Practical Case Study 2020 PCS-30

Volatility Estimation by GARCH-models in Financial Markets

Sander Schleeper (2622915) - s.h.schleeper@student.vu.nl
Walter Verwer (2611321) - w.verwer@student.vu.nl
Maurits van den Oever (2613642) - m.c.vanden.oever@student.vu.nl
Alex van Beek (2623023) – a.w.c.van.beek@student.vu.nl

April 5, 2020

Table of content

1 Abstract

In this paper the performance of several heteroskedasticity forecast models are examined for the S&P 500 index and the Euro Stoxx 50 index. These are the GARCH, GJR-GARCH, and E-GARCH models with normal, student-t, and skewed student-t innovation distribution assumptions. The model parameters are estimated over a fixed window of 1500 observations using maximum likelihood estimation as well as the Bayesian method. To compare models, the out of sample mean absolute error, logarithmic scoring rule, Diebold Mariano tests, unconditional coverage property tests, independence tests and joint tests are conducted. Overall, the GARCH with the student-t and the GARCH with the skewed student-t proved to outperform the E-GARCH and GJR-GARCH models. Moreover, after Bayesian estimation for the parameters of the GARCH student-t model, superior outcomes have been observed over the estimation by maximum likelihood based on the mean absolute error and the logarithmic scoring rule. However, Bayesian estimation does not provide superior results in regard to Value at Risk forecasting and the results on the backtest of these VaR forecasts. Overall, the main conclusion is that risk is highly overestimated in this report due turbulent financial times in the beginning, the in-sample data, of the data-set and stable financial times in the later stages of the data, namely the out-of-sample data.

Keywords: Daily Stock Returns, Conditional Variance, Auto Correlation, GARCH, EGARCH, GJR-GARCH. Maximum Likelihood Estimation, Innovations Distributions, Normal Distribution, Student-t Distribution, Skewed Student-t Distribution, Forecasting, Mean Absolute Error, Logarithmic Scoring, Diebold Mariano Test, Backtesting, Value at Risk (VaR), Unconditional Coverage Test, Independence Test, Joint Test, Bayesian Estimation

2 Introduction

This report is written by four students of the Vrije Universiteit Amsterdam for the course Practical Case Study 2020, part of the minor Applied Econometrics. The aim of this report is to use the daily returns of the Standard and Poor's 500, hereafter referred to as S&P 500, and the Euro Stoxx 50 index, for estimating the Generalized Auto Regressive Conditional Heteroskedasticity (GARCH) model and some of its extensions in order to compute one-day ahead forecast of the volatility. For the innovations of the models several distribution are compared namely normal, student-t, and skewed student-t. In order to evaluate whether the models correctly predict the conditional variances, the Diebold-Mariano test is applied to the MAE and the logarithmic scoring rule. From the conditional variances, the Value at Risk is calculated, which is evaluated using backtesting methods. Several questions are taken into account when evaluating these models. The first question is: which type of GARCH model can predict volatility the best? It is hypothesized that models that take a leverage effect into account (EGARCH, GJR-GARCH) will perform better than a model that does not take it into account (GARCH). The second question is to find out which distribution assumption is best suited for the model innovations. Because returns are generally fat-tailed and skewed, it is hypothesized that the skewed student-t distribution will lead to better forecasting performance compared to the student-t and normal distribution, and that the student-t will perform better than the normal distribution. The third question is to see if the GARCH models in consideration can accurately predict the risk measure Value at Risk. In research by So Yu (2005) and Smolović, Lipovina-Božović & Vujošević (2017) GARCH models perform well in forecasting the Value at Risk, so it is expected to see the same, especially considering the volatility cluster capturing capabilities of GARCH models. The fourth question is whether or not Bayesian estimation leads to better forecasting performance. Because, at the moment, the authors do not have a background in Bayesian econometrics, no hypothesis can be drawn in regard to the performances of Bayesian estimation in comparison to general maximum likelihood estimation.

3 Data Description

3.1 Returns

The end of day closing prices (of indices) are known to be non-stationary time series. For this reason, the returns are used instead. A return is the relative change in the price of a financial asset over a given time interval, often expressed as a percentage. In this paper continuously compounded returns are used. These are defined as the natural logarithm of daily returns, which are computed by:

$$y_t = \ln(P_t) - \ln(P_{t-1}) \tag{1}$$

Where P_t is the adjusted close price at time point t.

The reason log returns are used instead of simple returns, is that log returns are mathematically easier to use, because the log returns are symmetric and because the log return of a period of multiple days is simply the sum of the daily log returns. Moreover, in almost all other scientific research log returns are used for GARCH models for daily stock returns (Hoogerheide & Ardia, 2014).

3.2 Data

The data used are from the S&P 500 and Euro Stoxx 50 and are obtained from Yahoo Finance. The time interval for the S&P 500 is from 5 January 2010 till 3 January 2020 and for Euro Stoxx 50 from 6 January 2010 till 3 January 2020, the prices and log returns graphs of both data sets are given in appendix A. In table 1 one can see standard summary statistics of the entire data sets regarding the continuously compounded returns and the adjusted closing price of the respective indices.

Index	S&P 500(y_t)	Euro Stoxx 50(y_t)	S&P 500(P_t)	Euro Stoxx 50(P_t)
Observations	2516	2508	2517	2509
Mean	0.0004	0.0001	1964.0	3049.2
Median	0.0006	0.0004	1987.0	3056.4
St.dev	0.0094	0.0125	589.67	406.82
Min	-0.0670	-0.0901	1022.6	1995.0
Max	0.0484	0.0985	3257.9	3828.8
Skewness	-0.4964	-0.1593	0.2046	-0.3616
Kurtosis	7.5924	7.5789	1.9009	2.3220
JB Statistic	2314.3	2201.6	144.25	102.74

Table 1: Summary statistics of the S&P 500 and Euro Stoxx 50 index. y_t denotes the log returns and P_t denotes the adjusted closing price of the respective index. Note that y_t is not expressed as a percentage. JB statistic stands for the Jarque-Bera test statistic, obtained from the Jarque-Bera test, which tests for normality of the data; critical value for $\alpha = 0.05$ is approximately 5.970.

Appendix L Table 16 shows the summary statistics of the log returns divided in the in sample and out of sample periods. It's notable that the in sample standard deviation is larger than the out of sample for both data-sets. This might lead to implications in our conditional variance forecasts and parameter estimates, mainly, because of the use of a fixed estimation window the variances could be overestimated due to the parameters being estimated in more volatile times.

3.3 Auto correlation

As shown in figure 2 and 3 in Appendix B the S&P500 and Euro Stoxx 50 data sets have little to none autocorrelation in the lagged values of returns y_t which is a common aspect of returns. However the squared returns y_t^2 do contain a significant amount of autocorrelation in the lags indicating that the variances of both data sets can, in some extent, be predicted.

4 Models

The mean equation for all models is an AR(0) without constant:

$$y_t = \epsilon_t \tag{2}$$

The AR(0) equation for the returns was chosen above an AR(p) process to prevent misspecification in the models and also because there is no significant autocorrelation in the lags of the log-returns for both the S&P 500 and Euro Stoxx 50, shown in appendix B figure 2 and 3. Additionally no constant is used in the AR(0) equation because the mean log return is approximately zero in both data sets.

4.1 GARCH(1,1)

The GARCH model by Bollerslev (1986) is an extension of the earlier discovered ARCH model by adding the memory component σ_{t-1}^2 as regressor to create the following model for the conditional volatility:

$$\sigma_t^2 = \omega + \alpha\epsilon_{t-1}^2 + \beta\sigma_{t-1}^2 \tag{3}$$

The main advantage of the GARCH(1,1) model over an ARCH model is that it requires fewer parameters to be estimated, therefore it avoids over-fitting, saves computational power and bypasses a lot of estimation errors of the parameters. By performing recursive substitution of the σ_{t-1}^2 component it can be shown that the GARCH(1,1) model is actually a restricted ARCH(∞) model. Therefore it is in many cases not reasonable to prefer an ARCH(p) model over a GARCH(1,1). To ensure that the conditional variance is always positive, we impose that $\omega > 0$, $\alpha \geq 0$ and $\beta \geq 0$.

4.2 E-GARCH(1,1)

The second model used is the EGARCH(1,1) model. Introduced by Nelson (1991), its formula looks like the follows:

$$\log(\sigma_t^2) = \omega + \alpha(|\frac{\epsilon_{t-1}}{\sigma_{t-1}}| - \mathbb{E}[|\frac{\epsilon_{t-1}}{\sigma_{t-1}}|] + \gamma\frac{\epsilon_{t-1}}{\sigma_{t-1}}) + \beta\log(\sigma_{t-1}^2) \tag{4}$$

Because the E-GARCH is specified in logs, there are no additional restrictions necessary to keep the variance from becoming negative. Another advantage over a standard GARCH(p,q) is that the E-GARCH is able to capture an asymmetric effect on volatility based on the sign of innovations, denoted as ϵ_t. Positive innovations have a $\alpha + \gamma$ effect on the variance and negative innovations have a $\alpha - \gamma$ effect.

4.3 GJR-GARCH(1,1)

The GJR-GARCH model by Glosten, Jagannathan and Runkle (1993) is a variation on the standard GARCH model. Like the EGARCH, the GJR-GARCH can capture an asymmetric effect of the innovations, but does so by adding a dummy variable component for the ARCH part of the model. The formula of the GJR-GARCH(1,1) model is:

$$\sigma_t^2 = \omega + \alpha\epsilon_{t-1}^2 + \gamma\epsilon_{t-1}^2 D_{t-1} + \beta\sigma_{t-1}^2 \tag{5}$$

Where: $D_t = 1$ for $\epsilon_t < 0$ and $D_t = 0$ for $\epsilon_t \geq 0$. To ensure the variance is always positive, we impose that $\omega > 0,\ \alpha \geq 0,\ \beta \geq 0$ and $\alpha + \gamma \geq 0$.

5 Maximum Likelihood Estimation

The estimation of the model parameters is done using maximum likelihood estimation. This corresponds to the frequentist approach. In this report the Sequential Least Squares Programming algorithm (SLSQP) is used to optimize the log likelihood function $f(\cdot)$ with respect to the parameters θ, conditional on ϵ_t. The optimizer maximizes the likelihood by finding the minimum value of the negative log likelihood. Mathematically the log likelihood is equivalent to:

$$\ln\mathcal{L}(\epsilon_1, ..., \epsilon_T|\theta) = \sum_{t=1}^{T}\ln f(\epsilon_t|\theta) \tag{6}$$

The log likelihood function is defined in logs, because it is often difficult to maximize a product of densities. Nevertheless, the optima of the parameters are unaffected by the logarithmic transformation. In order to get sensible estimates, the optimizer restricts the parameters α, β and ω to be larger or equal to 0, in order to ensure a positive volatility estimate. In this report no stationarity constraint has been imposed, for this can lead to multiple problems, including multiple parameter estimates, non-unique volatility forecasts and volatility forecasts that jump over time. To estimate the parameters given the data that is assumed to be distributed according to a selected distribution, ϵ_t have been scaled by a factor of 100. The reason for this is that our optimizer can estimate the parameters better when the scale of ϵ_t is between 1 and 1000. Our scale value of ϵ_t without scaling is approximately 0.0001, for this scale value our optimizer recommended to scale ϵ_t by a factor of 100. All the affected estimates have been scaled back after estimation.

6 Innovation Distributions

For the distribution of the innovations the normal, the Student-T and an asymmetric fat-tailed skewed Student-T distribution is used. The subsections below show the formulas for the mentioned distributions.

6.1 Normal distribution:

If the data is normally distributed it is likely that the normal distribution will outperform the other models. However, the past exhibits that this might not be the case due to the observation of fat tails. In this scenario, the Student-T distribution and skewed Student-T distribution are likely to outperform the performance of the model based on a normal distribution.

The normal probability density function for ϵ_t is given by:

$$f(\epsilon_t|\theta) = \frac{1}{\sqrt{2\pi\sigma_t^2}}\exp\left(-\frac{1}{2}\frac{\epsilon_t^2}{\sigma_t^2}\right) \tag{7}$$

6.2 Student-T distribution:

After running tests for normality of the residuals, see appendix C, it is possible to conclude that the returns are not perfectly normally distributed and therefore the Student-T is expected to produce superior estimations over the normal distribution. The non-normality can be caused by fat tails in the data and this indicates that the data exhibits more extreme, and less normal outcomes than one would expect from a random variable with the same mean and variance.

The probability density of ϵ_t is given by (Tsay, 2010):

$$f(\epsilon_t|\theta,\nu) = \frac{\Gamma((v+1)/2)}{\Gamma(v/2)\sqrt{(v-2)\pi\sigma_t^2}}\left(1+\frac{\epsilon_t^2}{(v-2)\sigma_t^2}\right)^{-(v+1)/2}, v>2 \tag{8}$$

6.3 Skewed Student-T distribution:

Besides the fact that empirical distributions of financial returns exhibit fat tails they could also be skewed. After testing for skewness of the S&P 500 and Euro Stoxx 50, see table 1, it is possible to conclude that the data are skewed. A way to model this extra return characteristic is by means of a skewed Student-T distribution. The probability density function of a skewed Student-T distribution is given by Hansen (1994):

$$f(\epsilon_t|\theta,\nu,\lambda) = \frac{bc}{\sigma}\left(1+\frac{1}{\nu-2}\left(\frac{a+b\epsilon_t/\sigma}{1+sgn(\epsilon_t/\sigma+a/b)\lambda}\right)^2\right)^{-(\nu+1)/2} \tag{9}$$

The parameters have the following interpretation: λ is the skewness parameter, ν is the degrees of freedom and a, b and c are given below:

$$a = 4\lambda c\frac{\nu-2}{\nu-1}, \quad b = \sqrt{1+3\lambda^2-a^2}, \quad c = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\pi(\nu-2)}\Gamma\left(\frac{\nu}{2}\right)}$$

Note: if λ is set to zero, then it can be proved that the skewed student-T distribution becomes a student-T distribution.

7 Forecasting

For the forecasting of the volatility and the Value at Risk, the data sets are divided in a in-sample estimation period of the first 1500 observations and a out-of-sample period for the rest of the observations. For the S&P 500 this ranges from 06-Jan-2010 to 21-Dec-2015. The out of sample period used for predicting one-day ahead estimations ranges from 22-Dec-2015 to 03-Jan-2020 with 1017 observations. For Euro Stoxx 50 the in-sample period is from 18-Dec-2015 up to and including 03-Jan-2020 with also 1500 observations. The out of sample window covers the last 1009 observations. The in-sample estimation period estimates the model parameters based on maximum likelihood estimation, as described in the section on maximum likelihood. Then the obtained parameters are used to forecast the one-day ahead predictions in the out-of-sample. This method of estimation is called fixed-window estimation. This method has been chosen for because it is less computationally demanding than a rolling-window estimation. However, the results obtained are less accurate. In order to examine which model produces the best forecasts for the one-day ahead conditional variance and Value at Risk, several performance measures are computed. In the next section these methods are elaborated upon.

7.1 Mean Absolute Error:

The Mean Absolute Error measures the absolute difference between the predicted value of the conditional variance and its observed conditional variance. Due to the fact that 'observed' volatility is something which is difficult to estimate, therefore it is assumed that in the computation of the MAE the observed conditional variance is equal to the squared daily returns. Obviously, the squared return is not equal to the variance, but to the expectation of the variance. So that better models have a smaller MAE.

$$\text{MAE} = \frac{1}{N}\sum_{t=1}^{n}|\sigma_t^2 - \hat{\sigma}_t^2| \tag{10}$$

7.2 Logarithmic scoring:

Logarithmic Scoring Rule:

$$\text{LSR}(\hat{f}; y_{t+1}) = \ln f(y_{t+1}) \tag{11}$$

Where $f(\cdot)$ is the chosen distribution, and y_{t+1} is the return at time $t+1$.

A way of describing logarithmic scoring is the out-of-sample log likelihood, as to measure the chance of observing the data given the estimated parameters of the models and the distribution of the innovations. The individual log likelihoods per data point between two models are then compared in a Diebold-Mariano test, as explained below.

7.3 Diebold-Mariano test

The Diebold-Mariano test can be used to compare the forecasts performance between different models. First the loss differential is calculated by subtracting the measure of the first model from the measure of the second model at each out of sample time point, the loss differential thus becomes a vector of differences with length equal to the out of sample window. Then the DM-statistic is computed in order to test the null hypothesis whether the mean of the loss differential is significantly different from zero. If the null-hypothesis is rejected then there is evidence that one of the models outperforms the other based on the measure.

$$\text{DM-statistic} = \sqrt{N}\frac{\bar{d} - 0}{\sqrt{Var(d)}} \sim Student - T(N-1) \tag{12}$$

Where:
$H_0 : \bar{d} = 0, \qquad H_1 : \bar{d} \neq 0$
Loss differential: $d_t = Measure_{t,model\ 1} - Measure_{t,model\ 2}$
$Var(d) = \frac{1}{N-1}\sum_{t=1}^{N}(d_t - \bar{d})^2$
$N =$ out of sample length

For the mean absolute error, the loss differential is calculated by subtracting the absolute error between the forecasted conditional variance and the observed variance for both models from each other at each out of sample time point. If the null hypothesis is rejected then there is a significant difference in the MAEs between the two models. The loss differential for MAE is summarized in equation (13) below:

$$d_t = |\sigma_t^2 - \hat{\sigma}_{t,model\ 1}^2| - |\sigma_t^2 - \hat{\sigma}_{t,model\ 2}^2| \tag{13}$$

For the logarithmic scoring rule, the loss differential vector is calculated by subtracting the log-likelihood vector of one model from the log-likelihood vector of the other:

$$d_t = LSR_{t,model\ 1} - LSR_{t,model\ 2} \tag{14}$$

The hypotheses and outcome of the Diebold-Mariano test stays the same with logarithmic scoring rule.

7.4 Value at Risk

The Value at Risk (VaR) is a widely used risk measure in the financial industry. It aims to measure the severity of a loss of an investment with a certain probability, over a specified time period. Formally, the VaR forecast is defined as the following:

$$\hat{VaR}_{T+1}(p) = \mu_t + \hat{\sigma}_{T+1} CDF_R^{-1}(p) \tag{15}$$

Where p is the probability of losses exceeding the VaR, μ_t is the time-varying mean of the returns, $\hat{\sigma}_{T+1}$ is the time-varying conditional volatility estimate, obtained by a certain GARCH-model and CDF_R^{-1} is the inverse of the selected cumulative density function. In this report μ_t is specified to be zero because it is assumed that y_t is generated by an AR(0) process with no intercept.

