
        index3_MN = i
        maxi3_MN = GSt3_MN[i]
    else :
        continue
GSt3_MN = np.array(GSt3_MN)
max(GSt3_MN)


plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.plot(ecdfs_MN, xY_MN,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs_MN,ecdfs_MN,color='k',lw=0.8,linestyle='dashed',label="Mixture CDF")
plt.scatter(index_MN/len(AZ_GARCH_resid_std),ecdfs_MN[index_MN],s=7,color='r')
plt.scatter(index_MN/len(AZ_GARCH_resid_std),xY_MN[index_MN],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index_MN/len(AZ_GARCH_resid_std)], ecdfs_MN[index_MN], xY_MN[index_MN], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (Az GARCH residuals)')
```

```
plt.subplot(132)
plt.plot(ecdfs2_MN, xY2_MN,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs2_MN,ecdfs2_MN,color='k',lw=0.8,linestyle='dashed',label="Mixture CDF")
plt.scatter(index2_MN/len(AZ_EGARCH_resid_std),ecdfs2_MN[index2_MN],s=7,color='r')
plt.scatter(index2_MN/len(AZ_EGARCH_resid_std),xY2_MN[index2_MN],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index2_MN/len(AZ_EGARCH_resid_std)], ecdfs2_MN[index2_MN], xY2_MN[index2_MN], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (Az EGARCH residuals)')


plt.subplot(133)
plt.plot(ecdfs3_MN, xY3_MN,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs3 MN,ecdfs3 MN,color='k',lw=0.8,linestyle='dashed',label="Mixture CDF")
plt.scatter(index3_MN/len(AZ_GJRGARCH_resid_std),ecdfs3_MN[index3_MN],s=7,color='r')
plt.scatter(index3_MN/len(AZ_GJRGARCH_resid_std),xY3_MN[index3_MN],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index3_MN/len(AZ_GJRGARCH_resid_std)], ecdfs3_MN[index3_MN], xY3_MN[index3_MN], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (Az GJR-GARCH residuals)')
plt.tight_layout()



#For Italian Index
xY4_MN = ML_MN(Parameters_It_GARCH_MN, np.sort(It_GARCH_resid_std))
xY5_MN = ML_MN(Parameters_It_EGARCH_MN, np.sort(It_EGARCH_resid_std))
xY6_MN = ML_MN(Parameters_It_GJRGARCH_MN, np.sort(It_GJRGARCH_resid_std))


ecdfs4_MN = np.arange(len(It_GARCH_resid_std), dtype=float)/len(It_GARCH_resid_std)
GSt4_MN = []
maxi4_MN = 0
index4_MN = 0
for i in range(len(It_GARCH_resid_std)) :
    GSt4 MN.append(abs(xY4 MN[i] - (i/len(It GARCH resid std))))
    if GSt4_MN[i] > maxi4_MN :
        index4 = i
        maxi4_MN = GSt4_MN[i]
    else :
        continue
GSt4_MN = np.array(GSt4_MN)
max(GSt4_MN)


ecdfs5_MN = np.arange(len(It_EGARCH_resid_std), dtype=float)/len(It_EGARCH_resid_std)
GSt5_MN = []
maxi5_MN = 0
inde5_MN = 0
for i in range(len(It_EGARCH_resid_std)) :
    GSt5_MN.append(abs(xY5_MN[i] - (i/len(It_EGARCH_resid_std))))
    if GSt5_MN[i] > maxi5_MN :
        index5_MN = i
        maxi5_MN = GSt5_MN[i]
    else :
        continue
GSt5_MN = np.array(GSt5_MN)
max(GSt5 MN)


ecdfs6 MN = np.arange(len(It GJRGARCH resid std), dtype=float)/len(It GJRGARCH resid std)
GSt6_MN = []
maxi6_MN = 0
index6_MN = 0
for i in range(len(It_GJRGARCH_resid_std)) :
    GSt6_MN.append(abs(xY6_MN[i] - (i/len(It_GJRGARCH_resid_std))))
    if GSt6_MN[i] > maxi6_MN :
        index6_MN = i
        maxi6_MN = GSt6_MN[i]
    else :
        continue
GSt6_MN = np.array(GSt6_MN)
max(GSt6_MN)



plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.plot(ecdfs4_MN, xY4_MN,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs4_MN,ecdfs4_MN,color='k',lw=0.8,linestyle='dashed',label="Mixture CDF")
plt.scatter(index4_MN/len(It_GARCH_resid_std),ecdfs4_MN[index4_MN],s=7,color='r')
plt.scatter(index4_MN/len(It_GARCH_resid_std),xY4_MN[index4_MN],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index4 MN/len(It GARCH resid std)], ecdfs4 MN[index4 MN], xY4 MN[index4 MN], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (It GARCH residuals)')


plt.subplot(132)
plt.plot(ecdfs5_MN, xY5_MN,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs5_MN,ecdfs5_MN,color='k',lw=0.8,linestyle='dashed',label="Mixture CDF")
plt.scatter(index5_MN/len(It_EGARCH_resid_std),ecdfs5_MN[index5_MN],s=7,color='r')
plt.scatter(index5_MN/len(It_EGARCH_resid_std),xY5_MN[index5_MN],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
```

```
plt.vlines([index5_MN/len(It_EGARCH_resid_std)], ecdfs5_MN[index5_MN], xY5_MN[index5_MN], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (It EGARCH residuals)')




plt.subplot(133)
plt.plot(ecdfs6_MN, xY6_MN,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs6_MN,ecdfs6_MN,color='k',lw=0.8,linestyle='dashed',label="Mixture CDF")
plt.scatter(index6 MN/len(It GJRGARCH resid std),ecdfs6 MN[index6 MN],s=7,color='r')
plt.scatter(index6_MN/len(It_GJRGARCH_resid_std),xY6_MN[index6_MN],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index6_MN/len(It_GJRGARCH_resid_std)], ecdfs6_MN[index6_MN], xY6_MN[index6_MN], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (It GJR-GARCH residuals)')
plt.tight_layout()



#For Silver Index
xY7_MN = ML_MN(Parameters_Silver_GARCH_MN, np.sort(Silv_GARCH_resid_std))
xY8_MN = ML_MN(Parameters_Silver_EGARCH_MN, np.sort(Silv_EGARCH_resid_std))
xY9_MN = ML_MN(Parameters_Silver_GJRGARCH_MN, np.sort(Silv_GJRGARCH_resid_std))


ecdfs7_MN = np.arange(len(Silv_GARCH_resid_std), dtype=float)/len(Silv_GARCH_resid_std)
GSt7_MN = []
maxi7_MN = 0
index7_MN = 0
for i in range(len(Silv_GARCH_resid_std)) :
    GSt7_MN.append(abs(xY7_MN[i] - (i/len(Silv_GARCH_resid_std))))
    if GSt7_MN[i] > maxi7_MN :
        index7 MN = i
        maxi7_MN = GSt7_MN[i]
    else :
        continue
GSt7_MN = np.array(GSt7_MN)
max(GSt7_MN)


ecdfs8_MN = np.arange(len(Silv_EGARCH_resid_std), dtype=float)/len(Silv_EGARCH_resid_std)
GSt8_MN = []
maxi8_MN = 0
inde8_MN = 0
for i in range(len(Silv_EGARCH_resid_std)) :
    GSt8_MN.append(abs(xY5_MN[i] - (i/len(Silv_EGARCH_resid_std))))
    if GSt8_MN[i] > maxi8_MN :
        index8_MN = i
        maxi8_MN = GSt8_MN[i]
    else :
        continue
GSt8_MN = np.array(GSt8_MN)
max(GSt8_MN)


ecdfs9_MN = np.arange(len(Silv_GJRGARCH_resid_std), dtype=float)/len(Silv_GJRGARCH_resid_std)
GSt9_MN = []
maxi9_MN = 0
index9_MN = 0
for i in range(len(Silv_GJRGARCH_resid_std)) :
    GSt9 MN.append(abs(xY9 MN[i] - (i/len(Silv GJRGARCH resid std))))
    if GSt9_MN[i] > maxi9_MN :
        index9_MN = i
        maxi9_MN = GSt9_MN[i]
    else :
        continue
GSt9_MN = np.array(GSt9_MN)
max(GSt9_MN)


plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.plot(ecdfs7_MN, xY7_MN,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs7_MN,ecdfs7_MN,color='k',lw=0.8,linestyle='dashed',label="Mixture CDF")
plt.scatter(index7_MN/len(Silv_GARCH_resid_std),ecdfs7_MN[index7_MN],s=7,color='r')
plt.scatter(index7_MN/len(Silv_GARCH_resid_std),xY7_MN[index7_MN],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index7_MN/len(Silv_GARCH_resid_std)], ecdfs7_MN[index7_MN], xY7_MN[index7_MN], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (Silv GARCH residuals)')


plt.subplot(132)
plt.plot(ecdfs8 MN, xY8 MN,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs8_MN,ecdfs8_MN,color='k',lw=0.8,linestyle='dashed',label="Mixture CDF")
plt.scatter(index8_MN/len(Silv_EGARCH_resid_std),ecdfs8_MN[index8_MN],s=7,color='r')
plt.scatter(index8_MN/len(Silv_EGARCH_resid_std),xY8_MN[index8_MN],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index8_MN/len(Silv_EGARCH_resid_std)], ecdfs8_MN[index8_MN], xY8_MN[index8_MN], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (Silv EGARCH residuals)')


plt.subplot(133)
```

```
plt.plot(ecdfs9_MN, xY9_MN,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs9_MN,ecdfs9_MN,color='k',lw=0.8,linestyle='dashed',label="Mixture CDF")
plt.scatter(index9_MN/len(Silv_GJRGARCH_resid_std),ecdfs9_MN[index9_MN],s=7,color='r')
plt.scatter(index9_MN/len(Silv_GJRGARCH_resid_std),xY9_MN[index9_MN],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index9_MN/len(Silv_GJRGARCH_resid_std)], ecdfs9_MN[index9_MN], xY9_MN[index9_MN], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (Silv GJR-GARCH residuals)')
plt.tight_layout()