To assess the quality of the VaR forecasts VaR-violations are defined. A violation has said to occur if the observed loss of a portfolio is smaller than the estimated VaR. In mathematical terms, a violation η_t is defined as:

$$\eta_t = \begin{cases} 1, & \text{if } y_t \leq \text{VaR}_t \\ 0, & \text{if } y_t > \text{VaR}_t \end{cases} \tag{16}$$

Violations can be used to backtest the forecast VaR and compare it to the actual profit-and-losses observed. A good VaR forecast should not have a significantly different number of violations than which can be expected of the theoretical confidence level p that is chosen. Another property that should be satisfied is that the violations should be independently distributed through time, meaning that there is no autocorrelation in the violations.

7.5 Unconditional coverage property:

The unconditional coverage property ensures that the theoretical confidence level p matches the empirical probability of the VaR violations. In this report the Bernoulli coverage test is conducted. In this test η_t is a sequence of Bernoulli-distributed random variables and the VaR violations are compared to these sequence. The null-hypothesis for the VaR violations is given by:

$$H_0 : \eta \sim B(p), \tag{17}$$

where B stands for the Bernoulli distribution. The Logarithmic Ratio (LR) test statistic is given by:

$$\mathrm{LR}_{uc} = 2\left(\log \mathcal{L}_U(\hat{p}) - \log \mathcal{L}_R(p)\right) \overset{\text{asymptotic}}{\sim} \chi^2_{(1)} \tag{18}$$

Further explanation is given in appendix D.

7.6 Independence property:

The independence property requires any two observations in the VaR hit sequence to be independent. In order to determine the independence the probabilities of two consecutive violations and the probability of a violation if there was no violation on the previous day are computed. This can be expressed by:

$$p_{i,j} = Pr(\eta_t = j | \eta_{t1} = i), \tag{19}$$

where i and j are either 0 or 1.

Given the restricted and unrestricted estimated transition matrices, which are explained appendix E, the likelihood ratio test is given by.

$$\mathrm{LR}_{ind} = 2\left(\log \mathcal{L}_U(\hat{\Pi}_1) - \log \mathcal{L}_R(\hat{\Pi}_0)\right) \overset{\text{asymptotic}}{\sim} \chi^2_{(1)} \tag{20}$$

There is however a weakness to the independence test. Namely, the test only takes direct subsequent lags into account and does not take lags into account that are more than one lag apart. An adequate VaR forecast should not only have no autocorrelation between subsequent lags, but also no autocorrelation between lags that are further apart in time.

7.7 Joint test:

By combining the coverage test and the independence test it is possible to jointly test whether VaR violations are significantly different from expected and whether there is clustering of violations by the following test statistic:

$$\mathrm{LR}_{joint} = \mathrm{LR}_{uc} + \mathrm{LR}_{ind} \sim \chi^2_{(2)} \tag{21}$$

The joint test could be used to asses the total quality of the VaR forecast.

8 Results

8.1 Model estimates

The parameter estimates of the models as described in section 4 are displayed in appendix F. All parameters in the appendix are estimated using maximum likelihood estimation and in general most parameters are highly significant. Furthermore, the log likelihood, Akaike information criterion (AIC) and Bayesian information criterion (BIC) of the models with the different distributions are computed. Overall, the skewed-T distribution has the best values on the parameter criteria followed by the student-T distribution and thereafter the normal distribution. This rank order was expected due to the fact that the skewed T distribution contains the most information given the data set. Now that the model parameters are computed it is possible to construct the volatility forecasts, estimate the various value at risk measures and perform backtests.

8.2 In sample model diagnostics

To evaluate whether the conditional volatility models as described in section 3 are well specified and able to capture the characteristics of the returns, an in sample analysis of the standardized residuals is done.The residuals are standardized using the conditional volatility from the most simple GARCH(1,1) model with Normal distribution. The main interest is. if the residuals have mean of zero, a variance of one. as well as no autocorrelation in the residuals and residuals squared. Appendix G figure 4, shows the autocorrelation of the returns and squared returns on the left and on the right the autocorrelation of the standardized residuals and the squared standardized residuals. The autocorrelation of the squared standardized residuals is around zero, which means that the GARCH model is able to capture this property of the volatility well.

8.3 Graphical backtest

In order to evaluate the accuracy of the volatility forecasts and to test the performance of different models the predicted values are compared to the observed out of sample realized volatility. In appendix H figure 5 and 6 show the volatility predictions against the out of sample observed returns. From a graphical point of view the forecasts do not seem to bear any abnormalities like negative volatility values.

8.4 Out of sample analysis using Mean Absolute Error

Table 2 shows the mean absolute errors for each model described in section 4. For the S&P 500, the lowest MAE and best performing model based on this measure is the GJR-GARCH(1,1) Normal. The worst performing model is the EGARCH(1,1) skewed Student-t with an MAE of $6.7120 \cdot 10^{-05}$.Looking at the Euro Stoxx 50 data, it is found that the best model based on MAE is the GJR-GARCH(1,1) with Skewed Student-t distribution and the worst performing model the EGARCH(1,1) with a Student-t distribution for the innovations. When comparing the data sets S&P500 Euro Stoxx 50 with each other it is found that the MAE per model is lower for the S&P 500 data overall, indicating that for this data set the volatility is more accurately predicted based on MAE.

Model	**MAE S&P 500**	**MAE Euro Stoxx 50**
GARCH Normal	$6.5532 \cdot 10^{-05}$	$1.0592 \cdot 10^{-04}$
GARCH Student-t	$6.6278 \cdot 10^{-05}$	$1.0608 \cdot 10^{-04}$
GARCH Skewed Student-t	$6.6537 \cdot 10^{-05}$	$1.0605 \cdot 10^{-04}$
EGARCH Normal	$6.4271 \cdot 10^{-05}$	$1.1064 \cdot 10^{-04}$
EGARCH Student-t	$6.6230 \cdot 10^{-05}$	$1.1065 \cdot 10^{-04}$
EGARCH Skewed Student-t	$6.7120 \cdot 10^{-05}$	$1.1061 \cdot 10^{-04}$
GJR-GARCH Normal	$6.3905 \cdot 10^{-05}$	$1.0654 \cdot 10^{-04}$
GJR-GARCH Student-t	$6.4003 \cdot 10^{-05}$	$1.0597 \cdot 10^{-04}$
GJR-GARCH Skewed Student-t	$6.4533 \cdot 10^{-05}$	$1.0585 \cdot 10^{-04}$

Table 2: Mean absolute errors (MAE) per model.

Appendix I tables 7 and 8 show the comparison of the mean absolute errors between different distributions and models. To evaluate whether the found MAEs are actually statistically different from each other the Diebold-Mariano test is used, where the null hypothesis is that the difference in MAE is not statistically different from zero. Holding the type of model constant, it is found for the S&P 500 data that the Normal (N) distribution significantly outperforms the Student-T (T) and Skewed Student-T (ST) distribution in all instances except for the GJR-GARCH type models. Furthermore, the Student-T (T) distribution shows significantly better results in terms of MAE than the Skewed Student-T (ST) distribution for all cases. Also, keeping the distribution fixed no evidence is found that the GARCH model results in any significant decline in MAE over the EGARCH and GJR-GARCH, however the GJR-GARCH model does show significantly better results than the EGARCH when using the Student-T and Skewed Student-T innovation distributions. Moving on to the results for the Euro Stoxx 50, persistent evidence is found that the GARCH as well as GJR-GARCH models result in significantly lower MAEs than the EGARCH model. There is no significant difference in MAE between GARCH and GJR-GARCH models. Lastly, when comparing distributions only evidence is found that the Skewed Student-T distribution outperforms the standard Student-T for GJR-GARCH models.

8.5 Out of sample analysis using logarithmic scoring rule

Ranking	S&P 500		Euro Stoxx 50	
	model	dmstat	model	dmstat
1	GARCH-T	1689.49469	GARCH-ST	600.1601
2	GARCH-ST	354.677869	GARCH-T	485.7948
3	GJR-T	223.2439151	GJR-ST	367.5836
4	GJR-ST	164.633192	EGARCH-ST	255.0887
5	EGARCH-T	148.443183	EGARCH-T	228.8169
6	EGARCH-ST	109.4487509	GJR-T	191.5931
7	EGARCH-N	-253.92976	GJR-N	-521.3175
8	GJR-N	-363.63631	EGARCH-N	-586.3308
9	GARCH-N	-2072.37553	GARCH-N	-1021.3890

Table 3: A ranking table where dmstat is the summation of the logarithmic scoring rule DM-statistics per model.

Table 3 shows a ranking of the models based on the summation of DM-statistics, which can be seen in appendix J tables 9 and 10. With the S&P 500, one sees that the models using a student-t and skewed student-t perform comparable to each other. They do however significantly outperform models using a normal distribution. Another thing to note is that with exception of the models using a normal distribution, the GARCH performs best, followed by the GJR-GARCH and the EGARCH respectively. The best performing model is the GARCH-T, and the worst performing model is the GARCH-N.The outcome of the ranking using the Euro Stoxx 50 returns yields similar results to the S&P 500. The models using a student-t or skewed student-t perform comparable with each other, and significantly outperform the models using a normal distribution. There is no strong hierarchy in model types but it can be seen that, with exception of the models using a normal distribution, the GARCH performs better than the GJR- and the EGARCH models. The best performing model is the GARCH-ST and the worst performing model is the GARCH-N. In table 3 a ranking is made from the summations of the Diebold-Mariano statistics. The outcomes of these statistics are shown in appendix J, tables 9 and 10.

8.6 Value at Risk violations backtest results

In table 4 and table 5 the observed violations are shown, with their corresponding violation ratios and estimated probabilities. From these tables one can infer that our models systematically overestimate the value at risk. The reason for this is that the violation ratios are not close to one and the estimated probabilities are not close to our selected VaR probability. The number of violations observed for some models is even zero. Because of this one is unable to compute the independence test statistic. These tables however do not show whether the differences between expected and observed are significantly away from zero. This is done in table 6.

Model	Expected Violations	Observed Violations	Violation Ratio	Estimated Probability ($\hat{p}$)
Normal:				
GARCH-N VaR(5%)	51	23	0.45	2.26%
GARCH-N VaR(1%)	10	3	0.30	0.30%
EGARCH-N VaR(5%)	51	10	0.20	0.99%
EGARCH-N VaR(1%)	10	0	0.00	0.00%
GJR-GARCH-N VaR(5%)	51	8	0.16	0.79%
GJR-GARCH-N VaR(1%)	10	0	0.00	0.00%
Student-T:				
GARCH-T VaR(5%)	51	24	0.47	2.36%
GARCH-T VaR(1%)	10	0	0.00	0.00%
EGARCH-T VaR(5%)	51	8	0.16	0.79%
EGARCH-T VaR(1%)	10	0	0.00	0.00%
GJR-GARCH-T VaR(5%)	51	3	0.06	0.30%
GJR-GARCH-T VaR(1%)	10	0	0.00	0.00%
Skewed Student-T:				
GARCH-ST VaR(5%)	51	23	0.45	2.27%
GARCH-ST VaR(1%)	10	0	0.00	0.00%
EGARCH-ST VaR(5%)	51	6	0.12	0.59%
EGARCH-ST VaR(1%)	10	0	0.00	0.00%
GJR-GARCH-ST VaR(5%)	51	0	0.00	0.00%
GJR-GARCH-ST VaR(1%)	10	0	0.00	0.00%

Table 4: Value at Risk violations table for the S&P 500, based on frequentist estimation. Where the expected number of violations is defined as the number of violations expected based on the selected p, the observed violations is the total sum of η_t, violation ratio is the ratio of observed violations and expected violations, and the estimated probability is the probability corresponding to the observed number of violations with the length of the vector η_t.

Model	Expected Violations	Observed Violations	Violation Ratio	Estimated Probability ($\hat{p}$)
Normal:				
GARCH-N VaR(5%)	50	23	0.46	2.28%
GARCH-N VaR(1%)	10	3	0.30	0.30%
EGARCH-N VaR(5%)	50	11	0.22	1.09%
EGARCH-N VaR(1%)	10	1	0.10	0.10%
GJR-GARCH-N VaR(5%)	50	10	0.20	0.99%
GJR-GARCH-N VaR(1%)	10	0	0.00	0.00%
Student-T:				
GARCH-T VaR(5%)	50	30	0.60	2.98%
GARCH-T VaR(1%)	10	1	0.10	0.10%
EGARCH-T VaR(5%)	50	12	0.24	1.19%
EGARCH-T VaR(1%)	10	0	0.00	0.00%
GJR-GARCH-T VaR(5%)	50	9	0.18	0.89%
GJR-GARCH-T VaR(1%)	10	0	0.00	0.00%
Skewed Student-T:				
GARCH-ST VaR(5%)	50	25	0.50	2.48%
GARCH-ST VaR(1%)	10	1	0.10	0.10%
EGARCH-ST VaR(5%)	50	11	0.22	1.09%
EGARCH-ST VaR(1%)	10	0	0.00	0.00%
GJR-GARCH-ST VaR(5%)	50	6	0.12	0.60%
GJR-GARCH-ST VaR(1%)	10	0	0.00	0.00%

Table 5: Value at Risk violations table for Euro Stoxx 50, based on frequentist estimation. Where the expected violations is defined as the amount of violations expected based on the selected p, the observed violations is the total sum of η_t, violation ratio is the ratio of observed violations and expected violations, and the estimated probability is the probability corresponding to the observed amount of violations with the length of the vector η_t.

In table 6 one can see the backtest results regarding the VaR for 5% and 1%, for the S&P 500 and the Euro Stoxx 50 data sets. Interestingly, we find no model that correctly estimates risk. There even are some NaN's shown in table 6, because an independence test can not be calculated when a model has no violations, which can be seen in table 4 and 5. This means that this model systematically overestimates risk with a large amount. The models perform especially poor on the unconditional coverage test, indicating that our selected p does not match our estimated $\hat{p}$. The independence tests on the other hand are not statistically significant for some models. For example VaR(5%) with an EGARCH(1,1) with skewed student-T distributed residuals for the S&P 500 has a statistic of 0.0714 and a corresponding p-value of 0.7893. This means that we do not find a statistically significant relationship regarding the autocorrelation between subsequent violations on a commonly used statistical significance level.
Another interesting insight gained from table 6 is that our models show higher statistics for VaR(1%) in comparison to VaR(5%). This indicates that our models tend to perform worse for estimating VaR(1%) than for VaR(5%). The joint tests are, like the unconditional coverage tests, all highly statistically significant. This is mainly due to the fact that our models violate the unconditional coverage tests. From the joint test we can conclude that our models are not able to correctly estimate the Value at Risk measure for 5% and 1% using a fixed window estimation.

Model	Method	S&P 500-N	Euro Stoxx 50-N	S&P 500-T	Euro Stoxx 50-T	S&P 500-SkewT	Euro Stoxx 50-SkewT
GARCH VaR(5%)	LR_{uc}	19.8859	19.4334	18.2901	10.0618	19.8859	16.3597
		[0.0000]	[0.0000]	[0.0000]	[0.0015]	[0.0000]	[0.0001]
GARCH VaR(1%)	LR_{uc}	7.0376	6.9243	20.4022	13.6031	20.4022	13.6031
		[0.0080]	[0.0085]	[0.0000]	[0.0002]	[0.0000]	[0.0002]
EGARCH VaR(5%)	LR_{uc}	50.7118	46.8326	57.8084	43.7990	65.9236	46.8326
		[0.0000]	[0.0000]	[0.0000]	[0.0000]	[0.0000]	[0.0000]
EGARCH VaR(1%)	LR_{uc}	20.4022	13.6031	20.4022	20.2414	20.4022	20.2414
		[0.0000]	[0.0002]	[0.0000]	[0.0000]	[0.0000]	[0.0000]
GJR-GARCH VaR(5%)	LR_{uc}	57.8084	50.0502	80.8567	53.4702	104.1254	65.1981
		[0.0000]	[0.0000]	[0.0000]	[0.0000]	[0.0000]	[0.0000]
GJR-GARCH VaR(1%)	LR_{uc}	20.4022	20.2414	20.4022	20.2414	20.4022	20.2414
		[0.0000]	[0.0000]	[0.0000]	[0.0000]	[0.0000]	[0.0000]
GARCH VaR(5%)	LR_{ind}	0.3655	1.0764	0.2834	1.8446	0.3655	1.2743
		[0.5455]	[0.2995]	[0.5945]	[0.1744]	[0.5455]	[0.2590]
GARCH VaR(1%)	LR_{ind}	0.0178	0.0179	NaN	0.0020	NaN	0.0020
		[0.8939]	[0.8934]		[0.9644]		[0.9644]
EGARCH VaR(5%)	LR_{ind}	3.0004	0.2432	3.8831	0.2897	0.0714	0.2432
		[0.0832]	[0.6219]	[0.0488]	[0.5904]	[0.7893]	[0.6219]
EGARCH VaR(1%)	LR_{ind}	NaN	0.0020	NaN	NaN	NaN	NaN
			[0.9644]				
GJR-GARCH VaR(5%)	LR_{ind}	0.1272	0.2008	0.0178	0.1625	NaN	0.0720
		[0.7213]	[0.6541]	[0.8939]	[0.6869]		[0.7884]
GJR-GARCH VaR(1%)	LR_{ind}	NaN	NaN	NaN	NaN	NaN	NaN
GARCH VaR(5%)	LR_{joint}	20.2513	20.5098	18.5736	11.9063	20.2513	17.6341
		[0.0000]	[0.0001]	[0.0026]	[0.0000]	[0.0000]	[0.0000]
GARCH VaR(1%)	LR_{joint}	7.0554	6.9423	NaN	13.6051	NaN	13.6051
		[0.0311]		[0.0011]	[0.0000]	[0.0000]	[0.0000]
EGARCH VaR(5%)	LR_{joint}	53.7122	47.0758	61.6915	44.0888	65.9950	47.0758
		[0.0000]	[0.0000]	[0.0000]	[0.0000]	[0.0000]	[0.0000]
EGARCH VaR(1%)	LR_{joint}	NaN	13.6051	NaN	NaN	NaN	NaN
		[0.0011]					
GJR-GARCH VaR(5%)	LR_{joint}	57.9356	50.2510	80.8745	53.6327	NaN	65.2701
		[0.0000]	[0.0000]	[0.0000]	[0.0000]	[0.0000]	[0.0000]
GJR-GARCH VaR(1%)	LR_{joint}	NaN	NaN	NaN	NaN	NaN	NaN

Table 6: Value at Risk (VaR) backtests, for 5% and 1%. The p-value of the likelihood ratio test statistic is in brackets underneath the corresponding statistic. Where LR_{uc} stands for the LR statistic of the unconditional coverage test, LR_{ind} is the LR statistic of the independence test and LR_{joint} refers to the joint test statistic. P-values are calculated based on their corresponding degrees of freedom using a chi-square survival function. The columns show the corresponding data set and the distribution used for the residuals.