#==================================
#Skewed student distribution
#==================================



def Gamma_function(x, a):
    Function = np.power(x, a - 1) * np.exp(- x)
    return Function

def Gamma(a):
    Gamma = integrate.quad(Gamma_function, 0, np.inf, args = a)
    return Gamma[0]

def Function(x, Lambda, Nu):
    Function = sqrt(1 + 3 * np.power(Lambda, 2) - np.power(4 * Lambda * Gamma((Nu + 1) / 2) / (sqrt(pi * (Nu - 2)) * Gamma(Nu / 2)) *
((Nu - 2) / (Nu - 1)), 2)) * Gamma((Nu + 1) / 2) / (sqrt(pi * (Nu - 2)) * Gamma(Nu / 2)) * np.power(1 + (np.power((sqrt(1 + 3 *
np.power(Lambda, 2) - np.power(4 * Lambda * Gamma((Nu + 1) / 2) / (sqrt(pi * (Nu - 2)) * Gamma(Nu / 2)) * ((Nu - 2) / (Nu - 1)), 2)) *
x + 4 * Lambda * Gamma((Nu + 1) / 2) / (sqrt(pi * (Nu - 2)) * Gamma(Nu / 2)) * ((Nu - 2) / (Nu - 1))) / (1 - Lambda), 2) / (Nu - 2)),
- (Nu + 1) / 2)
    return Function

def Skewed_t_ML(parameters, data, out = None):
    Lambda = parameters[0]
    Nu = parameters[1]
    c = Gamma((Nu + 1) / 2) / (sqrt(pi * (Nu - 2)) * Gamma(Nu / 2))
    a = 4 * Lambda * c * ((Nu - 2) / (Nu - 1))
    b = sqrt(1 + 3 * np.power(Lambda, 2) - np.power(a, 2))
    Epsilon = np.zeros((len(data)))

    for i in range(len(data)):
        if data[i] < - (a / b):
            Epsilon[i] = (b * data[i] + a) / (1 - Lambda)
        if data[i] >= - (a / b):
            Epsilon[i] = (b * data[i] + a) / (1 + Lambda)
    PDF = np.zeros((len(data)))
    Likelihood = np.zeros((len(data)))

    for i in range(len(data)):
        PDF[i] = b * c * np.power(1 + (np.power(Epsilon[i], 2) / (Nu - 2)), - (Nu + 1) / 2)
        Likelihood[i] = log(b) + log(c) + (- (Nu + 1) / 2) * log(1 + (np.power(Epsilon[i], 2) / (Nu - 2)))
    Total_Likelihood = (-1) * np.sum(Likelihood)

    if out is None:
        return Total_Likelihood
    else:
        return PDF


def Skewed_t_CDF(parameters, data):
    Lambda = parameters[0]
    Nu = parameters[1]
    c = Gamma((Nu + 1) / 2) / (sqrt(pi * (Nu - 2)) * Gamma(Nu / 2))
    a = 4 * Lambda * c * ((Nu - 2) / (Nu - 1))
    b = sqrt(1 + 3 * np.power(Lambda, 2) - np.power(a, 2))
    CDF = np.zeros((len(data)))
    for i in range(len(data)):
        if data[i] < - (a / b):
            CDF[i] = (1 - Lambda) * t.cdf(((b * data[i] + a) / (1 - Lambda)) * np.power(Nu / (Nu - 2), 0.5), Nu)
            print(CDF[i])

        if data[i] >= - (a / b):
            CDF[i] = (1 + Lambda) * t.cdf(((b * data[i] + a) / (1 + Lambda)) * np.power(Nu / (Nu - 2), 0.5), Nu)  - Lambda
            print(CDF[i])
    return CDF


Bounds_Skewed = [(-1.001, 1.001), (2.001, None)]
x0 = np.array([0, 5])
arg =  np.linspace(-5, 5, 100)



#==========
#For Azimut
#==========



Estimation = minimize(Skewed_t_ML, x0, method = 'SLSQP', bounds = Bounds_Skewed, args = AZ_GARCH_resid_std)
Parameters_AZ_GARCH = Estimation.x

Estimation2 = minimize(Skewed_t_ML, x0, method = 'SLSQP', bounds = Bounds_Skewed, args = AZ_EGARCH_resid_std)
Parameters_AZ_EGARCH = Estimation2.x

Estimation3 = minimize(Skewed_t_ML, x0, method = 'SLSQP', bounds = Bounds_Skewed, args = AZ_GJRGARCH_resid_std)
Parameters_AZ_GJRGARCH = Estimation3.x
```

```
plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.hist(AZ_GARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(arg, Skewed_t_ML(Parameters_AZ_GARCH, arg), color = 'darkred', linewidth = 1.4)
plt.legend(['Skewed t PDF'])
plt.title('Histogram Residuals Azimut (GARCH)')
plt.xlabel('Azimut filtered residuals')
plt.ylabel('Frequency')
plt.subplot(132)
plt.hist(AZ_EGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(arg, Skewed t ML(Parameters AZ EGARCH, arg), color = 'darkred', linewidth = 1.4)
plt.legend(['Skewed t PDF'])
plt.title('Histogram Residuals Azimut (EGARCH)')
plt.xlabel('Azimut filtered residuals')
plt.ylabel('Frequency')

plt.subplot(133)
plt.hist(AZ_GJRGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(arg, Skewed_t_ML(Parameters_AZ_GJRGARCH, arg), color = 'darkred', linewidth = 1.4)
plt.legend(['Skewed t PDF'])
plt.title('Histogram Residuals Azimut (GJR-GARCH)')
plt.xlabel('Azimut filtered residuals')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()



#=================
#For Italian Index
#=================



Estimation4 = minimize(Skewed t ML, x0, method = 'SLSQP', bounds = Bounds Skewed, args = It GARCH resid std)
Parameters_It_GARCH = Estimation4.x

Estimation5 = minimize(Skewed t ML, x0, method = 'SLSQP', bounds = Bounds Skewed, args = It EGARCH resid std)
Parameters_It_EGARCH = Estimation5.x

Estimation6 = minimize(Skewed_t_ML, x0, method = 'SLSQP', bounds = Bounds_Skewed, args = It_GJRGARCH_resid_std)
Parameters_It_GJRGARCH = Estimation6.x


plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.hist(It_GARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(arg, Skewed_t_ML(Parameters_It_GARCH, arg), color = 'darkred', linewidth = 1.4)
plt.legend(['Skewed t PDF'])
plt.title('Histogram Residuals Index (GARCH)')
plt.xlabel('Index filtered residuals')
plt.ylabel('Frequency')

plt.subplot(132)
plt.hist(It_EGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(arg, Skewed_t_ML(Parameters_It_EGARCH, arg), color = 'darkred', linewidth = 1.4)
plt.legend(['Skewed t PDF'])
plt.title('Histogram Residuals Index (EGARCH)')
plt.xlabel('Index filtered residuals')
plt.ylabel('Frequency')

plt.subplot(133)
plt.hist(It_GJRGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(arg, Skewed_t_ML(Parameters_It_GJRGARCH, arg), color = 'darkred', linewidth = 1.4)
plt.legend(['Skewed t PDF'])
plt.title('Histogram Residuals Index (GJR-GARCH)')
plt.xlabel('Index filtered residuals')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()



#================
#For Silver Index
#================



Estimation7 = minimize(Skewed_t_ML, x0, method = 'SLSQP', bounds = Bounds_Skewed, args = Silv_GARCH_resid_std)
Parameters_Silver_GARCH = Estimation7.x

Estimation8 = minimize(Skewed_t_ML, x0, method = 'SLSQP', bounds = Bounds_Skewed, args = Silv_EGARCH_resid_std)
Parameters_Silver_EGARCH = Estimation8.x