9 Alternative Approach: Bayesian Estimation

In statistics there are two different groups which oppose each other in the manner how they estimate parameters. Namely, the group of frequentists and the group of Bayesians. The frequentist estimates parameters according to the following formula:

$$f(y|\phi) \tag{22}$$

For the frequentist approach, the observed values of y are unknown and random and ϕ is a fixed value. On the other hand, the Bayesian method involves estimating the parameters according to the inverse probability $p(\phi|y)$ and its posterior distribution which is given by the following formula:

$$p(\phi|y) = \frac{p(y|\phi)p(\phi)}{\int p(\phi)p(y|\phi)d\phi} \tag{23}$$

In the Bayesian approach the data y are fixed, in contrast to the frequentist approach, where the data is random. The model parameters, given by ϕ, themselves may not be random but the Bayesian approach uses probability distributions to describe the uncertainty in parameter values and are therefore treated as random. In this paper the GARCH(1,1) with Student-t innovations are estimated by Bayesian methods. The main source of inspiration is given by an article written by Ardia and Hoogerheide (2006). According to those two researchers, Bayesian estimation offers an attractive alternative for maximum likelihood estimation which enables small sample results, robust estimation, model discrimination, model combination, and probabilistic statements on nonlinear functions of the parameters of the model. For the sake of accuracy and the lowest possible standard errors, 10 Markov Chain Monte Carlo, hereafter MCMC, chains with a iteration length of 100.000 for each

chain are run in R for the two indices. The vector of starting values of the chains are left unchanged, namely the default setting. The log-returns for this estimation is defined by:

$$y_t = 100(\log(P_t) - \log(P_{t-1}) \tag{24}$$

After the construction of the MCMC chains a sample was generated and from there the parameters estimations were obtained, outcome displayed in appendix K. Moreover, in the same appendix descriptive statistics of the 2-million simulations are given to offer more informative insights in the simulation structures. In order to compare the precision of the newly obtained parameters the volatility of the stock indices is once again forecast, based on these parameter values and thereafter backtesting procedures are applied.

Firstly, MAEs are computed, see section 7.1 for theory about the concept, but now for the parameters obtained by Bayesian estimation. The MAE, given the Bayesian GARCH Student-t, for the S&P500 index is $6.5852 \cdot 10^{-05}$ and for the Euro Stoxx 50 index the MAE is $8.1863 \cdot 10^{-05}$. Compared to the MAEs of the models of which the parameters are obtained by maximum likelihood estimation, which is $6.6278 \cdot 10^{-05}$ for S&P500 and $1.0608 \cdot 10^{-04}$ for Euro Stoxx 50, the MAEs of the Bayesian estimation forecast are in both case lower and therefore more accurate.

Out of sample log likelihoods were also calculated and compared to other models using the Diebold-Mariano test. In appendix K table 13 the results of these DM-tests are shown. One can see that the model estimated using the Bayesian method significantly outperforms the models estimated using the frequentist method.

Moreover, comparing the backtest result of the Bayesian estimation method, included in appendix L, it is possible to conclude that the Bayesian estimation method does have similar results as maximum likelihood estimation. Which do not indicates superior results of the Bayesian estimation methods on Value at Risk forecasting.

10 Conclusions

Beforehand, we would like to acknowledge that we experienced the case study to be educational given our non-econometric backgrounds. In the beginning of this report a solid theoretical base has been formed. Firstly, the model specifications of the GARCH(1,1) model, E-GARCH(1,1) model, and the GJR-GARCH(1,1) model have been explained. Secondly, maximum likelihood estimation has been explained. Thirdly, innovations distributions such as the normal distribution, student-t distribution, and the skewed student-t distribution are elucidated. Fourthly, the performances measures such as Mean Absolute Error, logarithmic scoring, the Diebold Mariono test, Value at Risk, unconditional coverage property, independence test, and the joint test were explained. In the process of the gathering of all of this information we did already learn a lot of new and interesting concepts.

After computing the tests the following conclusions were drawn:

Based on the log-likelihood, the Akaike Information Criterion, and the Bayesian Information Criterion, the skewed student-t distribution does have the highest performance followed by the student-t distribution and lastly the normal distribution.

Based on the Mean Absolute Error of the S&P 500 data set it is possible to conclude that for the S&P 500 the distributional assumption is more important. The ranking for the S&P 500 is, firstly the normal distribution, secondly the student-t distribution, and thirdly, the skewed student-t distribution. For the Euro Stoxx 50 data set, it is possible to conclude that the model type is more important than the distributional assumption. The ranking for the Euro Stoxx 50 is, shared first and second place the GARCH and GJR-GARCH models and third place, significantly worst, the E-GARCH model.

Based on the logarithmic scoring rule it is possible to conclude that for the S&P 500 the distributional assumption is more important than the model type. The GARCH student-t distribution does have the highest LSR, followed by GARCH skewed student-t distribution, and thereafter th GJR student-t distribution. Moreover, the S&P 500 data shows that the Skewed student-t distribution and student-t distribution are comparable but also significantly better than the normal distribution. Based on the data of the Euro Stoxx 50 it is possible to conclude that, contrary to the results from the MAE-test, that the distributional assumption is more important than the model type. The ranking is as follows: firstly, the GARCH skewed student-t, secondly, the GARCH student-t, and thirdly the GJR skewed student-t.

An explanation for the fact that the GARCH model wins from the models that can capture a leverage effect is perhaps a structural change in the leverage effect between the in-sample and out-of-sample periods. To test this, a table was made, shown in appendix N, table 17, that shows the $\hat{\gamma}$'s estimated for the in-sample and out-of-sample periods separately for both the S&P 500 and the Euro Stoxx 50. In this table one can see that most of the $\hat{\gamma}$'s, with exception of the GJR estimates, do in fact differ with between the in-sample and out-of-sample period.

Given the computed Value at Risk calculations and the several backtests performed it is possible to conclude that the estimated models highly overestimate risks. After computing the summary statistics of the in-sample and the out-sample data, we find a reasonable explanation for the overestimating of risk. Namely, our in-sample period is more volatility than our out-sample period. This results in parameters that are estimated on a more volatile period, which causes overestimating of risk.

For the models based on Bayesian methods it possible to conclude that the estimated parameters does have superior precision over maximum likelihood resulting in lower mean absolute errors, higher logarithmic scoring rule scores, and lower standard errors. However, looking at the Value at Risk violation ratios the Bayesian estimated models does not necessarily returns preciser estimates.

Based on this report the following suggestions for further research are acknowledged. Firstly, it is recommended to implement a rolling window regime in the estimation of the parameters, which will result in more accurate estimates. To elaborate, the turbulent financial times in the beginning of the data set, namely the in-sample data, resulted in the overestimation of the volatility in the later stage, namely the out-of-sample data, and this estimation error could possibly be fixed by a rolling window. Secondly, it is recommend to use, besides a rolling window, a longer in-sample estimation window. This is due to fact that the ν or degrees of freedom parameter is estimated based on rare events, and this can possibly ruin the precision of the estimation results of ν. Thirdly, for the Bayesian estimation implemented in this report only the GARCH(1,1) with student-t innovations are reviewed, which provided superior parameter estimations over the estimations based on maximum likelihood. A suggestion for further research would be to run Bayesian estimations for the other models and distributions to compare the precision of Bayesian estimation against maximum likelihood estimation. Moreover for Bayesian estimation, model combinations, such as Bayesian Model Averaging, could be applied, Fourthly, in this report the "standard" models of order (1,1) are estimated, a interesting follow-up question for further research would be what will happen with the precision of the models if higher order-models are implemented. Lastly, the robustness of the conclusions drawn in this report are improved due to the fact that the tests are based on two indices instead of merely one. However, to improve the robustness of the results the test could be run on more indices or data-sets.

11 Appendix

11.1 Appendix A - Plots of the Prices Returns:

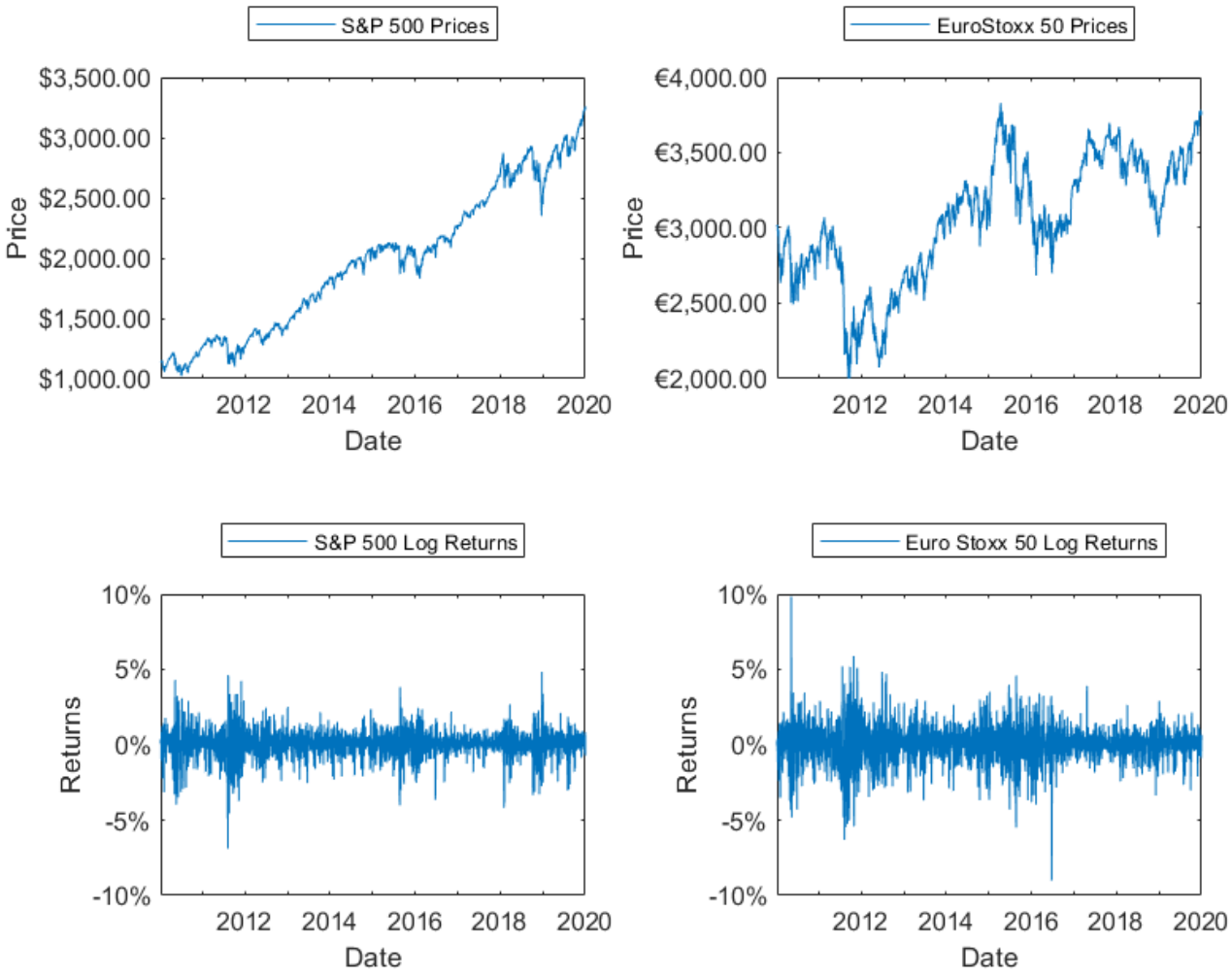


Figure 1: Plot of the adjusted close price of the S&P 500 and the Euro Stoxx 50 index, indicated by prices, and the logarithmic returns of the respective indices.

11.2 Appendix B - Auto Correlation Function Plots:

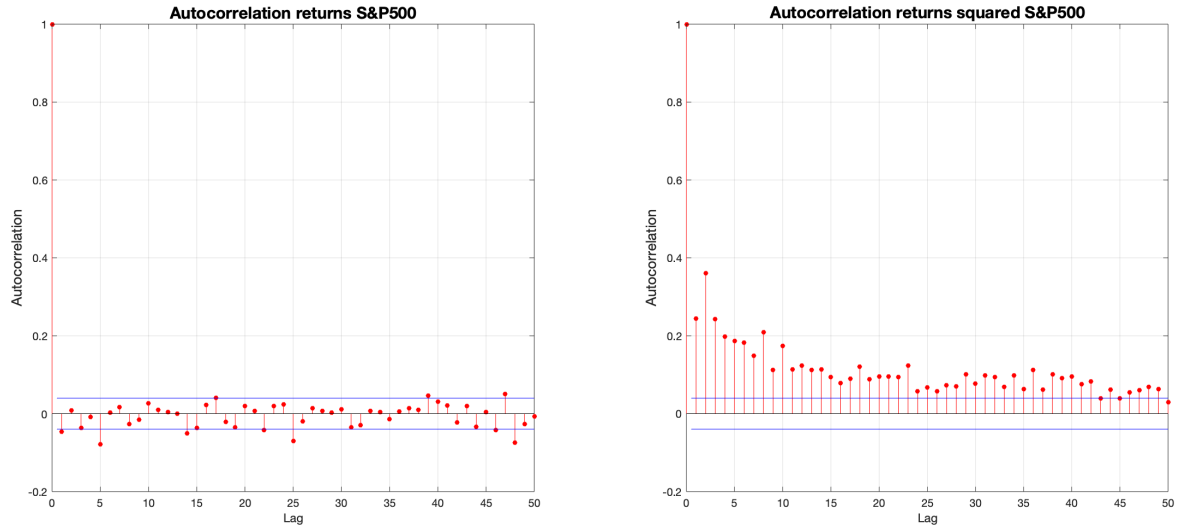


Figure 2: Plot of the auto correlations of the returns and squared returns up to and including 50 lags for the S&P500 data.

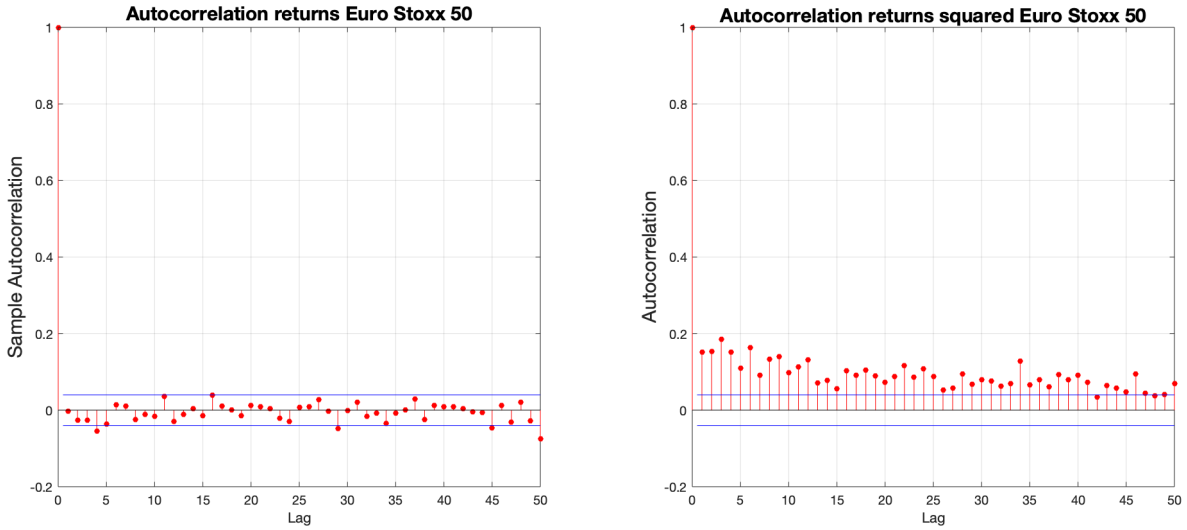
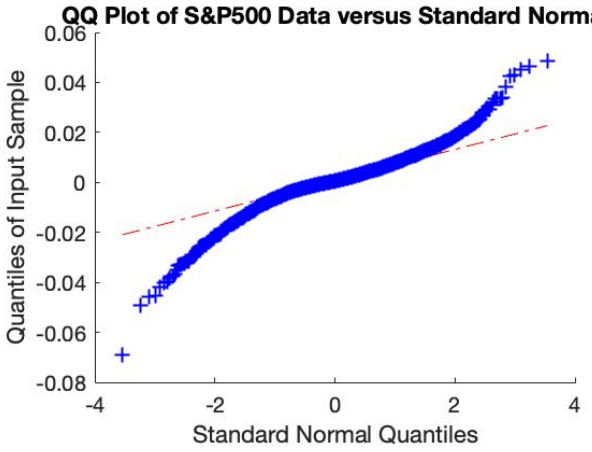
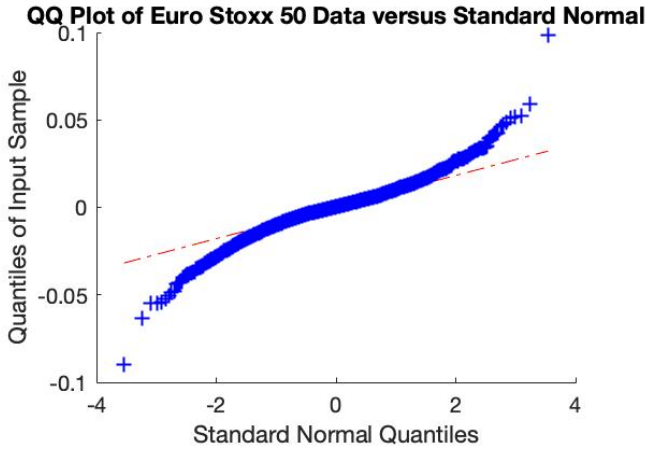


Figure 3: Plot of the auto correlations of the returns and squared returns up to and including 50 lags for the Euro Stoxx 50 data.

11.3 Appendix C - Quantile to Quantile Plots:





11.4 Appendix D - Full Bernoulli Density Explanation:

The Bernoulli probability mass function is given by:

$$(1-p)^{1-\eta_t}(p)^{\eta_t}, \eta_t = 0, 1. \tag{25}$$

The probability p can be estimated by:

$$\hat{p} = \frac{v_1}{W_T} \tag{26}$$

The unrestricted likelihood function is given by:

$$\mathcal{L}_{\mathcal{U}}(\hat{p}) = \prod_{t=W_E+1}^{T} (1-\hat{p})^{1-\eta_t}(\hat{p})^{\eta_t} = (1-\hat{p})^{v_0}\hat{p}^{v_1} \tag{27}$$

The restricted likelihood function is given by:

$$\mathcal{L}_{\mathcal{R}}(p) = \prod_{t=W_E+1}^{T} (1-p)^{1-\eta_t}(p)^{\eta_t} = (1-p)^{v_0}(p)^{v_1} \tag{28}$$

To test whether $\mathcal{L}_{\mathcal{R}} = \mathcal{L}_{\mathcal{U}}$ an LR Test is conducted:

$$LR = 2(\log \mathcal{L}_{\mathcal{U}}(\hat{p} - \log \mathcal{L}_{\mathcal{R}}(p)) \tag{29}$$

$$= 2\log \frac{(1-\hat{p})^{v_0}(\hat{p})^{v_1}}{(1-p)^{v_0}(p)^{v_1}} \overset{\text{asymptotic}}{\sim} \chi^2_{(1)}$$

11.5 Appendix E - Full Probability Matrices Explanation:

The first-order transition probability matrix is:

$$\Pi_1 = \begin{pmatrix} 1-p_{01} & p_{01} \\ 1-p_{11} & p_{11} \end{pmatrix} \tag{30}$$

The unrestricted likelihood function is given by:

$$\mathcal{L}_{\mathcal{U}}(\Pi_1) = (1-p_{01})^{v_{00}} p_{01}^{v_{01}} (1-p_{11})^{v_{10}} p_{11}^{v_{11}} \tag{31}$$

Where v_{ij} is the number of observations where j follows i.

The Maximum likelihood (ML) estimates are obtained by maximizing $\mathcal{L}_{\mathcal{U}}(\hat{\Pi}_1)$:

$$\hat{\Pi}_1 = \begin{pmatrix} \frac{v_{00}}{v_{00}+v_{01}} & \frac{v_{01}}{v_{00}+v_{01}} \\ \frac{v_{10}}{v_{10}+v_{11}} & \frac{v_{11}}{v_{10}+v_{11}} \end{pmatrix} \tag{32}$$

Given that there is no volatility clustering under H_0, $p_{01} = p_{11} = p$ the estimated transition matrix simplifies to:

$$\hat{\Pi}_0 = \begin{pmatrix} 1-\hat{p} & \hat{p} \\ 1-\hat{p} & \hat{p} \end{pmatrix} \tag{33}$$

where:

$$\hat{p} = \frac{v_{01} + v_{11}}{v_{00} + v_{10} + v_{01} + v_{11}}$$

The restricted likelihood function under the null hypothesis is:

$$\mathcal{L}_{\mathcal{R}}(\hat{\Pi}_0) = (1-\hat{p})^{v_{00}+v_{10}} \hat{p}^{v_{01}+v_{11}} \tag{34}$$

11.6 Appendix F - Parameter Estimation:

Estimated parameters for the GARCH(1,1) models of the S&P500 and Euro Stoxx 50

Index	S&P500			Euro 50		
	Normal	Student-T	Skewed T	Normal	Student-T	Skewed T
ω	3.8640e-06***	3.7990e-06***	3.6959e-06***	5.3047e-06**	4.5586e-06**	4.4271e-06**
(SD)	(9.536e-07)	(9.882e-07)	(9.939e-07)	(2.367e-06)	(1.797e-06)	(1.775e-06)
α	0.1351***	0.1367***	0.1396***	0.0871***	0.0851***	0.0840***
(SD)	(2.492e-02)	(2.519e-02)	(2.562e-02)	(2.423e-02)	(1.887e-02)	(1.888e-02)
β	0.8260	0.8296***	0.8307 ***	0.8862**	0.8940***	0.8958***
(SD)	(2.628e-02)	(2.614e-02)	(2.657e-02)	(3.114e-02)	(2.247e-02)	(2.260e-02)
ν		6.2346***	6.3214***		6.9098***	6.9295***
(SD)		(0.972)	(1.023)		(1.164)	(1.182)
λ			-0.1405***			-0.0663***
(SD)			(2.779e-02)			(2.956e-02)
Likelihood	-1937.4929	-1915.1234	-1905.0280	-2512.9997	-2490.3524	-2488.2422
AIC	3880.9858	3838.2468	3820.0561	5031.9994	4988.7048	4986.4843
BIC	3898.4771	3861.5685	3849.2082	5049.4811	5012.0138	5015.6205

Note: *p<0.1; **p<0.05; ***p<0.01

Estimated parameters for the GJR(1,1) models of the S&P500 and Euro Stoxx 50

Index	S&P500			Euro 50		
	Normal	Student-T	Skewed T	Normal	Student-T	Skewed T
ω	4.2284e-06***	4.3022e-06***	4.3045e-06***	6.6151e-06**	5.9905e-06**	5.9112e-06**
(SD)	(8.669e-07)	(9.399e-07	(9.524e-07)	(2.987e-06)	(2.338e-06)	(2.406e-06)
α	0.0000	8.6975e-11	0.0000	6.1692e-10	4.7806e-10	0.0000
(SD)	(4.901e-02)	(6.697e-02)	(6.474e-02)	(2.248e-02)	(2.510e-02)	(2.566e-0)
β	0.8227***	0.8075***	0.8058***	0.8662***	0.8640***	0.8646***
(SD)	(3.906e-02)	(5.107e-02)	(5.073e-02)	(4.695e-02)	(4.136e-02)	(4.347e-02)
γ	0.2806***	0.3331***	0.3473***	0.2040***	0.2237***	0.2231***
(SD)	(5.376e-02)	(5.888e-02)	(5.939e-02)	(5.669e-02)	(5.236e-02)	(5.341e-02)
ν		7.2259***	7.2089***		8.1852***	8.3536***
(SD)		(1.552)	(1.594)		(1.610)	(1.677)
λ			-0.1768***			-0.0807**
(SD)			(3.293e-02)			(3.291e-02)
Likelihood	-1885.9074	-1867.7766	-1853.6656	-2468.6521	-2450.8118	-2448.0230
AIC	3779.8148	3745.5532	3719.3313	4945.3042	4911.6235	4908.0560
BIC	3803.1365	3774.7053	3754.3138	4968.6131	4940.7597	4943.0234

Note: *p<0.1; **p<0.05; ***p<0.01

Estimated parameters for E-GARCH(1,1) models of the S&P500 and Euro Stoxx 50

Index	S&P500			Euro 50		
	Normal	Student-T	Skewed T	Normal	Student-T	Skewed T
ω	-8.1909e-07	-7.5619e-07	-5.2340e-07	1.9282e-06 *	1.9131e-06**	1.8537e-06**
(SD)	(6.442e-07)	(6.876e-07)	(3.938e-08)	(9.918e-07)	(8.002e-07)	(7.839e-07)
α	0.1442***	0.1273***	0.1234***	0.0943**	0.1017***	0.1004***
(SD)	(2.330e-02)	(2.439e-02)	(2.360e-02)	(3.111e-02)	(2.518e-02)	(2.485e-02)
β	0.9361***	0.9330***	0.9355	0.9580***	0.9591***	0.9597***
(SD)	(1.234e-02)	(1.461e-02)	(1.465e-02)	(2.046e-02)	(1.536e-02)	(1.535e-02)
γ	-0.2479***	-0.2956***	-0.2980***	-0.2062***	-0.21574***	-0.2168***
(SD)	(3.340e-02)	(3.793e-02)	(3.750e-02)	(4.729e-02)	(3.866e-02)	(3.913e-02)
ν		7.0145***	7.0262***		8.9407***	9.2880***
(SD)		(1.167)	(1.188)		(1.825)	(1.964)
λ			-0.1902			-0.0968***
(SD)			(3.338e-02)			(3.439e-02)
Likelihood	-1875.5865	-1853.9133	-1838.2045	-2452.5359	-2437.4980	-2433.6725
AIC	3759.1731	3717.8267	3688.4091	4913.0718	4884.9960	4879.3450
BIC	3782.4948	3746.9788	3723.3917	4936.3808	4914.1322	4914.3084

Note: *p<0.1; **p<0.05; ***p<0.01

11.7 Appendix G - Auto Correlation in the Standardized Residuals:

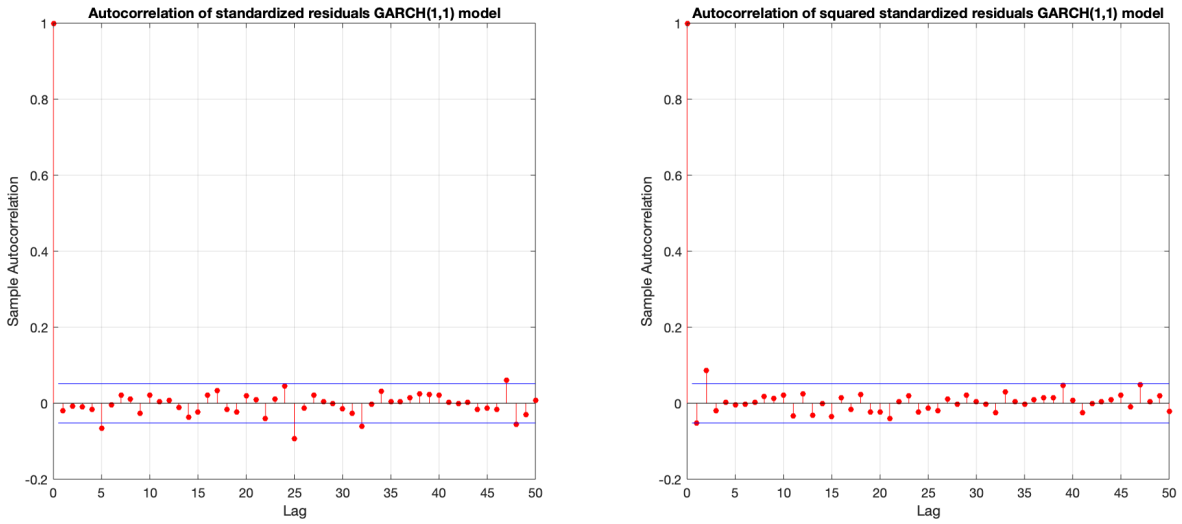


Figure 4: S&P 500 autocorrelation up to and including 50 lagged values for the in sample standardized residuals and standardized residuals squared using GARCH(1,1) Normal volatility model.

11.8 Appendix H - Graphical Backtest:

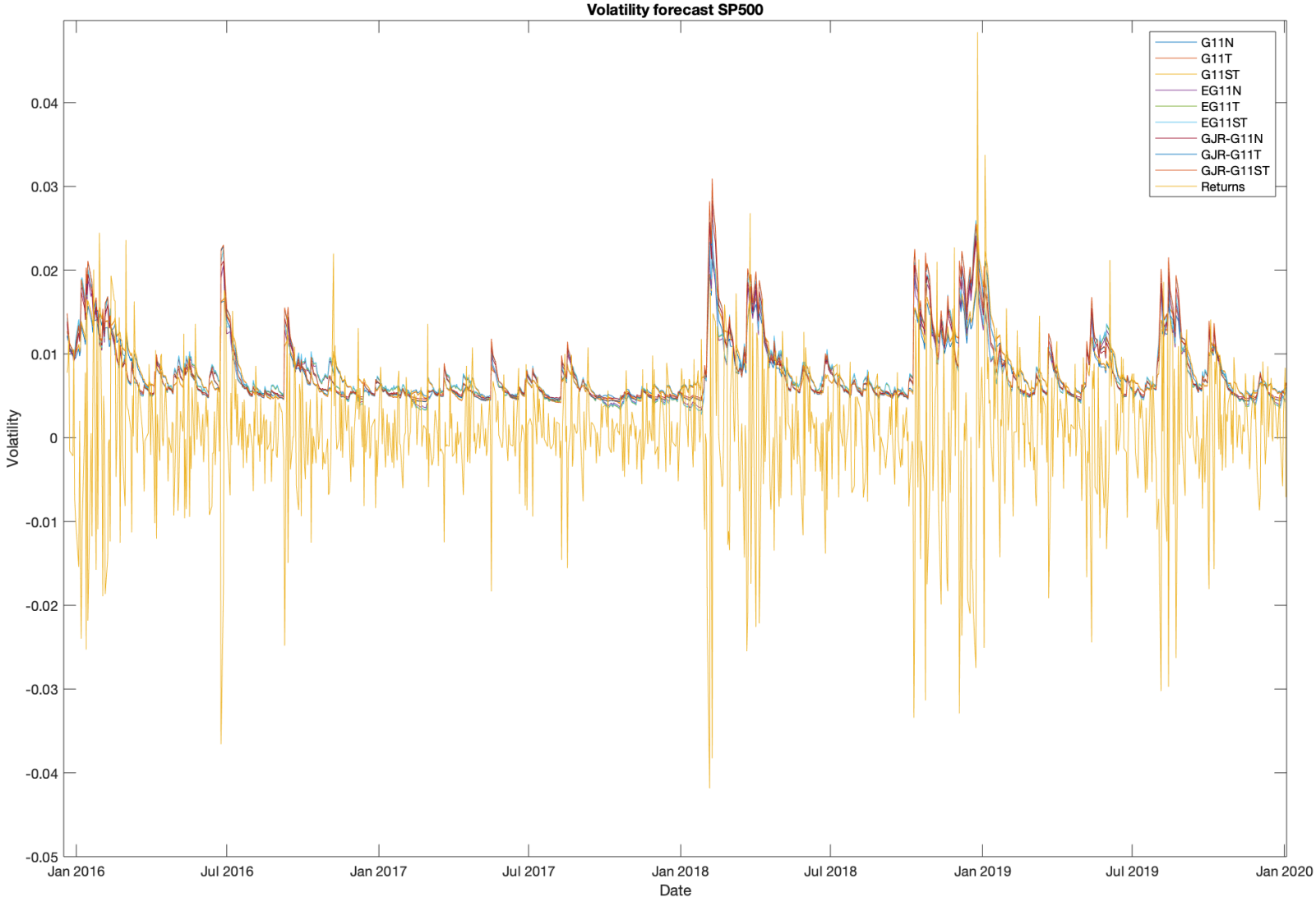


Figure 5: Graphical representation of volatility forecasts SP500 per model SP500

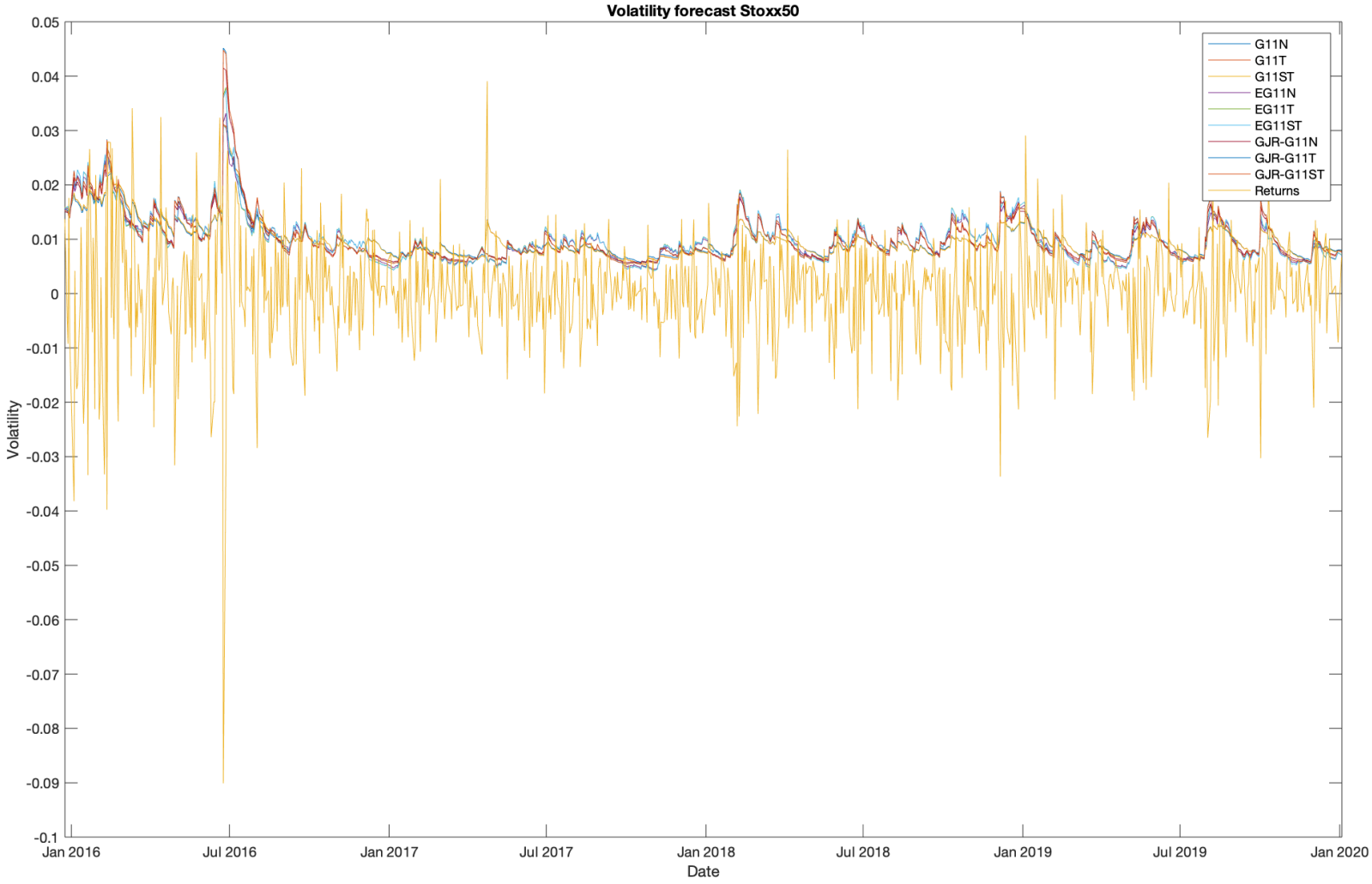


Figure 6: Graphical representation of volatility forecasts per model Stoxx50

11.9 Appendix I - Mean Absolute Error Tables:

Model	GARCH-N	GARCH-T	GARCH-ST	EGARCH-N	EGARCH-T	EGARCH-ST	GJR-GARCH-N	GJR-GARCH-T	GJR-GARCH-ST
GARCH-N	X	-9.4398	-7.2339	1.0340	-0.4631	-1.0286	1.3020	0.9793	0.6009
		0.0000	0.0000	0.2336	0.3583	0.2349	0.1709	0.2469	0.3329
GARCH-T	9.4398	X	-4.2105	1.6426	0.0320	-0.5483	1.9321	1.4817	1.0675
	0.0000		0.0001	0.1035	0.3986	0.3431	0.0618	0.1331	0.2255
GARCH-ST	7.2339	4.2105	X	1.8520	0.2057	-0.3808	2.1777	1.6771	1.2461
	0.0000	0.0001		0.0719	0.3905	0.3709	0.0374	0.0978	0.1835
EGARCH-N	-1.0340	-1.6426	-1.8520	X	-5.0023	-6.2223	0.4984	0.2736	-0.2419
	0.2336	0.1035	0.0719		0.0000	0.0000	0.3522	0.3842	0.3873
EGARCH-T	0.4631	-0.0320	-0.2057	5.0023	X	-9.8145	3.0020	2.5368	1.7915
	0.3583	0.3986	0.3905	0.0000		0.0000	0.0045	0.0161	0.0802
EGARCH-ST	1.0286	0.5483	0.3808	6.2223	9.8145	X	4.0674	3.5908	2.7856
	0.2349	0.3431	0.3709	0.0000	0.0000		0.0001	0.0007	0.0083
GJR-GARCH-N	-1.3020	-1.9321	-2.1777	-0.4984	-3.0020	-4.0674	X	-0.2519	-1.2229
	0.1709	0.0618	0.0374	0.3522	0.0045	0.0001		0.3864	0.1888
GJR-GARCH-T	-0.9793	-1.4817	-1.6771	-0.2736	-2.5368	-3.5908	0.2519	X	-4.0752
	0.2469	0.1331	0.0978	0.3842	0.0161	0.0007	0.3864		0.0001
GJR-GARCH-ST	-0.6009	-1.0675	-1.2461	0.2419	-1.7915	-2.7856	1.2229	4.0752	X
	0.3329	0.2255	0.1835	0.3873	0.0802	0.0083	0.1888	0.0001	

Table 7: MAE model comparison using the S&P 500 data. A positive (negative) value means that the row (column) model has a lower MAE. A colored cell means that the DM-statistic is significantly different from zero at a 5% level. A green (red) cell means that the row (column) has a lower MAE.