Estimation9 = minimize(Skewed_t_ML, x0, method = 'SLSQP', bounds = Bounds_Skewed, args = Silv_GJRGARCH_resid_std)
Parameters_Silver_GJRGARCH = Estimation9.x


plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.hist(Silv_GARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(arg, Skewed_t_ML(Parameters_Silver_GARCH, arg), color = 'darkred', linewidth = 1.4)
plt.legend(['Skewed t PDF'])
plt.title('Histogram Residuals Silver (GARCH)')
plt.xlabel('Silver filtered residuals')
```

```
plt.ylabel('Frequency')

plt.subplot(132)
plt.hist(Silv_EGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(arg, Skewed_t_ML(Parameters_Silver_EGARCH, arg), color = 'darkred', linewidth = 1.4)
plt.legend(['Skewed t PDF'])
plt.title('Histogram Residuals Silver (EGARCH)')
plt.xlabel('Silver filtered residuals')
plt.ylabel('Frequency')

plt.subplot(133)
plt.hist(Silv GJRGARCH resid std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(arg, Skewed_t_ML(Parameters_Silver_GJRGARCH, arg), color = 'darkred', linewidth = 1.4)
plt.legend(['Skewed t PDF'])
plt.title('Histogram Residuals Silver (GJR-GARCH)')
plt.xlabel('Silver filtered residuals')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()



#=============================================================================================
#Kolmogorov Smirnov Test for Skewed t distribution
#=============================================================================================



#CDF
xY = Skewed_t_CDF(Parameters_AZ_GARCH, np.sort(AZ_GARCH_resid_std))
xY2 = Skewed_t_CDF(Parameters_AZ_EGARCH, np.sort(AZ_EGARCH_resid_std))
xY3 = Skewed_t_CDF(Parameters_AZ_GJRGARCH, np.sort(AZ_GJRGARCH_resid_std))


#For Azimut
ecdfs = np.arange(len(AZ GARCH resid std), dtype=float)/len(AZ GARCH resid std)
GSt = []
maxi = 0
index = 0
for i in range(2890) :
    GSt.append(abs(xY[i] - (i/len(AZ_GARCH_resid_std))))
    if GSt[i] > maxi :
        index = i
        maxi = GSt[i]
    else :
        continue
GSt = np.array(GSt)
max(GSt)


ecdfs2 = np.arange(len(AZ_EGARCH_resid_std), dtype=float)/len(AZ_EGARCH_resid_std)
GSt2 = []
maxi2 = 0
index2 = 0
for i in range(len(AZ_EGARCH_resid_std)) :
    GSt2.append(abs(xY2[i] - (i/len(AZ_EGARCH_resid_std))))
    if GSt2[i] > maxi2 :
        index2 = i
        maxi2 = GSt2[i]
    else :
        continue
GSt2 = np.array(GSt2)
max(GSt2)



ecdfs3 = np.arange(len(AZ_GJRGARCH_resid_std), dtype=float)/len(AZ_GJRGARCH_resid_std)
GSt3 = []
maxi3 = 0
index3 = 0
for i in range(len(AZ_GJRGARCH_resid_std)) :
    GSt3.append(abs(xY3[i] - (i/len(AZ_GJRGARCH_resid_std))))
    if GSt3[i] > maxi3 :
        index3 = i
        maxi3 = GSt3[i]
    else :
        continue
GSt3 = np.array(GSt3)
max(GSt3)


plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.plot(ecdfs, xY,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs,ecdfs,color='k',lw=0.8,linestyle='dashed',label="Skewed t CDF")
plt.scatter(index/len(AZ GARCH resid std),ecdfs[index],s=7,color='r')
plt.scatter(index/len(AZ_GARCH_resid_std),xY[index],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index/len(AZ_GARCH_resid_std)], ecdfs[index], xY[index], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (Az GARCH residuals)')


plt.subplot(132)
plt.plot(ecdfs2, xY2,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs2,ecdfs2,color='k',lw=0.8,linestyle='dashed',label="Skewed t CDF")
plt.scatter(index2/len(AZ_EGARCH_resid_std),ecdfs2[index2],s=7,color='r')
plt.scatter(index2/len(AZ_EGARCH_resid_std),xY2[index2],s=7,color='r')
plt.xlabel("t / T")
```

```
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index2/len(AZ_EGARCH_resid_std)], ecdfs2[index2], xY2[index2], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (Az EGARCH residuals)')



plt.subplot(133)
plt.plot(ecdfs3, xY3,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs3,ecdfs3,color='k',lw=0.8,linestyle='dashed',label="Skewed t CDF")
plt.scatter(index3/len(AZ GJRGARCH resid std),ecdfs3[index3],s=7,color='r')
plt.scatter(index3/len(AZ_GJRGARCH_resid_std),xY3[index3],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index3/len(AZ_GJRGARCH_resid_std)], ecdfs3[index3], xY3[index3], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (Az GJR-GARCH residuals)')
plt.tight_layout()



#For Italian Index
xY4 = Skewed_t_CDF(Parameters_It_GARCH, np.sort(It_GARCH_resid_std))
xY5 = Skewed_t_CDF(Parameters_It_EGARCH, np.sort(It_EGARCH_resid_std))
xY6 = Skewed_t_CDF(Parameters_It_GJRGARCH, np.sort(It_GJRGARCH_resid_std))


ecdfs4 = np.arange(len(It_GARCH_resid_std), dtype=float)/len(It_GARCH_resid_std)
GSt4 = []
maxi4 = 0
index4 = 0
for i in range(len(It_GARCH_resid_std)) :
    GSt4.append(abs(xY4[i] - (i/len(It_GARCH_resid_std))))
    if GSt4[i] > maxi4 :
        index4 = i
        maxi4 = GSt4[i]
    else :
        continue
GSt4 = np.array(GSt4)
max(GSt4)


ecdfs5 = np.arange(len(It_EGARCH_resid_std), dtype=float)/len(It_EGARCH_resid_std)
GSt5 = []
maxi5 = 0
inde5 = 0
for i in range(len(It_EGARCH_resid_std)) :
    GSt5.append(abs(xY5[i] - (i/len(It_EGARCH_resid_std))))
    if GSt5[i] > maxi5 :
        index5 = i
        maxi5 = GSt5[i]
    else :
        continue
GSt5 = np.array(GSt5)
max(GSt5)


ecdfs6 = np.arange(len(It_GJRGARCH_resid_std), dtype=float)/len(It_GJRGARCH_resid_std)
GSt6 = []
maxi6 = 0
index6 = 0
for i in range(len(It GJRGARCH resid std)) :
    GSt6.append(abs(xY6[i] - (i/len(It_GJRGARCH_resid_std))))
    if GSt6[i] > maxi6 :
        index6 = i
        maxi6 = GSt6[i]
    else :
        continue
GSt6 = np.array(GSt6)
max(GSt6)



plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.plot(ecdfs4, xY4,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs4,ecdfs4,color='k',lw=0.8,linestyle='dashed',label="Skewed t CDF")
plt.scatter(index4/len(It_GARCH_resid_std),ecdfs4[index4],s=7,color='r')
plt.scatter(index4/len(It_GARCH_resid_std),xY4[index4],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index4/len(It_GARCH_resid_std)], ecdfs4[index4], xY4[index4], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (It GARCH residuals)')


plt.subplot(132)
plt.plot(ecdfs5, xY5,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs5,ecdfs5,color='k',lw=0.8,linestyle='dashed',label="Skewed t CDF")
plt.scatter(index5/len(It_EGARCH_resid_std),ecdfs5[index5],s=7,color='r')
plt.scatter(index5/len(It_EGARCH_resid_std),xY5[index5],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index5/len(It_EGARCH_resid_std)], ecdfs5[index5], xY5[index5], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (It EGARCH residuals)')
```

```
plt.subplot(133)
plt.plot(ecdfs6, xY6,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs6,ecdfs6,color='k',lw=0.8,linestyle='dashed',label="Skewed t CDF")
plt.scatter(index6/len(It_GJRGARCH_resid_std),ecdfs6[index6],s=7,color='r')
plt.scatter(index6/len(It_GJRGARCH_resid_std),xY6[index6],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index6/len(It GJRGARCH resid std)], ecdfs6[index6], xY6[index6], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (It GJR-GARCH residuals)')
plt.tight layout()