Model	GARCH-N	GARCH-T	GARCH-ST	EGARCH-N	EGARCH-T	EGARCH-ST	GJR-GARCH-N	GJR-GARCH-T	GJR-GARCH-ST
GARCH-N	X	-1.2718	-0.9361	-2.4679	-2.2223	-2.1910	-0.3059	-0.0199	0.0293
		0.1776	0.2573	0.0191	0.0339	0.0363	0.3806	0.3988	0.3987
GARCH-T	1.2718	X	1.0361	-2.3674	-2.1398	-2.1098	-0.2306	0.0447	0.0941
	0.1776		0.2331	0.0243	0.0405	0.0432	0.3884	0.3984	0.3971
GARCH-ST	0.9361	-1.0361	X	-2.3672	-2.1397	-2.1100	-0.2412	0.0339	0.0830
	0.2573	0.2331		0.0243	0.0405	0.0432	0.3874	0.3986	0.3975
EGARCH-N	2.4679	2.3674	2.3672	X	-0.0492	0.0619	2.8008	2.7567	2.8260
	0.0191	0.0243	0.0243		0.3984	0.3981	0.0080	0.0090	0.0074
EGARCH-T	2.2223	2.1398	2.1397	0.0492	X	1.7918	3.1549	3.1859	3.2644
	0.0339	0.0405	0.0405	0.3984		0.0802	0.0028	0.0025	0.0020
EGARCH-ST	2.1910	2.1098	2.1100	-0.0619	-1.7918	X	3.1168	3.1603	3.2391
	0.0363	0.0432	0.0432	0.3981	0.0802		0.0032	0.0028	0.0021
GJR-GARCH-N	0.3059	0.2306	0.2412	-2.8008	-3.1549	-3.1168	X	1.5140	1.8296
	0.3806	0.3884	0.3874	0.0080	0.0028	0.0032		0.1268	0.0749
GJR-GARCH-T	0.0199	-0.0447	-0.0339	-2.7567	-3.1859	-3.1603	-1.5140	X	9.4736
	0.3988	0.3984	0.3986	0.0090	0.0025	0.0028	0.1268		0.0000
GJR-GARCH-ST	-0.0293	-0.0941	-0.0830	-2.8260	-3.2644	-3.2391	-1.8296	-9.4736	X
	0.3987	0.3971	0.3975	0.0074	0.0020	0.0021	0.0749	0.0000	

Table 8: MAE model comparison using the S&P 500 data. A positive (negative) value means that the row (column) model has a lower MAE. A colored cell means that the DM-statistic is significantly different from zero at a 5% level. A green (red) cell means that the row (column) has a lower MAE.

11.10 Appendix J - Logarithmic Scoring Rule Model Comparison Tables:

Model	GARCH-N	GARCH-T	GARCH-ST	EGARCH-N	EGARCH-T	EGARCH-ST	GJRGARCH-N	GJRGARCH-T	GJRGARCH-ST
GARCH-N	X	-1659.1	-305.249	-5.31083	-20.3723	-18.9096	-5.2642	-30.1095	-28.0601
		[0.0000]	[0.0000]	[0.0000]	[0.0000]	[0.0000]	[0.0000]	[0.0000]	[0.0000]
GARCH-T	1659.1000	X	-7.56692	17.2024	-1.39459	-1.10321	27.049	-1.7919	-2.0001
	[0.0000]		[0.0000]	[0.0000]	[0.1509]	[0.2171]	[0.0000]	[0.0801]	[0.0540]
GARCH-ST	305.2490	7.5669	X	17.7558	-0.942084	-0.680687	28.2889	-1.14297	-1.4170
	[0.0000]	[0.0000]		[0.0000]	[0.2569]	[0.3164]	[0.0000]	[0.2076]	[0.1462]
EGARCH-N	5.3108	-17.2024	-17.7558	X	-92.8293	-64.2556	3.15161	-35.679	-34.6701
	[0.0000]	[0.0000]	[0.0000]		[0.0000]	[0.0000]	[0.0028]	[0.0000]	[0.0000]
EGARCH-T	20.3723	1.39459	0.942084	92.8293	X	2.23282	30.4422	0.346653	-0.116764
	[0.0000]	[0.1509]	[0.2560]	[0.0000]		[0.0330]	[0.0000]	[0.3757]	[0.3962]
EGARCH-ST	18.9096	1.10321	0.680687	64.2556	-2.2328	X	27.2593	-0.0485581	-0.478268
	[0.0000]	[0.2171]	[0.3164]	[0.0000]	[0.0330]		[0.0000]	[0.3985]	[0.3558]
GJRGARCH-N	5.2642	-27.0490	-28.2889	-3.1516	-30.4422	-27.2593	X	-158.002	-94.7075
	[0.0000]	[0.0000]	[0.0000]	[0.0028]	[0.0000]	[0.0000]		[0.0000]	[0.0000]
GJRGARCH-T	30.1095	1.7919	1.14297	35.6790	-0.3467	0.0485581	158.0020	X	-3.18336
	[0.0000]	[0.0801]	[0.2076]	[0.0000]	[0.3757]	[0.3985]	[0.0000]		[0.0025]
GJRGARCH-ST	28.0601	2.0001	1.41701	34.6701	0.116764	0.4783	94.7075	3.1834	X
	[0.0000]	[0.0540]	[0.1462]	[0.0000]	[0.3962]	[0.3558]	[0.0000]	[0.0025]	

Table 9: Logarithmic scoring rule model comparison using the S&P 500. A positive (negative) value means that the row (column) model has a higher likelihood. A colored cell means that the DM-statistic is significantly different from zero at a 5% level. A green (red) cell means that the row (column) has a higher likelihood.

Model	GARCH-N	GARCH-T	GARCH-ST	EGARCH-N	EGARCH-T	EGARCH-ST	GJRGARCH-N	GJRGARCH-T	GJRGARCH-ST
GARCH-N	X	-550.869	-379.63	1.55464	-13.7526	-17.6898	-0.756697	-27.3497	-32.8958
		[0.0000]	[0.0000]	[0.1191]	[0.0000]	[0.0000]	[0.2996]	[0.0000]	[0.0000]
GARCH-T	550.8690	X	-125.523	21.9003	6.1049	0.971157	32.1179	3.20335	-3.84879
	[0.0000]		[0.0000]	[0.0000]	[0.0000]	[0.2489]	[0.0000]	[0.0024]	[0.0002]
GARCH-ST	379.6300	125.5230	X	26.584	10.6371	5.22012	39.722	10.1434	2.70049
	[0.0000]	[0.0000]		[0.0000]	[0.0000]	[0.0002]	[0.0000]	[0.0000]	[0.0104]
EGARCH-N	-1.55464	-21.9003	-26.5840	X	-294.505	-157.063	-3.50635	-35.9335	-45.284
	[0.1191]	[0.0000]	[0.0000]		[0.0000]	[0.0000]	[0.0009]	[0.0000]	[0.0000]
EGARCH-T	13.7526	-6.1049	-10.6371	294.5050	X	-62.0355	23.181	-7.38942	-16.4548
	[0.0000]	[0.0000]	[0.0000]	[0.0000]		[0.0000]	[0.0000]	[0.0000]	[0.0000]
EGARCH-ST	17.6898	-0.971157	-5.2201	157.0630	62.0355	X	29.0736	1.79134	-6.37326
	[0.0000]	[0.2489]	[0.0000]	[0.0000]	[0.0000]		[0.0000]	[0.0802]	[0.0000]
GJRGARCH-N	0.756697	-32.1179	-39.7220	3.5064	-23.1810	-29.0736	X	-227.857	-173.629
	[0.2996]	[0.0000]	[0.0000]	[0.0000]	[0.0000]	[0.0000]		[0.0000]	[0.0009]
GJRGARCH-T	27.3497	-3.2034	-10.1434	35.9335	7.38942	-1.79134	227.8570	X	-91.7984
	[0.0000]	[0.0024]	[0.0000]	[0.0000]	[0.0000]	[0.0802]	[0.0000]		[0.0000]
GJRGARCH-ST	32.8958	3.8488	-2.7005	45.2840	16.4548	6.3733	173.6290	91.7984	X
	[0.0000]	[0.0002]	[0.0104]	[0.0000]	[0.0000]	[0.0009]	[0.0000]	[0.0000]	

Table 10: Logarithmic scoring rule model comparison Euro Stoxx 50. A positive (negative) value means that the row (column) model has a higher likelihood. A colored cell means that the DM-statistic is significantly different from zero at a 5% level. A green (red) cell means that the row (column) has a higher likelihood.

11.11 Appendix K - Bayesian Estimation Simulation Analysis:

	$\phi_{posterior}$	C.I. 0.025	C.I. 0.975
ω	0.0488	0.02922	0.07439
α	0.1556	0.10990	0.21223
β	0.8030	0.74670	0.85144
ν	6.7707	4.62410	10.03515

Table 11: Posterior Mean of GARCH(1,1) with Student-t Innovations including the left and right quantile for the S&p500

	$\phi_{posterior}$	C.I. 0.025	C.I. 0.975
ω	0.07714	0.04065	0.1295
α	0.10801	0.07294	0.1528
β	0.85755	0.80378	0.8996
ν	7.45828	5.07757	11.1941

Table 12: Posterior Mean of GARCH(1,1) with Student-t Innovations including the left and right quantile for the Euro 50

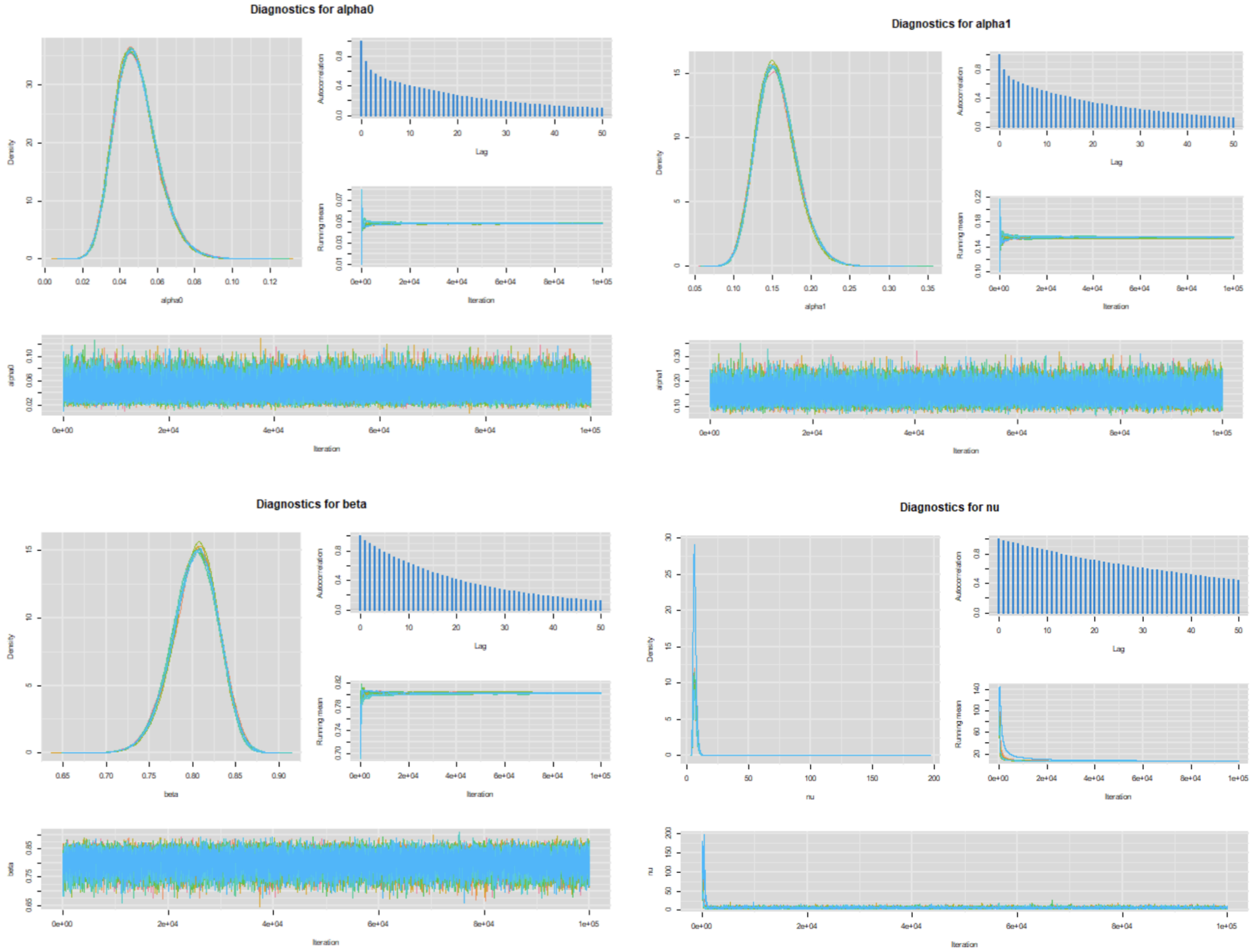


Figure 7: Diagnostics of the 1-million Markov Chain Monte Carlo simulations for the parameters of the S&P500.

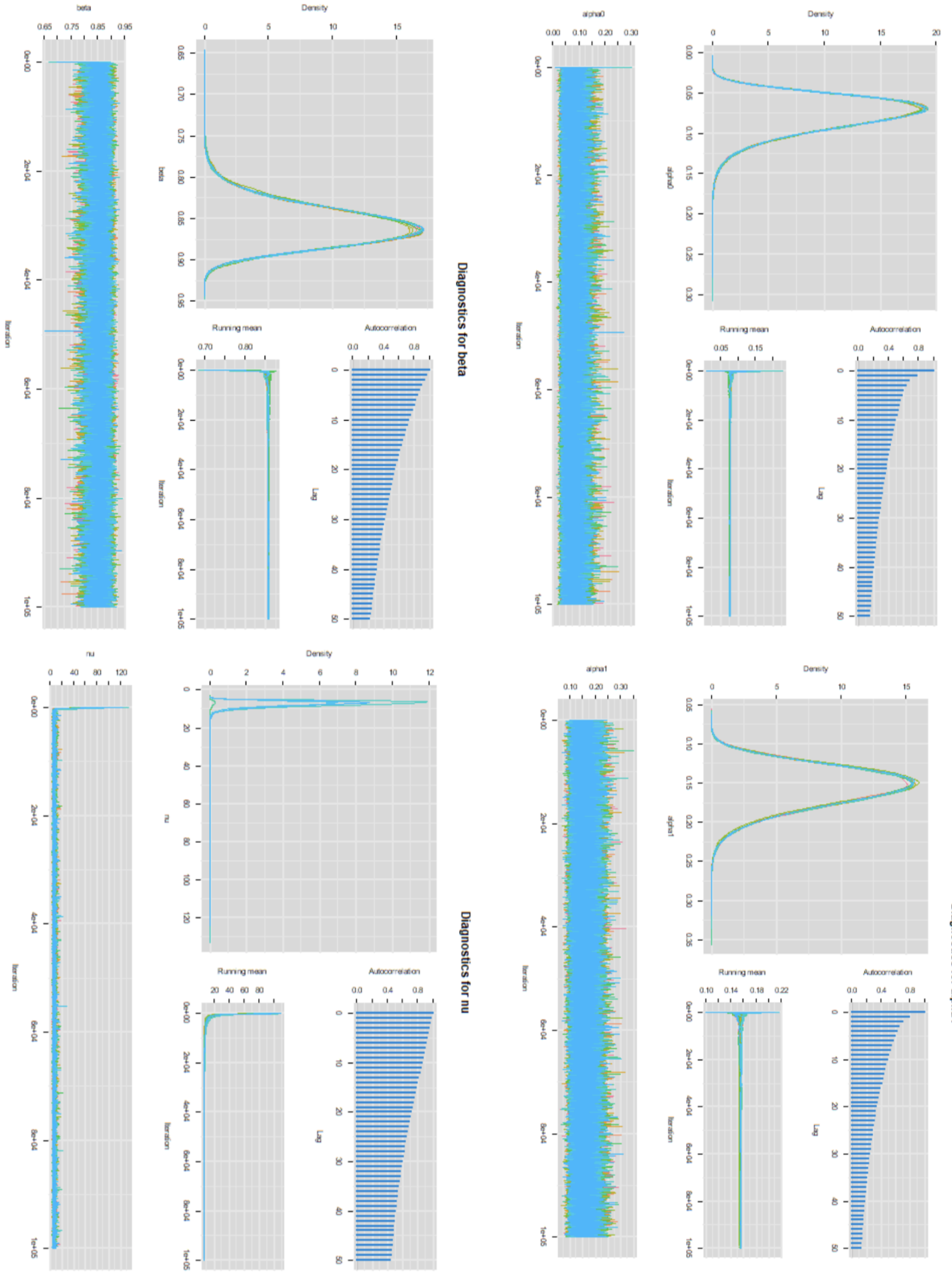


Figure 8: Diagnostics of the 1-million Markov Chain Monte Carlo simulations for the parameters of the Euro Stoxx 50.

S&P 500	dmstat	Euro Stoxx 50	dmstat
Bayesian vs GARCH(1,1)-N	56.0085	Bayesian vs GARCH(1,1)-N	59.2913
	[0.0000]		[0.0000]
Bayesian vs GARCH(1,1)-T	41.9187	Bayesian vs GARCH(1,1)-T	41.1199
	[0.0000]		[0.0000]
Bayesian vs GARCH(1,1)-ST	39.8659	Bayesian vs GARCH(1,1)-ST	36.0401
	[0.0000]		[0.0000]
Bayesian vs EGARCH-N	43.2703	Bayesian vs EGARCH-N	42.9609
	[0.0000]		[0.0000]
Bayesian vs EGARCH-T	31.8525	Bayesian vs EGARCH-T	32.3522
	[0.0000]		[0.0000]
Bayesian vs EGARCH-ST	30.4468	Bayesian vs EGARCH-ST	27.8622
	[0.0000]		[0.0000]
Bayesian vs GJR-N	44.8208	Bayesian vs GJR-N	47.6261
	[0.0000]		[0.0000]
Bayesian vs GJR-T	32.4451	Bayesian vs GJR-T	33.4261
	[0.0000]		[0.0000]
Bayesian vs GJR-ST	30.8850	Bayesian vs GJR-ST	29.137
	[0.0000]		[0.0000]

Table 13: Bayesian estimated GARCH(1,1)-T (denoted as Bayesian) DM-test statistics based on the log scoring rule compared to other models.

11.12 Appendix L - Backtesting Bayesian models:

Model	Expected Violations	Observed Violations	Violations Ratio	Estimated Probability ($\hat{p}$)
S&P 500				
GARCH-T VaR(5%)	51	23	0.45	2.27%
GARCH-T VaR(1%)	10	0	0.00	0.00%
Euro Stoxx 50				
GARCH-T VaR(5%)	50	23	0.46	2.28%
GARCH-T VaR(1%)	10	1	0.10	0.01%

Table 14: Value at Risk violations table based on Bayesian estimation. Where the expected number of violations is defined as the number of violations expected based on the selected p, the observed violations is the total sum of η_t, violation ratio is the ratio of observed violations and expected violations, and the estimated probability is the probability corresponding to the observed number of violations with the length of the vector η_t.

Model	Method	S&P 500-T-BAYESIAN	Euro Stoxx 50-T-BAYESIAN
GARCH-T VaR(5%)	LR_{uc}	19.8859	19.4334
		[0.0000]	[0.0000]
GARCH-T VaR(1%)	LR_{uc}	20.4022	13.6031
		[0.0000]	[0.0002]
GARCH-T VaR(5%)	LR_{ind}	0.3655	1.0764
		[0.5455]	[0.2995]
GARCH-T VaR(1%)	LR_{ind}	NaN	0.0020
			[0.9644]
GARCH-T VaR(5%)	LR_{joint}	20.2513	20.5098
		[0.0000]	[0.0000]
GARCH-T VaR(1%)	LR_{joint}	NaN	13.6051
			[0.0011]

Table 15: Backtesting of the model obtained by Bayesian estimation. UC is unconditional coverage test and IND is independence test.

11.13 Appendix M - In-sample & Out-of-sample summary statistics:

Index	S&P 500(y_t) In sample	S&P 500(y_t) Out of sample	Euro Stoxx 50(y_t) In sample	Euro Stoxx 50(y_t) Out of sample
Observations	1500	1016	1500	1008
Mean	0.0004	0.0005	0.0000	0.0002
Median	0.0006	0.0006	0.0003	0.0005
St.dev	0.0100	0.0081	0.0141	0.0098
Min	-0.0690	-0.0418	-0.0632	-0.0901
Max	0.0463	0.0484	0.0985	0.0391
Skewness	-0.4357	-0.6280	0.0264	-0.9313
Kurtosis	7.2369	7.5428	6.1750	11.3616
JB Statistic	1169.4	940.4	630.2	3082.2

Table 16: Summary statistics of the S&P 500 and Euro Stoxx 50 index grouped by in sample and out of sample period. y_t denotes the log returns of the respective index. Note that y_t is not expressed as a percentage. JB statistic stands for the Jarque-Bera test statistic, obtained from the Jarque-Bera test, which tests for normality of the data; critical value for $\alpha = 0.05$ is approximately 5.970.

11.14 Appendix N - $\hat{\gamma}$ comparison

$\hat{\gamma}$	S&P 500		Euro Stoxx 50	
	In-sample	Out-of-sample	In-sample	Out-of-sample
EGARCH-N	-0.2479	-0.2136	-0.2062	-0.1306
EGARCH-T	-0.2956	-0.2121	-0.2157	-0.1906
EGARCH-ST	-0.2980	-0.2138	-0.2168	-0.1868
GJR-N	0.2806	0.2934	0.2040	0.1941
GJR-T	0.3331	0.3350	0.2237	0.2687
GJR-ST	0.3473	0.3373	0.2231	0.2497

Table 17: Comparison of in-sample and out-of-sample gamma estimations, to analyse leverage effect estimates incongruencies.

In appendix M table 17 $\hat{\gamma}$'s are shown, and one can see that the $\hat{\gamma}$'s do indeed differ between the in-sample period and the out-of-sample period.

References

[1] L.F. Hoogerheide & D. Ardia. "GARCH models for daily stock returns: Impact of estimation frequency on Value-at-Risk and Expected Shortfall forecasts". In: *Elsevier* 123.2 (2014), pp. 187–190.

[2] T. Bollerslev. "Generalized Autoregressive Conditional Heteroskedasticity". In: *Journal of Econometrics* 123 (1986), pp. 307–327.

[3] D. Ardia L.F. Hoogerheide. *Bayesian Estimation of the GARCH(1,1) Model with Student-t Innovations*. Erasmus University, Rotterdam: Tinbergen Institute Discussion Paper, 2010.

[4] M. Lipovina-Božović S. Vujošević J.C. Smolović. "GARCH models in value at risk estimation: empirical evidence from the Montenegrin stock exchange". In: *Economic Research-Ekonomska Istraživanja* 30 (2017), pp. 477–498.

[5] E.Jondeau & S.H. Poon & M. Rockinger. *Financial Modeling Under Non-Gaussian Distributions*. New York, New York: Springer, 2007.

[6] R.S. Tsay. *Analysis of Financial Time Series*. Hoboken, New Jersey: John Wiley & Sons, 2010.

[7] M. So & P. Yu. "Empirical analysis of GARCH models in value at risk estimation". In: *Journal of international financial markets, institutions and money* 16 (2005), pp. 180–197.

PCS-30

Read me: the python file to estimate the variance forecasts, also estimates the inverse CDFs of the corresponding distribution, which is in turn used in the Value at Risk forecast. These quantilles are hard-coded into MATLAB, because they can not be cross-referenced. Also, in the various MATLB codes, we often state that we calculate a normal VaR forecast, but not all these VaR forecasts are normal. This is due to copy pasting of the code. To know what distribution is used in the VaR forecast code see the title of the code.

```
# Python file to estimate the variance forecasts


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from arch.univariate import arch_model

# Data Sets:
sp          = '^GSPC.csv'
eu          = '^STOXX50E.csv'

### Constant Parameters and the data set you want: ###
FILE = sp
SCALING = 100

def main(y, model, distr, SCALING):


    split_date = 1500


    if model == 'garch':
        am = arch_model(y,mean='Zero', dist = distr)
        res = am.fit(last_obs=split_date)
        forecasts = res.forecast(horizon=1, start = split_date)

        std_error = res.std_err
        std_error[0] = std_error[0]

        params = res.params
        params[0] = params[0]

        likelihood = res.loglikelihood
        aic_crit = 2*len(params) - 2*likelihood
        bic_crit = np.log(len(forecasts.variance)) * len(params) - 2*likelihood


    elif model == 'egarch':
        am = arch_model(y,mean='Zero', vol='EGARCH',p=1, o=1, q=1, dist = distr)
        res = am.fit(last_obs=split_date)
        forecasts = res.forecast(horizon=1, start = split_date)

        std_error = res.std_err
        std_error[0] = std_error[0]

        params = res.params
        params[0] = params[0]

        likelihood = res.loglikelihood
        aic_crit = 2*len(params) - 2*likelihood
        bic_crit = np.log(len(forecasts.variance)) * len(params) - 2*likelihood
```

```
    elif model == 'gjr-garch':
        am = arch_model(y,mean='Zero', vol='GARCH', p=1, o=1, q=1, dist = distr)
        res = am.fit(last_obs=split_date)
        forecasts = res.forecast(horizon=1, start = split_date)

        std_error = res.std_err
        std_error[0] = std_error[0]

        params = res.params
        params[0] = params[0]


        likelihood = res.loglikelihood
        aic_crit = 2*len(params) - 2*likelihood
        bic_crit = np.log(len(forecasts.variance)) * len(params) - 2*likelihood
        print(res.summary())



# To calculate the quantilles of the corresponding distribution, which are hard coded in MATLAB...

    if distr == 'gaussian':
        q = am.distribution.ppf([0.01, 0.05], None)

    elif distr == 't':
        q = am.distribution.ppf([0.01, 0.05], res.params[-1:])

    elif distr == 'skewt':
        q = am.distribution.ppf([0.01, 0.05], res.params[-2:])



    vf = forecasts.variance



    return vf, params, aic_crit, bic_crit, likelihood,q




# Possible models: 'GARCH' (default), 'ARCH', 'EGARCH', 'FIARCH' and 'HARCH'

# Possible distributions: Normal: 'normal', 'gaussian' (default)
# Students's t: 't', 'studentst'
# Skewed Student's t: 'skewstudent', 'skewt'
# Generalized Error Distribution: 'ged', 'generalized error"

###### Main Parameters ######



# Create data frame with dates
df              = pd.read_csv(FILE)
df['y']         = np.log(df['Adj Close']).diff()

y = SCALING*df['y'].dropna().to_numpy()
```

```
# if scaling is off:
#vf = np.square((np.sqrt(forecasts.variance)/1000))


# Volatility Models
garch               = 'garch'
e                   = 'egarch'
gjr                 = 'gjr-garch'

# Distributions:
normal              = 'gaussian'
student_t           = 't'
skew_student_t      = 'skewt'
ged                 = 'ged'




### Main Function: create a df of all the vfs ###

# GARCH model:
vf, garch_norm_params,\
 garch_norm_aic_crit, \
 garch_norm_bic_crit, \
 garch_norm_likelihood,_ = main(y, garch, normal, SCALING)
vf.columns = ['garch_norm']

vf['garch_t'], \
garch_t_params, \
garch_t_aic_crit, \
garch_t_bic_crit, \
garch_t_likelihood,_ = main(y, garch, student_t, SCALING)

vf['garch_skewt'], \
garch_skewt_params, \
garch_skewt_aic_crit, \
garch_skewt_bic_crit, \
garch_skewt_likelihood,_ = main(y, garch, skew_student_t, SCALING)



# EGARCH model:
vf['egarch_norm'], \
egarch_norm_params, \
egarch_norm_aic_crit, \
egarch_norm_bic_crit, \
egarch_norm_likelihood,_ = main(y, e, normal, SCALING)

vf['egarch_t'], \
egarch_t_params, \
egarch_t_aic_crit, \
egarch_t_bic_crit, \
egarch_t_likelihood,_ = main(y, e, student_t, SCALING)

vf['egarch_skewt'], \
egarch_skewt_params, \
egarch_skewt_aic_crit, \
egarch_skewt_bic_crit, \
```

```
egarch_skewt_likelihood,_ = main(y, e, skew_student_t, SCALING)



# GJR-GARCH model:
vf['gjr-garch_norm'], \
gjr_norm_params, \
gjr_norm_aic_crit, \
gjr_norm_bic_crit, \
gjr_norm_likelihood,_ = main(y, gjr, normal, SCALING)

vf['gjr-garch_t'], \
gjr_t_params, \
gjr_t_aic_crit, \
gjr_t_bic_crit, \
gjr_t_likelihood,_ = main(y, gjr, student_t, SCALING)

vf['gjr-garch_skewt'], \
gjr_skewt_params, \
gjr_skewt_aic_crit, \
gjr_skewt_bic_crit, \
gjr_skewt_likelihood,_ = main(y, gjr, skew_student_t, SCALING)




# Date
vf['Date'] = df['Date'].shift(-1)
vf['Date'] = pd.to_datetime(vf['Date'])
vf.set_index('Date', inplace=True)


# Write to csv

# SP
#vf.to_csv('vf_sp.csv', index=True)
#
#garch_norm_params.to_csv('garch_norm_params_sp.csv',index=True)
#garch_t_params.to_csv('garch_t_params_sp.csv',index=True)
#garch_skewt_params.to_csv('garch_skewt_params_sp.csv',index=True)
#
#egarch_norm_params.to_csv('egarch_norm_params_sp.csv',index=True)
#egarch_t_params.to_csv('egarch_t_params_sp.csv',index=True)
#egarch_skewt_params.to_csv('egarch_skewt_params_sp.csv',index=True)
#
#gjr_norm_params.to_csv('gjr_norm_params_sp.csv',index=True)
#gjr_t_params.to_csv('gjr_t_params_sp.csv',index=True)
#gjr_skewt_params.to_csv('gjr_skewt_params_sp.csv',index=True)



# EU
#vf.to_csv('vf_eu.csv', index=True)
##
#garch_norm_params.to_csv('garch_norm_params_eu.csv',index=True)
#garch_t_params.to_csv('garch_t_params_eu.csv',index=True)
#garch_skewt_params.to_csv('garch_skewt_params_eu.csv',index=True)
#
#egarch_norm_params.to_csv('egarch_norm_params_eu.csv',index=True)
```

```
#egarch_t_params.to_csv('egarch_t_params_eu.csv',index=True)
#egarch_skewt_params.to_csv('egarch_skewt_params_eu.csv',index=True)
#
#gjr_norm_params.to_csv('gjr_norm_params_eu.csv',index=True)
#gjr_t_params.to_csv('gjr_t_params_eu.csv',index=True)
#gjr_skewt_params.to_csv('gjr_skewt_params_eu.csv',index=True)




### Calculate quantile per (skewed) student-t ###

# GARCH:

_,_,_,_,_, garch_q_t = main(y, garch, student_t, SCALING)
_,_,_,_,_, garch_q_skewt = main(y, garch, skew_student_t, SCALING)

# EGARCH:
_,_,_,_,_, egarch_q_t = main(y, e, student_t, SCALING)
_,_,_,_,_, egarch_q_skewt = main(y, e, skew_student_t, SCALING)

# GJR-GARCH:
_,_,_,_,_, gjr_q_t = main(y, gjr, student_t, SCALING)
_,_,_,_,_, gjr_q_skewt = main(y, gjr, skew_student_t, SCALING)
```