#For Silver Index
xY7 = Skewed_t_CDF(Parameters_Silver_GARCH, np.sort(Silv_GARCH_resid_std))
xY8 = Skewed_t_CDF(Parameters_Silver_EGARCH, np.sort(Silv_EGARCH_resid_std))
xY9 = Skewed_t_CDF(Parameters_Silver_GJRGARCH, np.sort(Silv_GJRGARCH_resid_std))


ecdfs7 = np.arange(len(Silv_GARCH_resid_std), dtype=float)/len(Silv_GARCH_resid_std)
GSt7 = []
maxi7 = 0
index7 = 0
for i in range(len(Silv_GARCH_resid_std)) :
    GSt7.append(abs(xY7[i] - (i/len(Silv_GARCH_resid_std))))
    if GSt7[i] > maxi7 :
        index7 = i
        maxi7 = GSt7[i]
    else :
        continue
GSt7 = np.array(GSt7)
max(GSt7)


ecdfs8 = np.arange(len(Silv EGARCH resid std), dtype=float)/len(Silv EGARCH resid std)
GSt8 = []
maxi8 = 0
inde8 = 0
for i in range(len(Silv_EGARCH_resid_std)) :
    GSt8.append(abs(xY8[i] - (i/len(Silv_EGARCH_resid_std))))
    if GSt8[i] > maxi8 :
        index8 = i
        maxi8 = GSt8[i]
    else :
        continue
GSt8 = np.array(GSt8)
max(GSt8)


ecdfs9 = np.arange(len(Silv_GJRGARCH_resid_std), dtype=float)/len(Silv_GJRGARCH_resid_std)
GSt9 = []
maxi9 = 0
index9 = 0
for i in range(len(Silv_GJRGARCH_resid_std)) :
    GSt9.append(abs(xY9[i] - (i/len(Silv_GJRGARCH_resid_std))))
    if GSt9[i] > maxi9 :
        index9 = i
        maxi9 = GSt9[i]
    else :
        continue
GSt9 = np.array(GSt9)
max(GSt9)


plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.plot(ecdfs7, xY7,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs7,ecdfs7,color='k',lw=0.8,linestyle='dashed',label="Skewed t CDF")
plt.scatter(index7/len(Silv_GARCH_resid_std),ecdfs7[index7],s=7,color='r')
plt.scatter(index7/len(Silv_GARCH_resid_std),xY7[index7],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index7/len(Silv_GARCH_resid_std)], ecdfs7[index7], xY7[index7], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (Silv GARCH residuals)')


plt.subplot(132)
plt.plot(ecdfs8, xY8,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs8,ecdfs8,color='k',lw=0.8,linestyle='dashed',label="Skewed t CDF")
plt.scatter(index8/len(Silv_EGARCH_resid_std),ecdfs8[index8],s=7,color='r')
plt.scatter(index8/len(Silv_EGARCH_resid_std),xY8[index8],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index8/len(Silv_EGARCH_resid_std)], ecdfs8[index8], xY8[index8], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (Silv EGARCH residuals)')


plt.subplot(133)
plt.plot(ecdfs9, xY9,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs9,ecdfs9,color='k',lw=0.8,linestyle='dashed',label="Skewed t CDF")
plt.scatter(index9/len(Silv_GJRGARCH_resid_std),ecdfs9[index9],s=7,color='r')
plt.scatter(index9/len(Silv_GJRGARCH_resid_std),xY9[index9],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index9/len(Silv_GJRGARCH_resid_std)], ecdfs9[index9], xY9[index9], color='b', lw=2.5,label="KS test stat")
```

```
plt.legend()
plt.title('Empirical VS Theoretical CDF (Silv GJR-GARCH residuals)')
plt.tight_layout()




#================================================================
#Hyperbolic distribution
#================================================================



def Hyperbolic_ML(parameters, data, out = 2):
    Lambda = parameters[0]
    Alpha = parameters[1]
    Beta = parameters[2]
    Delta = parameters[3]
    Mu = parameters[4]

    K_v = kv(Lambda, Delta * np.power(np.power(Alpha, 2) - np.power(Beta, 2), 0.5))

    petit_a = (np.power(np.sqrt(np.power(Alpha, 2) - np.power(Beta, 2)) / Delta, Lambda)) / (np.sqrt(2 * pi) * K_v)

    PDF = np.zeros((len(data)))
    Likelihood = np.zeros((len(data)))

    for i in range(len(data)):
        PDF[i] = petit_a * np.exp(Beta * (data[i] - Mu)) * kv(Lambda - 0.5, Alpha * np.power(np.power(Delta, 2)+np.power(data[i] - Mu,
2), 0.5)) / np.power(np.power(np.power(Delta, 2) + np.power(data[i] - Mu, 2), 0.5) / Alpha, 0.5 - Lambda)

        Likelihood[i] = np.log(PDF[i])
    Total_Likelihood = (-1) * np.sum(Likelihood)

    if out is None:
        return Total Likelihood
    else:
        return PDF


def Function2(x, parameters):
    Lambda = parameters[0]
    Alpha = parameters[1]
    Beta = parameters[2]
    Delta = parameters[3]
    Mu = parameters[4]

    K_v = kv(Lambda, Delta * np.power(np.power(Alpha, 2) - np.power(Beta, 2), 0.5))

    Function2 = (np.power(np.sqrt(np.power(Alpha, 2) - np.power(Beta, 2)) / Delta, Lambda)) / (np.sqrt(2 * pi) * K_v) * np.exp(Beta *
(x - Mu)) * kv(Lambda - 0.5, Alpha * np.power(np.power(Delta, 2)+np.power(x - Mu, 2), 0.5)) / np.power(np.power(np.power(Delta, 2) +
np.power(x - Mu, 2), 0.5) / Alpha, 0.5 - Lambda)
    return Function2


def Hyperbolic_CDF(parameters, data):
    CDF = np.zeros((len(data)))
    for i in range(len(data)):
        CDF[i] = integrate.quad(Function2, - np.inf, data[i], args = parameters)[0]
        print(CDF[i])
    return CDF


x0 = np.array([0.5, 0.7, 0.5, 0.5, 0.5])
arg =  np.linspace(-5, 5, 100)
cons44 = ({'type':'ineq', 'fun': lambda x: x[1] - abs(x[2]) - 0.01},
          {'type':'ineq', 'fun': lambda x: abs(x[2]) - 0.01},
          {'type':'ineq', 'fun': lambda x: x[3] - 0.01})



#==========
#For Azimut
#==========



Estimation_H = minimize(Hyperbolic_ML, x0, method = 'SLSQP', constraints = cons44, args = AZ_GARCH_resid_std)
Parameters_AZ_GARCH_H = Estimation_H.x

Estimation2_H = minimize(Hyperbolic_ML, x0, method = 'SLSQP', constraints = cons44, args = AZ_EGARCH_resid_std)
Parameters_AZ_EGARCH_H = Estimation2_H.x

Estimation3_H = minimize(Hyperbolic_ML, x0, method = 'SLSQP', constraints = cons44, args = AZ_GJRGARCH_resid_std)
Parameters_AZ_GJRGARCH_H = Estimation3_H.x


plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.hist(AZ_GARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(arg, Hyperbolic_ML(Parameters_AZ_GARCH_H, arg), color = 'darkred', linewidth = 1.4)
plt.legend(['Hyperbolic PDF'])
plt.title('Histogram Residuals Azimut (GARCH)')
plt.xlabel('Azimut filtered residuals')
plt.ylabel('Frequency')

plt.subplot(132)
plt.hist(AZ_EGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(arg, Hyperbolic_ML(Parameters_AZ_EGARCH_H, arg), color = 'darkred', linewidth = 1.4)
```

```
plt.legend(['Hyperbolic PDF'])
plt.title('Histogram Residuals Azimut (EGARCH)')
plt.xlabel('Azimut filtered residuals')
plt.ylabel('Frequency')




plt.subplot(133)
plt.hist(AZ_GJRGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(arg, Hyperbolic_ML(Parameters_AZ_GJRGARCH_H, arg), color = 'darkred', linewidth = 1.4)
plt.legend(['Hyperbolic PDF'])
plt.title('Histogram Residuals Azimut (GJR-GARCH)')
plt.xlabel('Azimut filtered residuals')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()



#=================
#For Italian Index
#=================



Estimation4_H = minimize(Hyperbolic_ML, x0, method = 'SLSQP', constraints = cons44, args = It_GARCH_resid_std)
Parameters_It_GARCH_H = Estimation4_H.x

Estimation5_H = minimize(Hyperbolic_ML, x0, method = 'SLSQP', constraints = cons44, args = It_EGARCH_resid_std)
Parameters_It_EGARCH_H = Estimation5_H.x

Estimation6_H = minimize(Hyperbolic_ML, x0, method = 'SLSQP', constraints = cons44, args = It_GJRGARCH_resid_std)
Parameters_It_GJRGARCH_H = Estimation6_H.x


plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.hist(It_GARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(arg, Hyperbolic_ML(Parameters_It_GARCH_H, arg), color = 'darkred', linewidth = 1.4)
plt.legend(['Hyperbolic PDF'])
plt.title('Histogram Residuals Index (GARCH)')
plt.xlabel('Index filtered residuals')
plt.ylabel('Frequency')

plt.subplot(132)
plt.hist(It_EGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(arg, Hyperbolic_ML(Parameters_It_EGARCH_H, arg), color = 'darkred', linewidth = 1.4)
plt.legend(['Hyperbolic PDF'])
plt.title('Histogram Residuals Index (EGARCH)')
plt.xlabel('Index filtered residuals')
plt.ylabel('Frequency')

plt.subplot(133)
plt.hist(It_GJRGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(arg, Hyperbolic_ML(Parameters_It_GJRGARCH_H, arg), color = 'darkred', linewidth = 1.4)
plt.legend(['Hyperbolic PDF'])
plt.title('Histogram Residuals Index (GJR-GARCH)')
plt.xlabel('Index filtered residuals')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()