VaR for Normals

```
    addpath 'D:\Files\VU Bestanden\Jaar 3\Practical Case Study (GARCH)\MATLAB PCS'
clear;
clc;
format longG

file_sp                     = 'sp500.csv';
file_eu                     = 'stoxx50.csv';

% Imports data and converts to datetime & log returns
[dates_sp,df_sp,close_sp,y_sp]       = process_data(file_sp);
[dates_eu,df_eu,close_eu,y_eu]       = process_data(file_eu);

y_sp = y_sp.*100;
y_eu = y_eu.*100;

dates_sp_ret = dates_sp(2:end);
close_sp_ret = close_sp(2:end);

dates_eu_ret = dates_eu(2:end);
close_eu_ret = close_eu(2:end);

WT_sp = 1015;
WT_eu = 1007;

VaR_matrix_sp = NaN(1015,6);
VaR_matrix_eu = NaN(1007,6);


%% SP
% Import
varForecastsPython_sp = csvread('vf_sp.csv',1,1);
```

```
% GARCH(1,1)
vfGARCH_sp = varForecastsPython_sp(1501:end,1);

% EGARCH(1,1)
vfE_sp = varForecastsPython_sp(1501:end,4);

% GJRGARCH(1,1)
vfGJR_sp = varForecastsPython_sp(1501:end,7);

%% EU

varForecastsPython_eu = csvread('vf_eu.csv',1,1);
% GARCH(1,1)

vfGARCH_eu = varForecastsPython_eu(1501:end,1);

% EGARCH(1,1)
vfE_eu = varForecastsPython_eu(1501:end,4);

% GJRGARCH(1,1)
vfGJR_eu = varForecastsPython_eu(1501:end,7);



%% VAR crap
%% VaR(5)/GARCH_NORM - S&P 500
VaR_norm_5_sp = VaR_norm(close_sp(2:end),vfGARCH_sp,0.05);
VaR_matrix_sp(1:end,1) = VaR_norm_5_sp;

%% VaR(5)/GARCH_NORM - Euro Stoxx 50
VaR_norm_5_eu = VaR_norm(close_eu(2:end),vfGARCH_eu,0.05);
VaR_matrix_eu(1:end,1) = VaR_norm_5_eu;

%% VaR(1)/GARCH_NORM - S&P 500
VaR_norm_1_sp = VaR_norm(close_sp(2:end),vfGARCH_sp,0.01);
VaR_matrix_sp(1:end,2) = VaR_norm_1_sp;

%% VaR(1)/GARCH_NORM - Euro Stoxx 50
VaR_norm_1_eu = VaR_norm(close_eu(2:end),vfGARCH_eu,0.01);
VaR_matrix_eu(1:end,2) = VaR_norm_1_eu;


%% VaR(5)/EGARCH_NORM - S&P 500
VaR_E_norm_5_sp = VaR_norm(close_sp(2:end),vfE_sp,0.05);
VaR_matrix_sp(1:end,3) = VaR_E_norm_5_sp;

%% VaR(5)/EGARCH_NORM - Euro Stoxx 50
VaR_E_norm_5_eu = VaR_norm(close_eu(2:end),vfE_eu,0.05);
VaR_matrix_eu(1:end,3) = VaR_E_norm_5_eu;

%% VaR(1)/EGARCH_NORM - S&P 500
VaR_E_norm_1_sp = VaR_norm(close_sp(2:end),vfE_sp,0.01);
VaR_matrix_sp(1:end,4) = VaR_E_norm_1_sp;

%% VaR(1)/EGARCH_NORM - Euro Stoxx 50
VaR_E_norm_1_eu = VaR_norm(close_eu(2:end),vfE_eu,0.01);
VaR_matrix_eu(1:end,4) = VaR_E_norm_1_eu;


%% VaR(5)/GJR-GARCH_NORM - S&P 500
```

```
VaR_GJR_norm_5_sp = VaR_norm(close_sp(2:end),vfGJR_sp,0.05);
VaR_matrix_sp(1:end,5) = VaR_GJR_norm_5_sp;

%% VaR(5)/GJR-GARCH_NORM - Euro Stoxx 50
VaR_GJR_norm_5_eu = VaR_norm(close_eu(2:end),vfGJR_eu,0.05);
VaR_matrix_eu(1:end,5) = VaR_GJR_norm_5_eu;

%% VaR(1)/GJR-GARCH_NORM - S&P 500
VaR_GJR_norm_1_sp = VaR_norm(close_sp(2:end),vfGJR_sp,0.01);
VaR_matrix_sp(1:end,6) = VaR_GJR_norm_1_sp;

%% VaR(1)/GJR-GARCH_NORM - Euro Stoxx 50
VaR_GJR_norm_1_eu = VaR_norm(close_eu(2:end),vfGJR_eu,0.01);
VaR_matrix_eu(1:end,6) = VaR_GJR_norm_1_eu;


%% Tabulate
VaR_table_sp = array2table(VaR_matrix_sp, ...
    'VariableNames',{'VaR(5) GARCH-NORM','VaR(1) GARCH-NORM',...
    'VaR(5) EGARCH-NORM','VaR(1) EGARCH-NORM',...
    'VaR(5) GJR-GARCH-NORM', 'VaR(1) GJR-GARCH-NORM'});

VaR_table_eu = array2table(VaR_matrix_eu, ...
    'VariableNames',{'VaR(5) GARCH-NORM','VaR(1) GARCH-NORM',...
    'VaR(5) EGARCH-NORM','VaR(1) EGARCH-NORM',...
    'VaR(5) GJR-GARCH-NORM', 'VaR(1) GJR-GARCH-NORM'});

%writetable(VaR_table_sp, 'VaR_table_sp_normal.csv');
%writetable(VaR_table_eu, 'VaR_table_eu_normal.csv');

%% Calculate Violations - S&P 500
v_GARCH_5_sp = violation(VaR_norm_5_sp, y_sp(1502:end));
v_GARCH_1_sp = violation(VaR_norm_1_sp, y_sp(1502:end));

v_E_5_sp = violation(VaR_E_norm_5_sp, y_sp(1502:end));
v_E_1_sp = violation(VaR_E_norm_1_sp, y_sp(1502:end));

v_GJR_5_sp = violation(VaR_GJR_norm_5_sp, y_sp(1502:end));
v_GJR_1_sp = violation(VaR_GJR_norm_1_sp, y_sp(1502:end));


%% Calculate Violations - Euro Stoxx 50
v_GARCH_5_eu = violation(VaR_norm_5_eu, y_eu(1502:end));
v_GARCH_1_eu = violation(VaR_norm_1_eu, y_eu(1502:end));

v_E_5_eu = violation(VaR_E_norm_5_eu, y_eu(1502:end));
v_E_1_eu = violation(VaR_E_norm_1_eu, y_eu(1502:end));

v_GJR_5_eu = violation(VaR_GJR_norm_5_eu, y_eu(1502:end));
v_GJR_1_eu = violation(VaR_GJR_norm_1_eu, y_eu(1502:end));



%% Bernoulli Coverage Tests - S&P 500
bern_GARCH_5_sp = bern_test(0.05,v_GARCH_5_sp);
bern_GARCH_1_sp = bern_test(0.01,v_GARCH_1_sp);

bern_E_5_sp = bern_test(0.05,v_E_5_sp);
bern_E_1_sp = bern_test(0.01,v_E_1_sp);
```

```
bern_GJR_5_sp = bern_test(0.05,v_GJR_5_sp);
bern_GJR_1_sp = bern_test(0.01,v_GJR_1_sp);

%% Bernoulli Coverage Tests - Euro Stoxx 50
bern_GARCH_5_eu = bern_test(0.05,v_GARCH_5_eu);
bern_GARCH_1_eu = bern_test(0.01,v_GARCH_1_eu);

bern_E_5_eu = bern_test(0.05,v_E_5_eu);
bern_E_1_eu = bern_test(0.01,v_E_1_eu);

bern_GJR_5_eu = bern_test(0.05,v_GJR_5_eu);
bern_GJR_1_eu = bern_test(0.01,v_GJR_1_eu);

%% Independence Tests - S&P 500
ind_GARCH_5_sp = ind_test(v_GARCH_5_sp);
ind_GARCH_1_sp = ind_test(v_GARCH_1_sp);

ind_E_5_sp = ind_test(v_E_5_sp);
ind_E_1_sp = ind_test(v_E_1_sp);

ind_GJR_5_sp = ind_test(v_GJR_5_sp);
ind_GJR_1_sp = ind_test(v_GJR_1_sp);

%% Independence Tests - Euro Stoxx 50

ind_GARCH_5_eu = ind_test(v_GARCH_5_eu);
ind_GARCH_1_eu = ind_test(v_GARCH_1_eu);

ind_E_5_eu = ind_test(v_E_5_eu);
ind_E_1_eu = ind_test(v_E_1_eu);

ind_GJR_5_eu = ind_test(v_GJR_5_eu);
ind_GJR_1_eu = ind_test(v_GJR_1_eu);

%% Tabulate
% S&P 500
bern_matrix_sp(1:length(bern_GARCH_5_sp),1) = bern_GARCH_5_sp;
bern_matrix_sp(1:length(bern_GARCH_1_sp),2) = bern_GARCH_1_sp;

bern_matrix_sp(1:length(bern_E_5_sp),3) = bern_E_5_sp;
bern_matrix_sp(1:length(bern_E_1_sp),4) = bern_E_1_sp;

bern_matrix_sp(1:length(bern_GJR_5_sp),5) = bern_GJR_5_sp;
bern_matrix_sp(1:length(bern_GJR_1_sp),6) = bern_GJR_1_sp;

% Euro Stoxx 50
bern_matrix_eu(1:length(bern_GARCH_5_eu),1) = bern_GARCH_5_eu;
bern_matrix_eu(1:length(bern_GARCH_1_eu),2) = bern_GARCH_1_eu;

bern_matrix_eu(1:length(bern_E_5_eu),3) = bern_E_5_eu;
bern_matrix_eu(1:length(bern_E_1_eu),4) = bern_E_1_eu;

bern_matrix_eu(1:length(bern_GJR_5_eu),5) = bern_GJR_5_eu;
bern_matrix_eu(1:length(bern_GJR_1_eu),6) = bern_GJR_1_eu;

bern_table_sp = array2table(bern_matrix_sp, ...
    'VariableNames',{'bern_GARCH_5_sp','bern_GARCH_1_sp',...
    'bern_E_5_sp','bern_E_1_sp',...
    'bern_GJR_5_sp', 'bern_GJR_1_sp'});
```

```
bern_table_eu = array2table(bern_matrix_eu, ...
    'VariableNames',{'bern_GARCH_5_eu','bern_GARCH_1_eu',...
    'bern_E_5_eu','bern_E_1_eu',...
    'bern_GJR_5_eu', 'bern_GJR_1_eu'});

writetable(bern_table_sp, 'bern_table_sp.csv');
writetable(bern_table_eu, 'bern_table_eu.csv');


%% Tabulate
% S&P 500
ind_matrix_sp(1:length(ind_GARCH_5_sp),1) = ind_GARCH_5_sp;
ind_matrix_sp(1:length(ind_GARCH_1_sp),2) = ind_GARCH_1_sp;

ind_matrix_sp(1:length(ind_E_5_sp),3) = ind_E_5_sp;
ind_matrix_sp(1:length(ind_E_1_sp),4) = ind_E_1_sp;

ind_matrix_sp(1:length(ind_GJR_5_sp),5) = ind_GJR_5_sp;
ind_matrix_sp(1:length(ind_GJR_1_sp),6) = ind_GJR_1_sp;

% Euro Stoxx 50
ind_matrix_eu(1:length(ind_GARCH_5_eu),1) = ind_GARCH_5_eu;
ind_matrix_eu(1:length(ind_GARCH_1_eu),2) = ind_GARCH_1_eu;

ind_matrix_eu(1:length(ind_E_5_eu),3) = ind_E_5_eu;
ind_matrix_eu(1:length(ind_E_1_eu),4) = ind_E_1_eu;

ind_matrix_eu(1:length(ind_GJR_5_eu),5) = ind_GJR_5_eu;
ind_matrix_eu(1:length(ind_GJR_1_eu),6) = ind_GJR_1_eu;

ind_table_sp = array2table(ind_matrix_sp, ...
    'VariableNames',{'ind_GARCH_5_sp','ind_GARCH_1_sp',...
    'ind_E_5_sp','ind_E_1_sp',...
    'ind_GJR_5_sp', 'ind_GJR_sp'});

ind_table_eu = array2table(ind_matrix_eu, ...
    'VariableNames',{'ind_GARCH_5_eu','ind_GARCH_1_eu',...
    'ind_E_5_eu','ind_E_1_eu',...
    'ind_GJR_5_eu', 'ind_GJR_1_eu'});

writetable(ind_table_sp, 'ind_table_sp.csv');
writetable(ind_table_eu, 'ind_table_eu.csv');


%% Joint test
% S&P 500
joint_GARCH_5_sp = bern_GARCH_5_sp + ind_GARCH_5_sp;
joint_GARCH_1_sp = bern_GARCH_1_sp + ind_GARCH_1_sp;

joint_E_5_sp = bern_E_5_sp + ind_E_5_sp;
joint_E_1_sp = bern_E_1_sp + ind_E_1_sp;

joint_GJR_5_sp = bern_GJR_5_sp + ind_GJR_5_sp;
joint_GJR_1_sp = bern_GJR_1_sp + ind_GJR_1_sp;

% Euro Stoxx 50
joint_GARCH_5_eu = bern_GARCH_5_eu + ind_GARCH_5_eu;
joint_GARCH_1_eu = bern_GARCH_1_eu + ind_GARCH_1_eu;

joint_E_5_eu = bern_E_5_eu + ind_E_5_eu;
```

```
joint_E_1_eu = bern_E_1_eu + ind_E_1_eu;

joint_GJR_5_eu = bern_GJR_5_eu + ind_GJR_5_eu;
joint_GJR_1_eu = bern_GJR_1_eu + ind_GJR_1_eu;

%% Tabulate
% S&P 500
joint_matrix_sp(1:length(ind_GARCH_5_sp),1) = joint_GARCH_5_sp;
joint_matrix_sp(1:length(ind_GARCH_5_sp),2) = joint_GARCH_1_sp;

joint_matrix_sp(1:length(ind_GARCH_5_sp),3) = joint_E_5_sp;
joint_matrix_sp(1:length(ind_GARCH_5_sp),4) = joint_E_1_sp;

joint_matrix_sp(1:length(ind_GARCH_5_sp),5) = joint_GJR_5_sp;
joint_matrix_sp(1:length(ind_GARCH_5_sp),6) = joint_GJR_1_sp;

% Euro Stoxx 50
joint_matrix_eu(1:length(ind_GARCH_5_sp),1) = joint_GARCH_5_eu;
joint_matrix_eu(1:length(ind_GARCH_5_sp),2) = joint_GARCH_1_eu;

joint_matrix_eu(1:length(ind_GARCH_5_sp),3) = joint_E_5_eu;
joint_matrix_eu(1:length(ind_GARCH_5_sp),4) = joint_E_1_eu;

joint_matrix_eu(1:length(ind_GARCH_5_sp),5) = joint_GJR_5_eu;
joint_matrix_eu(1:length(ind_GARCH_5_sp),6) = joint_GJR_1_eu;

joint_table_sp = array2table(joint_matrix_sp, ...
    'VariableNames',{'joint_GARCH_5_sp','joint_GARCH_1_sp',...
    'joint_E_5_sp','joint_E_1_sp',...
    'joint_GJR_5_sp', 'joint_GJR_1_sp'});

joint_table_eu = array2table(joint_matrix_eu, ...
    'VariableNames',{'joint_GARCH_5_eu','joint_GARCH_1_eu',...
    'joint_E_5_eu','joint_E_1_eu',...
    'joint_GJR_5_eu', 'joint_GJR_1_eu'});

writetable(joint_table_sp, 'joint_table_sp.csv');
writetable(joint_table_eu, 'joint_table_eu.csv');



plot(y_sp(1502:end))
hold on
plot(VaR_norm_5_sp)
hold off

%% Violations
exp_vio_5_sp = round(0.05*length(v_GARCH_5_sp))
exp_vio_1_sp = round(0.01*length(v_GARCH_1_sp))

exp_vio_5_eu = round(0.05*length(v_GARCH_5_eu))
exp_vio_1_eu = round(0.01*length(v_GARCH_1_eu))


vio_5_G_sp = sum(v_GARCH_5_sp);
vio_1_G_sp = sum(v_GARCH_1_sp);

vio_5_E_sp = sum(v_E_5_sp);
vio_1_E_sp = sum(v_E_1_sp);
```

```
vio_5_GJR_sp = sum(v_GJR_5_sp);
vio_1_GJR_sp = sum(v_GJR_1_sp);


vio_5_G_eu = sum(v_GARCH_5_eu)
vio_1_G_eu = sum(v_GARCH_1_eu)

vio_5_E_eu = sum(v_E_5_eu)
vio_1_E_eu = sum(v_E_1_eu)

vio_5_GJR_eu = sum(v_GJR_5_eu)
vio_1_GJR_eu = sum(v_GJR_1_eu)
```