#================
#For Silver Index
#================



Estimation7_H = minimize(Hyperbolic_ML, x0, method = 'SLSQP', constraints = cons44, args = Silv_GARCH_resid_std)
Parameters_Silver_GARCH_H = Estimation7_H.x

Estimation8_H = minimize(Hyperbolic_ML, x0, method = 'SLSQP', constraints = cons44, args = Silv_EGARCH_resid_std)
Parameters_Silver_EGARCH_H = Estimation8_H.x

Estimation9_H = minimize(Hyperbolic_ML, x0, method = 'SLSQP', constraints = cons44, args = Silv_GJRGARCH_resid_std)
Parameters_Silver_GJRGARCH_H = Estimation9_H.x


plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.hist(Silv_GARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(arg, Hyperbolic_ML(Parameters_Silver_GARCH_H, arg), color = 'darkred', linewidth = 1.4)
plt.legend(['Hyperbolic PDF'])
plt.title('Histogram Residuals Silver (GARCH)')
plt.xlabel('Silver filtered residuals')
plt.ylabel('Frequency')

plt.subplot(132)
plt.hist(Silv_EGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(arg, Hyperbolic_ML(Parameters_Silver_EGARCH_H, arg), color = 'darkred', linewidth = 1.4)
plt.legend(['Hyperbolic PDF'])
plt.title('Histogram Residuals Silver (EGARCH)')
plt.xlabel('Silver filtered residuals')
```

```
plt.ylabel('Frequency')

plt.subplot(133)
plt.hist(Silv_GJRGARCH_resid_std, bins = "auto", rwidth=2, histtype= 'bar', density = True, color = 'white', stacked=True, edgecolor =
'red')
plt.plot(arg, Hyperbolic_ML(Parameters_Silver_GJRGARCH_H, arg), color = 'darkred', linewidth = 1.4)
plt.legend(['Hyperbolic PDF'])
plt.title('Histogram Residuals Silver (GJR-GARCH)')
plt.xlabel('Silver filtered residuals')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()



#=============================================================================================
#Kolmogorov Smirnov Test for Hyperbolic
#=============================================================================================



#CDF
xY_H = Hyperbolic_CDF(Parameters_AZ_GARCH_H, np.sort(AZ_GARCH_resid_std))
xY2_H = Hyperbolic_CDF(Parameters_AZ_EGARCH_H, np.sort(AZ_EGARCH_resid_std))
xY3_H = Hyperbolic_CDF(Parameters_AZ_GJRGARCH_H, np.sort(AZ_GJRGARCH_resid_std))


#For Azimut
ecdfs_H = np.arange(len(AZ_GARCH_resid_std), dtype=float)/len(AZ_GARCH_resid_std)
GSt_H = []
maxi_H = 0
index_H = 0
for i in range(2890) :
    GSt_H.append(abs(xY_H[i] - (i/len(AZ_GARCH_resid_std))))
    if GSt_H[i] > maxi_H :
        index_H = i
        maxi H = GSt H[i]
    else :
        continue
GSt H = np.array(GSt H)
max(GSt_H)


ecdfs2_H = np.arange(len(AZ_EGARCH_resid_std), dtype=float)/len(AZ_EGARCH_resid_std)
GSt2_H = []
maxi2_H = 0
index2_H = 0
for i in range(len(AZ_EGARCH_resid_std)) :
    GSt2_H.append(abs(xY2_H[i] - (i/len(AZ_EGARCH_resid_std))))
    if GSt2_H[i] > maxi2_H :
        index2_H = i
        maxi2_H = GSt2_H[i]
    else :
        continue
GSt2_H = np.array(GSt2_H)
max(GSt2_H)



ecdfs3_H = np.arange(len(AZ_GJRGARCH_resid_std), dtype=float)/len(AZ_GJRGARCH_resid_std)
GSt3_H = []
maxi3_H = 0
index3_H = 0
for i in range(len(AZ_GJRGARCH_resid_std)) :
    GSt3 H.append(abs(xY3 H[i] - (i/len(AZ GJRGARCH resid std))))
    if GSt3_H[i] > maxi3_H :
        index3_H = i
        maxi3_H = GSt3_H[i]
    else :
        continue
GSt3_H = np.array(GSt3_H)
max(GSt3_H)



plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.plot(ecdfs_H, xY_H,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs_H,ecdfs_H,color='k',lw=0.8,linestyle='dashed',label="Hyperbolic CDF")
plt.scatter(index_H/len(AZ_GARCH_resid_std),ecdfs_H[index_H],s=7,color='r')
plt.scatter(index_H/len(AZ_GARCH_resid_std),xY_H[index_H],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index_H/len(AZ_GARCH_resid_std)], ecdfs_H[index_H], xY_H[index_H], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (Az GARCH residuals)')


plt.subplot(132)
plt.plot(ecdfs2_H, xY2_H,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs2_H,ecdfs2_H,color='k',lw=0.8,linestyle='dashed',label="Hyperbolic CDF")
plt.scatter(index2_H/len(AZ_EGARCH_resid_std),ecdfs2_H[index2_H],s=7,color='r')
plt.scatter(index2_H/len(AZ_EGARCH_resid_std),xY2_H[index2_H],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index2_H/len(AZ_EGARCH_resid_std)], ecdfs2_H[index2_H], xY2_H[index2_H], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (Az EGARCH residuals)')
```

```
plt.subplot(133)
plt.plot(ecdfs3_H, xY3_H,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs3_H,ecdfs3_H,color='k',lw=0.8,linestyle='dashed',label="Hyperbolic CDF")
plt.scatter(index3_H/len(AZ_GJRGARCH_resid_std),ecdfs3_H[index3_H],s=7,color='r')
plt.scatter(index3_H/len(AZ_GJRGARCH_resid_std),xY3_H[index3_H],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index3_H/len(AZ_GJRGARCH_resid_std)], ecdfs3_H[index3_H], xY3_H[index3_H], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (Az GJR-GARCH residuals)')
plt.tight_layout()



#For Italian Index
xY4_H = Hyperbolic_CDF(Parameters_It_GARCH_H, np.sort(It_GARCH_resid_std))
xY5_H = Hyperbolic_CDF(Parameters_It_EGARCH_H, np.sort(It_EGARCH_resid_std))
xY6_H = Hyperbolic_CDF(Parameters_It_GJRGARCH_H, np.sort(It_GJRGARCH_resid_std))


ecdfs4_H = np.arange(len(It_GARCH_resid_std), dtype=float)/len(It_GARCH_resid_std)
GSt4_H = []
maxi4_H = 0
index4_H = 0
for i in range(len(It_GARCH_resid_std)) :
    GSt4_H.append(abs(xY4_H[i] - (i/len(It_GARCH_resid_std))))
    if GSt4_H[i] > maxi4_H :
        index4_H = i
        maxi4_H = GSt4_H[i]
    else :
        continue
GSt4_H = np.array(GSt4_H)
max(GSt4_H)


ecdfs5_H = np.arange(len(It_EGARCH_resid_std), dtype=float)/len(It_EGARCH_resid_std)
GSt5_H = []
maxi5_H = 0
index5_H = 0
for i in range(len(It_EGARCH_resid_std)) :
    GSt5_H.append(abs(xY5_H[i] - (i/len(It_EGARCH_resid_std))))
    if GSt5_H[i] > maxi5_H :
        index5_H = i
        maxi5_H = GSt5_H[i]
    else :
        continue
GSt5_H = np.array(GSt5_H)
max(GSt5_H)


ecdfs6_H = np.arange(len(It_GJRGARCH_resid_std), dtype=float)/len(It_GJRGARCH_resid_std)
GSt6_H = []
maxi6_H = 0
index6_H = 0
for i in range(len(It_GJRGARCH_resid_std)) :
    GSt6_H.append(abs(xY6_H[i] - (i/len(It_GJRGARCH_resid_std))))
    if GSt6_H[i] > maxi6_H :
        index6_H = i
        maxi6_H = GSt6_H[i]
    else :
        continue
GSt6_H = np.array(GSt6_H)
max(GSt6_H)



plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.plot(ecdfs4_H, xY4_H,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs4_H,ecdfs4_H,color='k',lw=0.8,linestyle='dashed',label="Hyperbolic CDF")
plt.scatter(index4_H/len(It_GARCH_resid_std),ecdfs4_H[index4_H],s=7,color='r')
plt.scatter(index4_H/len(It_GARCH_resid_std),xY4_H[index4_H],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index4_H/len(It_GARCH_resid_std)], ecdfs4_H[index4_H], xY4_H[index4_H], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (It GARCH residuals)')


plt.subplot(132)
plt.plot(ecdfs5_H, xY5_H,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs5_H,ecdfs5_H,color='k',lw=0.8,linestyle='dashed',label="Hyperbolic CDF")
plt.scatter(index5_H/len(It_EGARCH_resid_std),ecdfs5_H[index5_H],s=7,color='r')
plt.scatter(index5_H/len(It_EGARCH_resid_std),xY5_H[index5_H],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index5_H/len(It_EGARCH_resid_std)], ecdfs5_H[index5_H], xY5_H[index5_H], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (It EGARCH residuals)')


plt.subplot(133)
plt.plot(ecdfs6_H, xY6_H,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs6_H,ecdfs6_H,color='k',lw=0.8,linestyle='dashed',label="Hyperbolic CDF")
plt.scatter(index6_H/len(It_GJRGARCH_resid_std),ecdfs6_H[index6_H],s=7,color='r')
plt.scatter(index6_H/len(It_GJRGARCH_resid_std),xY6_H[index6_H],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
```

```
plt.vlines([index6_H/len(It_GJRGARCH_resid_std)], ecdfs6_H[index6_H], xY6_H[index6_H], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (It GJR-GARCH residuals)')
plt.tight_layout()



#For Silver Index
xY7_H = Hyperbolic_CDF(Parameters_Silver_GARCH_H, np.sort(Silv_GARCH_resid_std))
xY8_H = Hyperbolic_CDF(Parameters_Silver_EGARCH_H, np.sort(Silv_EGARCH_resid_std))
xY9_H = Hyperbolic_CDF(Parameters_Silver_GJRGARCH_H, np.sort(Silv_GJRGARCH_resid_std))


ecdfs7_H = np.arange(len(Silv_GARCH_resid_std), dtype=float)/len(Silv_GARCH_resid_std)
GSt7_H = []
maxi7_H = 0
index7_H = 0
for i in range(len(Silv_GARCH_resid_std)) :
    GSt7_H.append(abs(xY7_H[i] - (i/len(Silv_GARCH_resid_std))))
    if GSt7_H[i] > maxi7_H :
        index7_H = i
        maxi7_H = GSt7_H[i]
    else :
        continue
GSt7_H = np.array(GSt7_H)
max(GSt7_H)


ecdfs8_H = np.arange(len(Silv_EGARCH_resid_std), dtype=float)/len(Silv_EGARCH_resid_std)
GSt8_H = []
maxi8_H = 0
inde8_H = 0
for i in range(len(Silv_EGARCH_resid_std)) :
    GSt8_H.append(abs(xY8_H[i] - (i/len(Silv_EGARCH_resid_std))))
    if GSt8_H[i] > maxi8_H :
        index8_H = i
        maxi8_H = GSt8_H[i]
    else :
        continue
GSt8_H = np.array(GSt8_H)
max(GSt8 H)


ecdfs9_H = np.arange(len(Silv_GJRGARCH_resid_std), dtype=float)/len(Silv_GJRGARCH_resid_std)
GSt9_H = []
maxi9_H = 0
index9_H = 0
for i in range(len(Silv_GJRGARCH_resid_std)) :
    GSt9_H.append(abs(xY9_H[i] - (i/len(Silv_GJRGARCH_resid_std))))
    if GSt9_H[i] > maxi9_H :
        index9_H = i
        maxi9_H = GSt9_H[i]
    else :
        continue
GSt9_H = np.array(GSt9_H)
max(GSt9_H)


plt.figure(dpi = 1000, figsize=(15, 4))
plt.subplot(131)
plt.plot(ecdfs7_H, xY7_H,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs7_H,ecdfs7_H,color='k',lw=0.8,linestyle='dashed',label="Hyperbolic CDF")
plt.scatter(index7 H/len(Silv GARCH resid std),ecdfs7 H[index7 H],s=7,color='r')
plt.scatter(index7_H/len(Silv_GARCH_resid_std),xY7_H[index7_H],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index7_H/len(Silv_GARCH_resid_std)], ecdfs7_H[index7_H], xY7_H[index7_H], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (Silv GARCH residuals)')


plt.subplot(132)
plt.plot(ecdfs8_H, xY8_H,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs8_H,ecdfs8_H,color='k',lw=0.8,linestyle='dashed',label="Hyperbolic CDF")
plt.scatter(index8_H/len(Silv_EGARCH_resid_std),ecdfs8_H[index8_H],s=7,color='r')
plt.scatter(index8_H/len(Silv_EGARCH_resid_std),xY8_H[index8_H],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index8_H/len(Silv_EGARCH_resid_std)], ecdfs8_H[index8_H], xY8_H[index8_H], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (Silv EGARCH residuals)')


plt.subplot(133)
plt.plot(ecdfs9_H, xY9_H,color='r',lw=1,label="Empirical CDF")
plt.plot(ecdfs9 H,ecdfs9 H,color='k',lw=0.8,linestyle='dashed',label="Hyperbolic CDF")
plt.scatter(index9_H/len(Silv_GJRGARCH_resid_std),ecdfs9_H[index9_H],s=7,color='r')
plt.scatter(index9_H/len(Silv_GJRGARCH_resid_std),xY9_H[index9_H],s=7,color='r')
plt.xlabel("t / T")
plt.ylabel("CDF")
plt.ylim([0, 1]); plt.grid(True)
plt.vlines([index9_H/len(Silv_GJRGARCH_resid_std)], ecdfs9_H[index9_H], xY9_H[index9_H], color='b', lw=2.5,label="KS test stat")
plt.legend()
plt.title('Empirical VS Theoretical CDF (Silv GJR-GARCH residuals)')
plt.tight_layout()



Dataframe = pd.DataFrame({'Skewed-t' : pd.Series([0.023, 0.015, 0.0175, 0.0234, 0.0123, 0.0098, 0.0235, 0.0194, 0.0234],
```

```
                                                index = ['Azimut G', 'Azimut GJR-G', 'Azimut EG', 'Italian G', 'Italian GJR-G',
'Italian EG', 'Silver G', 'Silver GJR-G', 'Silver EG']),
                           'Gaussian Mixture' : pd.Series([0.0195, 0.019, 0.0205, 0.011, 0.01297, 0.0125, 0.0179, 0.0168, 0.023],
                                                index = ['Azimut G', 'Azimut GJR-G', 'Azimut EG', 'Italian G', 'Italian GJR-G',
'Italian EG', 'Silver G', 'Silver GJR-G', 'Silver EG']),
                           'Hyperbolic' : pd.Series([0.0092, 0.0105, 0.0099, 0.0156, 0.014, 0.0165, 0.00988, 0.011, 0.00938],
                                                index = ['Azimut G', 'Azimut GJR-G', 'Azimut EG', 'Italian G', 'Italian GJR-G',
'Italian EG', 'Silver G', 'Silver GJR-G', 'Silver EG'])})

plt.figure(dpi = 1500, figsize=(11, 7))
sns.heatmap(Dataframe, annot = True)
sns.set(font_scale=1.2)
```

- *Part 2:*

```
import datetime as dt
import pandas_datareader.data as web
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from numpy.linalg import inv
from scipy.optimize import minimize
from scipy.stats import norm
import numpy.matlib
import math as mt
from scipy.optimize import fmin_slsqp
from scipy.stats import skew
from scipy.stats import kurtosis

# Importing the data from Google
start = dt.datetime(1989, 12, 31)
end = dt.datetime(2020, 12, 31)
sp500 = web.DataReader('^GSPC', 'yahoo', start=start, end=end)
Price = sp500['Adj Close']
#Price = sp500['Adj Close'].iloc[:7812]
#returns =  sp500['Adj Close'].pct_change().dropna()

logp = np.log(Price)
ret = logp.diff().dropna()
#ret = np.array(ret)

plt.figure(dpi = 1000, figsize=(16, 4))
plt.subplot(121)
plt.plot(logp.index, logp, color = 'red', linewidth = 0.8)
plt.xlabel('Periods (daily)')
plt.ylabel('Log-prices')
plt.title('Log-prices of the S&P 500')

plt.subplot(122)
plt.plot(logp.index[1:], ret, color = 'red', linewidth = 0.8)
plt.xlabel('Periods (daily)')
plt.ylabel('Log-returns')
plt.title('Log-returns of the S&P 500')
plt.tight_layout()

def Stats(Data):
    Mean = np.mean(Data, 0) * 252
    Std = np.std(Data, 0) * np.power(252, 0.5)
    Skewness = skew(Data)
    Kurtosis = kurtosis(Data, fisher = False)
    Min = min(Data)
    Max = max(Data)
    print('Mean is:', Mean)
    print('Std is:', Std)
    print('Skewness is:',  Skewness)
    print('Kurtosis is:', Kurtosis)
    print('Min is:', Min)
    print('Max is:', Max)

Stats(logp)
Stats(ret)



##################### MAXIMUM LIKELIHOOD ESTIMATION
def ML_TR(theta,logreturns,a =None):
    r = 0.0025/252 #daily risk free
    # unbundling parameters
    l=theta[0]
    w=theta[1]
    b=theta[2]
    a=theta[3]
    c=theta[4]
    # loglik computation
    loglik=0
    h= (w + a)/(1-b-(a*c*c))
    for i in range(0,len(logreturns)):
        temp= (logreturns[i]-r-l*h)/(h**0.5)
        #logliks.append(0.5*(np.log(2*np.pi) + np.log(h) + ((logreturns[i]-r-l*h)**2)/h))
        loglik += 0.5*(np.log(2*np.pi) + np.log(h) + ((logreturns[i]-r-l*h)**2)/h)
        x = (temp-c*(h**0.5))**2
        h = w + b*h + a*x

    print(theta)
    print(loglik)
    #logliks = np.array(logliks)
    if a is None:
        return loglik
```

```
    else:
        return loglik #, logliks

# when i had the list logliks the method Nelder-Mead was not working so i didn't use it for MLE
# this logliks list will be used after for the significance of the parameters.

x1 = [5.10, 4.33E-05,  0.589 , 7.97E-07 , 458.20]
x1 = [0.91, 5.04e-07,  0.664 , 1.4e-06 , 463.32]
#tested with the 2 starting values,got the same estimation of parameters
#We found that the negativ Log-Likelihood = -25666.616897536882
estimation_output = minimize(ML_TR, x1, method='Nelder-Mead', args=(ret),options={'maxiter': 10000})
theta = estimation_output.x



################## MONTE CARLO SIMULATION
def black_scholes(sigma,S,K,r,T):
    d1=(np.log(S/K)+(r+0.5*(sigma**2))*T)/(sigma*(T**.5))
    d2=d1-sigma*np.sqrt(T)
    C=S*norm.cdf(d1)-K*mt.exp(-r*T)*norm.cdf(d2)
    return C

def criterion_implied_vol(S,K,r,T,C):
    sigma=np.linspace(0.01,1,1000)
    err=np.ones(len(sigma))
    for i in range(0,len(err)):
        err[i]=(black_scholes(sigma[i],S,K,r,T)-C)**2
    sigma_opt=sigma[np.where(err==min(err))]
    if len(sigma_opt)>1:
        sigma_opt= sigma_opt[len(sigma_opt)-1]
    return sigma_opt



def MC(N,T,theta,r):
    #Depending on the standard normal noises generation the Monte-Carlo simulation can fail,but please just retry and it would work.
    r = r/252 #daily risk free
    S = Price[-1] # Price of S&P500 at 31/12/20
    K=np.linspace(S*0.9,S*1.1, 200)
    returns_sum=np.ones(N)*0
    l=theta[0]
    w=theta[1]
    b=theta[2]
    a=theta[3]
    c=theta[4]
    for i in range(0,N):
        z=np.random.randn(T) # Gaussian random variable
        returns =np.ones(T)*0 # returns only contains zeros
        h=np.ones(T) * ((w + a)/(1-b-(a*c*c)))
        for t in range(1,T):
            h[t] = w + a * (z[t-1] -c*(h[t-1]**0.5))**2 + b * h[t-1]
            returns[t] = r +l*h[t] + (h[t]**0.5)*z[t]
        returns_sum[i]=np.sum(returns)
    Hess_Prices = S*np.exp(returns_sum)
    Strikes=np.matlib.repmat(K,len(Hess_Prices),1)
    Hess_Option_Prices=np.matlib.repmat(Hess_Prices,len(K),1)-Strikes.T
    Hess_Option_Prices[np.where(Hess_Option_Prices<0)]=0
    Hess_Calls=np.mean(Hess_Option_Prices,1)*np.exp(r*(-T))

    Hess_Option_Prices= -np.matlib.repmat(Hess_Prices,len(K),1)+Strikes.T
    Hess_Option_Prices[np.where(Hess_Option_Prices<0)]=0
    Hess_Puts=np.mean(Hess_Option_Prices,1)*np.exp(r*(-T))

    IV_Hess=np.ones(len(Hess_Calls))
    for i in range(0,len(Hess_Calls)):
        IV Hess[i]=criterion implied vol(S,K[i],r*252,T/252,Hess Calls[i])

    return IV_Hess,Hess_Calls,Hess_Puts,Hess_Prices
r = 0.0025
a = MC(10000,63,theta,r)
b = MC(10000,126,theta,r)
c = MC(10000,252,theta,r)



S=Price[-1]
K=np.linspace(S*0.9,S*1.1, 200)

plt.figure(dpi = 1000, figsize=(8, 4))
plt.plot(K/S,a[0],'darkgrey',label='T = 3 months')
plt.plot(K/S,b[0],'red',label='T = 6 months')
plt.plot(K/S,c[0],'darkred',label='T = 12 months')
plt.title('IV on the 31/12/2020 for maturities equal to T months')
plt.xlabel('Moneyness')
plt.ylabel('Implied Volatility')
plt.legend(loc='upper right')
plt.tight_layout()
plt.savefig('IV.png', dpi=1400)


plt.figure(dpi = 1000, figsize=(16, 4))
plt.subplot(121)
plt.plot(K/S,a[1],'darkgrey',label='T = 3 months')
plt.plot(K/S,b[1],'r',label='T = 6 months')
plt.plot(K/S,c[1],'darkred',label='T = 12 months')
plt.title('Call prices on the 31/12/2020 for maturities equal to T months')
plt.xlabel('Moneyness')
plt.ylabel('Call prices')
plt.legend(loc='upper right')

plt.subplot(122)
plt.plot(K/S,a[2],'darkgrey',label='T = 3 months')
plt.plot(K/S,b[2],'r',label='T = 6 months')
```

```
plt.plot(K/S,c[2],'darkred',label='T = 12 months')
plt.title('Put prices on the 31/12/2020 for maturities equal to T months')
plt.xlabel('Moneyness')
plt.ylabel('Put prices')
plt.legend(loc='upper left')
plt.tight_layout()


K=np.linspace(0,10000, 10000)/10000
plt.plot(K,np.sort(a[3]),'darkgrey',label='T = 3 months')
plt.plot(K,np.sort(b[3]),'r',label='T = 6 months')
plt.plot(K,np.sort(c[3]),'darkred',label='T = 12 months')
plt.title('S&P500 Simulated prices for maturities equal to T months')
plt.xlabel('Prob')
plt.ylabel('Pice level')
plt.legend(loc='upper left')
plt.tight_layout()
plt.savefig('Prices2.png', dpi=1400)


# Numerical approximation of the score function and final estimation results display
def ML_TR(theta,logreturns,a =None):
    r = 0.0025/252 #daily risk free
    # unbundling parameters
    l=theta[0]
    w=theta[1]
    b=theta[2]
    a=theta[3]
    c=theta[4]
    # loglik computation
    loglik=0
    logliks = []
    h= (w + a)/(1-b-(a*c*c))
    for i in range(0,len(logreturns)):
        temp= (logreturns[i]-r-l*h)/(h**0.5)
        logliks.append(0.5*(np.log(2*np.pi) + np.log(h) + ((logreturns[i]-r-l*h)**2)/h))
        loglik += 0.5*(np.log(2*np.pi) + np.log(h) + ((logreturns[i]-r-l*h)**2)/h)
        x = (temp-c*(h**0.5))**2
        h = w + b*h + a*x

    print(theta)
    print(loglik)
    logliks = np.array(logliks)
    if a is None:
        return loglik
    else:
        return loglik,logliks

T = len(ret)

step = 1e-5 * theta
scores = np.zeros((T,5))
for i in range(5):
    h = step[i]
    delta = np.zeros(5)
    delta[i] = h

    loglik, logliksplus = ML_TR(theta + delta,ret,a=True)
    loglik, logliksminus = ML_TR(theta - delta,ret,a=True)

    scores[:,i] = (logliksplus - logliksminus)/(2*h)

I = (scores.T @ scores)/T
vcv=np.mat(inv(I))/T
vcv = np.asarray(vcv)

output = np.vstack((theta,np.sqrt(np.diag(vcv)),theta/np.sqrt(np.diag(vcv)))).T
print('Parameter   Estimate       Std. Err.      T-stat')
print("-------------------------------------------------")
param = ['lambda','omega','beta','alpha','gamma']
for i in range(len(param)):
    print('{0:<11} {1:>0.6f}        {2:0.6f}    {3: 0.5f}'.format(param[i],
          output[i,0], output[i,1], output[i,2]))


######################### PARAMETERS VARIATIONS OF THE MODEL


print(theta)
theta = [2.77438947e+00, -6.40531260e-07  ,8.35786228e-01 , 4.94804393e-06  ,1.58344732e+02]


######################### LAMBDA VARIATION

theta3 = [2.77721542e+00 *1.3, -6.40062014e-07  ,8.35774790e-01  ,4.94809522e-06 ,1.58348201e+02]
theta33 = [2.77721542e+00 *0.7, -6.40062014e-07  ,8.35774790e-01  ,4.94809522e-06 ,1.58348201e+02]

d1 = MC(10000,63,theta3,r)
dd1 = MC(10000,63,theta33,r)



######################### OMEGA VARIATION

theta3 = [2.77721542e+00 , -6.40062014e-07*1.3  ,8.35774790e-01  ,4.94809522e-06 ,1.58348201e+02]
theta33 = [2.77721542e+00 , -6.40062014e-07 *0.7 ,8.35774790e-01  ,4.94809522e-06 ,1.58348201e+02]


d2 = MC(10000,63,theta3,r)
dd2 = MC(10000,63,theta33,r)
```

```
######################### BETA VARIATION
theta3 = [2.77721542e+00 , -6.40062014e-07  ,8.35774790e-01 *1.04 ,4.94809522e-06 ,1.58348201e+02]
theta33 = [2.77721542e+00 , -6.40062014e-07 ,8.35774790e-01  *0.9601 ,4.94809522e-06 ,1.58348201e+02]


d3 = MC(10000,63,theta3,r)
dd3 = MC(10000,63,theta33,r)


plt.figure(dpi = 1000, figsize=(8, 4))
plt.plot(K/S,a[0],'r',label='Beta optimal')
plt.plot(K/S,d3[0],'darkred',label='Beta + 4% ')
plt.plot(K/S,dd3[0],'darkgrey',label='Beta - 4% ')
plt.title('Beta variation for 3 months')
plt.xlabel('Moneyness')
plt.ylabel('Implied Volatility')
plt.legend(loc='upper right')
plt.tight_layout()
plt.savefig('IVBeta.png', dpi=1400)


######################### ALPHA VARIATION
theta3 = [2.77721542e+00 , -6.40062014e-07  ,8.35774790e-01  ,4.94809522e-06 * 1.2, 1.58348201e+02]
theta33 = [2.77721542e+00 , -6.40062014e-07 ,8.35774790e-01  ,4.94809522e-06 * 0.8, 1.58348201e+02]


d4 = MC(10000,63,theta3,r)
dd4 = MC(10000,63,theta33,r)



######################### GAMMA VARIATION
theta3 = [2.77721542e+00 , -6.40062014e-07  ,8.35774790e-01  ,4.94809522e-06 ,1.58348201e+02 * 1.1]
theta33 = [2.77721542e+00 , -6.40062014e-07 ,8.35774790e-01   ,4.94809522e-06, 1.58348201e+0 * 0.9]


d5 = MC(10000,63,theta3,r)
dd5 = MC(10000,63,theta33,r)




plt.figure(dpi = 1000, figsize=(16, 8))
plt.subplot(221)
plt.plot(K/S,a[0],'r',label='lambda optimal')
plt.plot(K/S,d1[0],'darkred',label='lambda + 30% ')
plt.plot(K/S,dd1[0],'darkgrey',label='lambda - 30% ')
plt.title('Lambda variation for 3 months')
plt.xlabel('Moneyness')
plt.ylabel('Implied Volatility')
plt.legend(loc='upper right')
#plt.tight_layout()
#plt.savefig('IVLambda.png', dpi=1400)

plt.subplot(222)
plt.plot(K/S,a[0],'r',label='Omega optimal')
plt.plot(K/S,d2[0],'darkred',label='Omega + 30% ')
plt.plot(K/S,dd2[0],'darkgrey',label='Omega - 30% ')
plt.title('Omega variation for 3 months')
plt.xlabel('Moneyness')
plt.ylabel('Implied Volatility')
plt.legend(loc='upper right')
#plt.tight layout()
#plt.savefig('IVomega.png', dpi=1400)

plt.subplot(223)
plt.plot(K/S,a[0],'r',label='Gamma optimal')
plt.plot(K/S,d5[0],'darkred',label='Gamma + 10% ')
plt.plot(K/S,dd5[0],'darkgrey',label='Gamma - 10% ')
plt.title('Gamma variation for 3 months')
plt.xlabel('Moneyness')
plt.ylabel('Implied Volatility')
plt.legend(loc='upper right')
#plt.tight_layout()
#plt.savefig('IVGamma.png', dpi=1400)

plt.subplot(224)
plt.plot(K/S,a[0],'r',label='Alpha optimal')
plt.plot(K/S,d4[0],'darkred',label='Alpha + 20% ')
plt.plot(K/S,dd4[0],'darkgrey',label='Alpha - 20% ')
plt.title('Alpha variation for 3 months')
plt.xlabel('Moneyness')
plt.ylabel('Implied Volatility')
plt.legend(loc='upper right')
plt.tight_layout()
#plt.savefig('IVAlpha.png', dpi=1400)




##############################################################################################
# Options pricing with Risk neutral distributions

def MCN(N,T,theta,r):
    #Depending on the standard normal noises generation the Monte-Carlo simulation can fail,but please just retry and it would work.
    r = r/252 #daily risk free
    S = Price[-1] # Price of S&P500 at 31/12/20
    K=np.linspace(S*0.9,S*1.1, 200)
    returns_sum=np.ones(N)*0
    l=theta[0]
    w=theta[1]
```

```
    b=theta[2]
    a=theta[3]
    c=theta[4] + l + 0.5
    for i in range(0,N):
        z=np.random.randn(T) # Gaussian random variable
        returns =np.ones(T)*0 # returns only contains zeros
        h=np.ones(T) * ((w + a)/(1-b-(a*c*c)))
        for t in range(1,T):
            h[t] = w + a * (z[t-1] -c*(h[t-1]**0.5))**2 + b * h[t-1]
            returns[t] = r -0.5*h[t] + (h[t]**0.5)*z[t]
        returns_sum[i]=np.sum(returns)
    Hess_Prices = S*np.exp(returns_sum)
    Strikes=np.matlib.repmat(K,len(Hess Prices),1)
    Hess_Option_Prices=np.matlib.repmat(Hess_Prices,len(K),1)-Strikes.T
    Hess_Option_Prices[np.where(Hess_Option_Prices<0)]=0
    Hess Calls=np.mean(Hess Option Prices,1)*np.exp(r*(-T))

    Hess_Option_Prices= -np.matlib.repmat(Hess_Prices,len(K),1)+Strikes.T
    Hess_Option_Prices[np.where(Hess_Option_Prices<0)]=0
    Hess_Puts=np.mean(Hess_Option_Prices,1)*np.exp(r*(-T))

    IV_Hess=np.ones(len(Hess_Calls))
    for i in range(0,len(Hess_Calls)):
        IV_Hess[i]=criterion_implied_vol(S,K[i],r*252,T/252,Hess_Calls[i])

    return IV_Hess,Hess_Calls,Hess_Puts,Hess_Prices



a = MCN(10000,63,theta,r)
b = MCN(10000,126,theta,r)
c = MCN(10000,252,theta,r)

print(4.94804e-06*(158.345**2))


plt.figure(dpi = 1000, figsize=(8, 4))
plt.plot(K/S,a[0],'darkgrey',label='T = 3 months')
plt.plot(K/S,b[0],'red',label='T = 6 months')
plt.plot(K/S,c[0],'darkred',label='T = 12 months')
plt.title('IV on the 31/12/2020 for maturities equal to T months')
plt.xlabel('Moneyness')
plt.ylabel('Implied Volatility')
plt.legend(loc='upper right')
plt.tight_layout()
plt.savefig('IVRiskNeutral.png', dpi=1400)


plt.figure(dpi = 1000, figsize=(16, 4))
plt.subplot(121)
plt.plot(K/S,a[1],'darkgrey',label='T = 3 months')
plt.plot(K/S,b[1],'r',label='T = 6 months')
plt.plot(K/S,c[1],'darkred',label='T = 12 months')
plt.title('Call prices on the 31/12/2020 for maturities equal to T months')
plt.xlabel('Moneyness')
plt.ylabel('Call prices')
plt.legend(loc='upper right')
#plt.tight_layout()
#plt.savefig('CallPricesRiskNeutral.png', dpi=1400)

plt.subplot(122)
plt.plot(K/S,a[2],'darkgrey',label='T = 3 months')
plt.plot(K/S,b[2],'r',label='T = 6 months')
plt.plot(K/S,c[2],'darkred',label='T = 12 months')
plt.title('Put prices on the 31/12/2020 for maturities equal to T months')
plt.xlabel('Moneyness')
plt.ylabel('Put prices')
plt.legend(loc='upper left')
plt.tight_layout()
#plt.savefig('PutPricesRiskNeutral.png', dpi=1400)


plt.figure(dpi = 1000, figsize=(8, 4))
K=np.linspace(0,10000, 10000)/10000
plt.plot(K,np.sort(a[3]),'darkgrey',label='T = 3 months')
plt.plot(K,np.sort(b[3]),'r',label='T = 6 months')
plt.plot(K,np.sort(c[3]),'darkred',label='T = 12 months')
plt.title('S&P500 Simulated prices* for maturities equal to T months')
plt.xlabel('Prob')
plt.ylabel('Pice level')
plt.legend(loc='upper left')
plt.tight_layout()
plt.savefig('PricesRiskneutral.png', dpi=1400)
```