VaR for student t

```
% In the comments there is stated that normal garch models are estimated,
% this is however not true, see title for the kind of model used.

clear;
clc;
format longG

file_sp                     = 'sp500.csv';
file_eu                     = 'stoxx50.csv';

% Imports data and converts to datetime & log returns
[dates_sp,df_sp,close_sp,y_sp]       = process_data(file_sp);
[dates_eu,df_eu,close_eu,y_eu]       = process_data(file_eu);

y_sp = y_sp.*100;
y_eu = y_eu.*100;

dates_sp_ret = dates_sp(2:end);
close_sp_ret = close_sp(2:end);

dates_eu_ret = dates_eu(2:end);
close_eu_ret = close_eu(2:end);

WT_sp = 1015;
WT_eu = 1007;

VaR_matrix_sp = NaN(1015,6);
VaR_matrix_eu = NaN(1007,6);


%% SP
% Import
varForecastsPython_sp = csvread('vf_sp.csv',1,1);
varForecastsBAYES_sp = csvread('vf_sp_bays.csv',1,1);
% GARCH(1,1)
vfGARCH_sp = varForecastsPython_sp(1501:end,2);

% EGARCH(1,1)
vfE_sp = varForecastsPython_sp(1501:end,5);

% GJRGARCH(1,1)
vfGJR_sp = varForecastsPython_sp(1501:end,8);

%% EU
```

```
varForecastsPython_eu = csvread('vf_eu.csv',1,1);
varForecastsBAYES_eu = csvread('vf_eu_bays.csv',1,1);
% GARCH(1,1)

vfGARCH_eu = varForecastsPython_eu(1501:end,2);

% EGARCH(1,1)
vfE_eu = varForecastsPython_eu(1501:end,5);

% GJRGARCH(1,1)
vfGJR_eu = varForecastsPython_eu(1501:end,8);



%% VAR
%% VaR(5)/GARCH_NORM - S&P 500
VaR_norm_5_sp = VaR_skewt(close_sp(2:end),vfGARCH_sp,-1.59073);
VaR_matrix_sp(1:end,1) = VaR_norm_5_sp;

%% VaR(5)/GARCH_NORM - Euro Stoxx 50
VaR_norm_5_eu = VaR_skewt(close_eu(2:end),vfGARCH_eu,-1.60017);
VaR_matrix_eu(1:end,1) = VaR_norm_5_eu;

%% VaR(1)/GARCH_NORM - S&P 500
VaR_norm_1_sp = VaR_skewt(close_sp(2:end),vfGARCH_sp,-2.5577);
VaR_matrix_sp(1:end,2) = VaR_norm_1_sp;

%% VaR(1)/GARCH_NORM - Euro Stoxx 50
VaR_norm_1_eu = VaR_skewt(close_eu(2:end),vfGARCH_eu,-2.53633);
VaR_matrix_eu(1:end,2) = VaR_norm_1_eu;


%% VaR(5)/EGARCH_NORM - S&P 500
VaR_E_norm_5_sp = VaR_skewt(close_sp(2:end),vfE_sp,-1.60137);
VaR_matrix_sp(1:end,3) = VaR_E_norm_5_sp;

%% VaR(5)/EGARCH_NORM - Euro Stoxx 50
VaR_E_norm_5_eu = VaR_skewt(close_eu(2:end),vfE_eu,-1.61634);
VaR_matrix_eu(1:end,3) = VaR_E_norm_5_eu;

%% VaR(1)/EGARCH_NORM - S&P 500
VaR_E_norm_1_sp = VaR_skewt(close_sp(2:end),vfE_sp,-2.53332);
VaR_matrix_sp(1:end,4) = VaR_E_norm_1_sp;

%% VaR(1)/EGARCH_NORM - Euro Stoxx 50
VaR_E_norm_1_eu = VaR_skewt(close_eu(2:end),vfE_eu,-2.48934);
VaR_matrix_eu(1:end,4) = VaR_E_norm_1_eu;


%% VaR(5)/GJR-GARCH_NORM - S&P 500
VaR_GJR_norm_5_sp = VaR_skewt(close_sp(2:end),vfGJR_sp,-1.60365);
VaR_matrix_sp(1:end,5) = VaR_GJR_norm_5_sp;

%% VaR(5)/GJR-GARCH_NORM - Euro Stoxx 50
VaR_GJR_norm_5_eu = VaR_skewt(close_eu(2:end),vfGJR_eu,-1.61175);
VaR_matrix_eu(1:end,5) = VaR_GJR_norm_5_eu;

%% VaR(1)/GJR-GARCH_NORM - S&P 500
VaR_GJR_norm_1_sp = VaR_skewt(close_sp(2:end),vfGJR_sp,-2.52747);
VaR_matrix_sp(1:end,6) = VaR_GJR_norm_1_sp;
```

```
%% VaR(1)/GJR-GARCH_NORM - Euro Stoxx 50
VaR_GJR_norm_1_eu = VaR_skewt(close_eu(2:end),vfGJR_eu,-2.50433);
VaR_matrix_eu(1:end,6) = VaR_GJR_norm_1_eu;


%% Tabulate
VaR_table_sp = array2table(VaR_matrix_sp, ...
    'VariableNames',{'VaR(5) GARCH-NORM','VaR(1) GARCH-NORM',...
    'VaR(5) EGARCH-NORM','VaR(1) EGARCH-NORM',...
    'VaR(5) GJR-GARCH-NORM', 'VaR(1) GJR-GARCH-NORM'});

VaR_table_eu = array2table(VaR_matrix_eu, ...
    'VariableNames',{'VaR(5) GARCH-NORM','VaR(1) GARCH-NORM',...
    'VaR(5) EGARCH-NORM','VaR(1) EGARCH-NORM',...
    'VaR(5) GJR-GARCH-NORM', 'VaR(1) GJR-GARCH-NORM'});

%writetable(VaR_table_sp, 'VaR_table_sp_normal.csv');
%writetable(VaR_table_eu, 'VaR_table_eu_normal.csv');

%% Calculate Violations - S&P 500
v_GARCH_5_sp = violation(VaR_norm_5_sp, y_sp(1502:end));
v_GARCH_1_sp = violation(VaR_norm_1_sp, y_sp(1502:end));

v_E_5_sp = violation(VaR_E_norm_5_sp, y_sp(1502:end));
v_E_1_sp = violation(VaR_E_norm_1_sp, y_sp(1502:end));

v_GJR_5_sp = violation(VaR_GJR_norm_5_sp, y_sp(1502:end));
v_GJR_1_sp = violation(VaR_GJR_norm_1_sp, y_sp(1502:end));


%% Calculate Violations - Euro Stoxx 50
v_GARCH_5_eu = violation(VaR_norm_5_eu, y_eu(1502:end));
v_GARCH_1_eu = violation(VaR_norm_1_eu, y_eu(1502:end));

v_E_5_eu = violation(VaR_E_norm_5_eu, y_eu(1502:end));
v_E_1_eu = violation(VaR_E_norm_1_eu, y_eu(1502:end));

v_GJR_5_eu = violation(VaR_GJR_norm_5_eu, y_eu(1502:end));
v_GJR_1_eu = violation(VaR_GJR_norm_1_eu, y_eu(1502:end));



%% Bernoulli Coverage Tests - S&P 500
bern_GARCH_5_sp = bern_test(0.05,v_GARCH_5_sp);
bern_GARCH_1_sp = bern_test(0.01,v_GARCH_1_sp);

bern_E_5_sp = bern_test(0.05,v_E_5_sp);
bern_E_1_sp = bern_test(0.01,v_E_1_sp);

bern_GJR_5_sp = bern_test(0.05,v_GJR_5_sp);
bern_GJR_1_sp = bern_test(0.01,v_GJR_1_sp);

%% Bernoulli Coverage Tests - Euro Stoxx 50
bern_GARCH_5_eu = bern_test(0.05,v_GARCH_5_eu);
bern_GARCH_1_eu = bern_test(0.01,v_GARCH_1_eu);

bern_E_5_eu = bern_test(0.05,v_E_5_eu);
bern_E_1_eu = bern_test(0.01,v_E_1_eu);
```

```
bern_GJR_5_eu = bern_test(0.05,v_GJR_5_eu);
bern_GJR_1_eu = bern_test(0.01,v_GJR_1_eu);

%% Independence Tests - S&P 500
ind_GARCH_5_sp = ind_test(v_GARCH_5_sp);
ind_GARCH_1_sp = ind_test(v_GARCH_1_sp);

ind_E_5_sp = ind_test(v_E_5_sp);
ind_E_1_sp = ind_test(v_E_1_sp);

ind_GJR_5_sp = ind_test(v_GJR_5_sp);
ind_GJR_1_sp = ind_test(v_GJR_1_sp);

%% Independence Tests - Euro Stoxx 50

ind_GARCH_5_eu = ind_test(v_GARCH_5_eu);
ind_GARCH_1_eu = ind_test(v_GARCH_1_eu);

ind_E_5_eu = ind_test(v_E_5_eu);
ind_E_1_eu = ind_test(v_E_1_eu);

ind_GJR_5_eu = ind_test(v_GJR_5_eu);
ind_GJR_1_eu = ind_test(v_GJR_1_eu);

%% Tabulate
% S&P 500
bern_matrix_sp(1:length(bern_GARCH_5_sp),1) = bern_GARCH_5_sp;
bern_matrix_sp(1:length(bern_GARCH_1_sp),2) = bern_GARCH_1_sp;

bern_matrix_sp(1:length(bern_E_5_sp),3) = bern_E_5_sp;
bern_matrix_sp(1:length(bern_E_1_sp),4) = bern_E_1_sp;

bern_matrix_sp(1:length(bern_GJR_5_sp),5) = bern_GJR_5_sp;
bern_matrix_sp(1:length(bern_GJR_1_sp),6) = bern_GJR_1_sp;

% Euro Stoxx 50
bern_matrix_eu(1:length(bern_GARCH_5_eu),1) = bern_GARCH_5_eu;
bern_matrix_eu(1:length(bern_GARCH_1_eu),2) = bern_GARCH_1_eu;

bern_matrix_eu(1:length(bern_E_5_eu),3) = bern_E_5_eu;
bern_matrix_eu(1:length(bern_E_1_eu),4) = bern_E_1_eu;

bern_matrix_eu(1:length(bern_GJR_5_eu),5) = bern_GJR_5_eu;
bern_matrix_eu(1:length(bern_GJR_1_eu),6) = bern_GJR_1_eu;

bern_table_sp = array2table(bern_matrix_sp, ...
    'VariableNames',{'bern_GARCH_5_sp','bern_GARCH_1_sp',...
    'bern_E_5_sp','bern_E_1_sp',...
    'bern_GJR_5_sp', 'bern_GJR_1_sp'});

bern_table_eu = array2table(bern_matrix_eu, ...
    'VariableNames',{'bern_GARCH_5_eu','bern_GARCH_1_eu',...
    'bern_E_5_eu','bern_E_1_eu',...
    'bern_GJR_5_eu', 'bern_GJR_1_eu'});

writetable(bern_table_sp, 'bern_table_sp.csv');
writetable(bern_table_eu, 'bern_table_eu.csv');


%% Tabulate
```

```
% S&P 500
ind_matrix_sp(1:length(ind_GARCH_5_sp),1) = ind_GARCH_5_sp;
ind_matrix_sp(1:length(ind_GARCH_1_sp),2) = ind_GARCH_1_sp;

ind_matrix_sp(1:length(ind_E_5_sp),3) = ind_E_5_sp;
ind_matrix_sp(1:length(ind_E_1_sp),4) = ind_E_1_sp;

ind_matrix_sp(1:length(ind_GJR_5_sp),5) = ind_GJR_5_sp;
ind_matrix_sp(1:length(ind_GJR_1_sp),6) = ind_GJR_1_sp;

% Euro Stoxx 50
ind_matrix_eu(1:length(ind_GARCH_5_eu),1) = ind_GARCH_5_eu;
ind_matrix_eu(1:length(ind_GARCH_1_eu),2) = ind_GARCH_1_eu;

ind_matrix_eu(1:length(ind_E_5_eu),3) = ind_E_5_eu;
ind_matrix_eu(1:length(ind_E_1_eu),4) = ind_E_1_eu;

ind_matrix_eu(1:length(ind_GJR_5_eu),5) = ind_GJR_5_eu;
ind_matrix_eu(1:length(ind_GJR_1_eu),6) = ind_GJR_1_eu;

ind_table_sp = array2table(ind_matrix_sp, ...
    'VariableNames',{'ind_GARCH_5_sp','ind_GARCH_1_sp',...
    'ind_E_5_sp','ind_E_1_sp',...
    'ind_GJR_5_sp', 'ind_GJR_sp'});

ind_table_eu = array2table(ind_matrix_eu, ...
    'VariableNames',{'ind_GARCH_5_eu','ind_GARCH_1_eu',...
    'ind_E_5_eu','ind_E_1_eu',...
    'ind_GJR_5_eu', 'ind_GJR_1_eu'});

writetable(ind_table_sp, 'ind_table_sp.csv');
writetable(ind_table_eu, 'ind_table_eu.csv');


%% Joint test
% S&P 500
joint_GARCH_5_sp = bern_GARCH_5_sp + ind_GARCH_5_sp;
joint_GARCH_1_sp = bern_GARCH_1_sp + ind_GARCH_1_sp;

joint_E_5_sp = bern_E_5_sp + ind_E_5_sp;
joint_E_1_sp = bern_E_1_sp + ind_E_1_sp;

joint_GJR_5_sp = bern_GJR_5_sp + ind_GJR_5_sp;
joint_GJR_1_sp = bern_GJR_1_sp + ind_GJR_1_sp;

% Euro Stoxx 50
joint_GARCH_5_eu = bern_GARCH_5_eu + ind_GARCH_5_eu;
joint_GARCH_1_eu = bern_GARCH_1_eu + ind_GARCH_1_eu;

joint_E_5_eu = bern_E_5_eu + ind_E_5_eu;
joint_E_1_eu = bern_E_1_eu + ind_E_1_eu;

joint_GJR_5_eu = bern_GJR_5_eu + ind_GJR_5_eu;
joint_GJR_1_eu = bern_GJR_1_eu + ind_GJR_1_eu;

%% Tabulate
% S&P 500
joint_matrix_sp(1:length(ind_GARCH_5_sp),1) = joint_GARCH_5_sp;
joint_matrix_sp(1:length(ind_GARCH_5_sp),2) = joint_GARCH_1_sp;
```

```
joint_matrix_sp(1:length(ind_GARCH_5_sp),3) = joint_E_5_sp;
joint_matrix_sp(1:length(ind_GARCH_5_sp),4) = joint_E_1_sp;

joint_matrix_sp(1:length(ind_GARCH_5_sp),5) = joint_GJR_5_sp;
joint_matrix_sp(1:length(ind_GARCH_5_sp),6) = joint_GJR_1_sp;

% Euro Stoxx 50
joint_matrix_eu(1:length(ind_GARCH_5_sp),1) = joint_GARCH_5_eu;
joint_matrix_eu(1:length(ind_GARCH_5_sp),2) = joint_GARCH_1_eu;

joint_matrix_eu(1:length(ind_GARCH_5_sp),3) = joint_E_5_eu;
joint_matrix_eu(1:length(ind_GARCH_5_sp),4) = joint_E_1_eu;

joint_matrix_eu(1:length(ind_GARCH_5_sp),5) = joint_GJR_5_eu;
joint_matrix_eu(1:length(ind_GARCH_5_sp),6) = joint_GJR_1_eu;

joint_table_sp = array2table(joint_matrix_sp, ...
    'VariableNames',{'joint_GARCH_5_sp','joint_GARCH_1_sp',...
    'joint_E_5_sp','joint_E_1_sp',...
    'joint_GJR_5_sp', 'joint_GJR_1_sp'});

joint_table_eu = array2table(joint_matrix_eu, ...
    'VariableNames',{'joint_GARCH_5_eu','joint_GARCH_1_eu',...
    'joint_E_5_eu','joint_E_1_eu',...
    'joint_GJR_5_eu', 'joint_GJR_1_eu'});

writetable(joint_table_sp, 'joint_table_sp.csv');
writetable(joint_table_eu, 'joint_table_eu.csv');


%% Violations
exp_vio_5_sp = round(0.05*length(v_GARCH_5_sp))
exp_vio_1_sp = round(0.01*length(v_GARCH_1_sp))

exp_vio_5_eu = round(0.05*length(v_GARCH_5_eu))
exp_vio_1_eu = round(0.01*length(v_GARCH_1_eu))


vio_5_G_sp = sum(v_GARCH_5_sp);
vio_1_G_sp = sum(v_GARCH_1_sp);

vio_5_E_sp = sum(v_E_5_sp);
vio_1_E_sp = sum(v_E_1_sp);

vio_5_GJR_sp = sum(v_GJR_5_sp);
vio_1_GJR_sp = sum(v_GJR_1_sp);


vio_5_G_eu = sum(v_GARCH_5_eu)
vio_1_G_eu = sum(v_GARCH_1_eu)

vio_5_E_eu = sum(v_E_5_eu)
vio_1_E_eu = sum(v_E_1_eu)

vio_5_GJR_eu = sum(v_GJR_5_eu)
vio_1_GJR_eu = sum(v_GJR_1_eu)


plot(VaR_GJR_norm_5_sp)
hold on
```

```
plot(y_sp(1502:end))
```

VaR for skewed student-t

```
% In the comments there is stated that normal garch models are estimated,
% this is however not true, see title for the kind of model used.


clear;
clc;
format longG

file_sp                     = 'sp500.csv';
file_eu                     = 'stoxx50.csv';

% Imports data and converts to datetime & log returns
[dates_sp,df_sp,close_sp,y_sp]       = process_data(file_sp);
[dates_eu,df_eu,close_eu,y_eu]       = process_data(file_eu);

y_sp = y_sp.*100;
y_eu = y_eu.*100;

dates_sp_ret = dates_sp(2:end);
close_sp_ret = close_sp(2:end);

dates_eu_ret = dates_eu(2:end);
close_eu_ret = close_eu(2:end);

WT_sp = 1015;
WT_eu = 1007;

VaR_matrix_sp = NaN(1015,6);
VaR_matrix_eu = NaN(1007,6);


%% SP

varForecastsPython_sp = csvread('vf_sp.csv',1,1);
% GARCH(1,1)

vfGARCH_sp = varForecastsPython_sp(1501:end,3);

% EGARCH(1,1)
vfE_sp = varForecastsPython_sp(1501:end,6);

% GJRGARCH(1,1)
vfGJR_sp = varForecastsPython_sp(1501:end,9);

%% EU

varForecastsPython_eu = csvread('vf_eu.csv',1,1);
% GARCH(1,1)

vfGARCH_eu = varForecastsPython_eu(1501:end,3);

% EGARCH(1,1)
vfE_eu = varForecastsPython_eu(1501:end,6);

% GJRGARCH(1,1)
vfGJR_eu = varForecastsPython_eu(1501:end,9);
```

```
%% VAR crap
%% VaR(5)/GARCH_NORM - S&P 500
VaR_norm_5_sp = VaR_skewt(close_sp(2:end),vfGARCH_sp,-1.67958);
VaR_matrix_sp(1:end,1) = VaR_norm_5_sp;

%% VaR(5)/GARCH_NORM - Euro Stoxx 50
VaR_norm_5_eu = VaR_skewt(close_eu(2:end),vfGARCH_eu,-1.64278);
VaR_matrix_eu(1:end,1) = VaR_norm_5_eu;

%% VaR(1)/GARCH_NORM - S&P 500
VaR_norm_1_sp = VaR_skewt(close_sp(2:end),vfGARCH_sp,-2.77643);
VaR_matrix_sp(1:end,2) = VaR_norm_1_sp;

%% VaR(1)/GARCH_NORM - Euro Stoxx 50
VaR_norm_1_eu = VaR_skewt(close_eu(2:end),vfGARCH_eu,-2.64042);
VaR_matrix_eu(1:end,2) = VaR_norm_1_eu;


%% VaR(5)/EGARCH_NORM - S&P 500
VaR_E_norm_5_sp = VaR_skewt(close_sp(2:end),vfE_sp,-1.7147);
VaR_matrix_sp(1:end,3) = VaR_E_norm_5_sp;

%% VaR(5)/EGARCH_NORM - Euro Stoxx 50
VaR_E_norm_5_eu = VaR_skewt(close_eu(2:end),vfE_eu,-1.67642);
VaR_matrix_eu(1:end,3) = VaR_E_norm_5_eu;

%% VaR(1)/EGARCH_NORM - S&P 500
VaR_E_norm_1_sp = VaR_skewt(close_sp(2:end),vfE_sp,-2.81505);
VaR_matrix_sp(1:end,4) = VaR_E_norm_1_sp;

%% VaR(1)/EGARCH_NORM - Euro Stoxx 50
VaR_E_norm_1_eu = VaR_skewt(close_eu(2:end),vfE_eu,-2.62086);
VaR_matrix_eu(1:end,4) = VaR_E_norm_1_eu;


%% VaR(5)/GJR-GARCH_NORM - S&P 500
VaR_GJR_norm_5_sp = VaR_skewt(close_sp(2:end),vfGJR_sp,-1.70917);
VaR_matrix_sp(1:end,5) = VaR_GJR_norm_5_sp;

%% VaR(5)/GJR-GARCH_NORM - Euro Stoxx 50
VaR_GJR_norm_5_eu = VaR_skewt(close_eu(2:end),vfGJR_eu,-1.66263);
VaR_matrix_eu(1:end,5) = VaR_GJR_norm_5_eu;

%% VaR(1)/GJR-GARCH_NORM - S&P 500
VaR_GJR_norm_1_sp = VaR_skewt(close_sp(2:end),vfGJR_sp,-2.78965);
VaR_matrix_sp(1:end,6) = VaR_GJR_norm_1_sp;

%% VaR(1)/GJR-GARCH_NORM - Euro Stoxx 50
VaR_GJR_norm_1_eu = VaR_skewt(close_eu(2:end),vfGJR_eu,-2.61984);
VaR_matrix_eu(1:end,6) = VaR_GJR_norm_1_eu;


%% Tabulate
VaR_table_sp = array2table(VaR_matrix_sp, ...
    'VariableNames',{'VaR(5) GARCH-NORM','VaR(1) GARCH-NORM',...
    'VaR(5) EGARCH-NORM','VaR(1) EGARCH-NORM',...
    'VaR(5) GJR-GARCH-NORM', 'VaR(1) GJR-GARCH-NORM'});
```

```
VaR_table_eu = array2table(VaR_matrix_eu, ...
    'VariableNames',{'VaR(5) GARCH-NORM','VaR(1) GARCH-NORM',...
    'VaR(5) EGARCH-NORM','VaR(1) EGARCH-NORM',...
    'VaR(5) GJR-GARCH-NORM', 'VaR(1) GJR-GARCH-NORM'});

%writetable(VaR_table_sp, 'VaR_table_sp_normal.csv');
%writetable(VaR_table_eu, 'VaR_table_eu_normal.csv');

%% Calculate Violations - S&P 500
v_GARCH_5_sp = violation(VaR_norm_5_sp, y_sp(1502:end));
v_GARCH_1_sp = violation(VaR_norm_1_sp, y_sp(1502:end));

v_E_5_sp = violation(VaR_E_norm_5_sp, y_sp(1502:end));
v_E_1_sp = violation(VaR_E_norm_1_sp, y_sp(1502:end));

v_GJR_5_sp = violation(VaR_GJR_norm_5_sp, y_sp(1502:end));
v_GJR_1_sp = violation(VaR_GJR_norm_1_sp, y_sp(1502:end));


%% Calculate Violations - Euro Stoxx 50
v_GARCH_5_eu = violation(VaR_norm_5_eu, y_eu(1502:end));
v_GARCH_1_eu = violation(VaR_norm_1_eu, y_eu(1502:end));

v_E_5_eu = violation(VaR_E_norm_5_eu, y_eu(1502:end));
v_E_1_eu = violation(VaR_E_norm_1_eu, y_eu(1502:end));

v_GJR_5_eu = violation(VaR_GJR_norm_5_eu, y_eu(1502:end));
v_GJR_1_eu = violation(VaR_GJR_norm_1_eu, y_eu(1502:end));



%% Bernoulli Coverage Tests - S&P 500
bern_GARCH_5_sp = bern_test(0.05,v_GARCH_5_sp);
bern_GARCH_1_sp = bern_test(0.01,v_GARCH_1_sp);

bern_E_5_sp = bern_test(0.05,v_E_5_sp);
bern_E_1_sp = bern_test(0.01,v_E_1_sp);

bern_GJR_5_sp = bern_test(0.05,v_GJR_5_sp);
bern_GJR_1_sp = bern_test(0.01,v_GJR_1_sp);

%% Bernoulli Coverage Tests - Euro Stoxx 50
bern_GARCH_5_eu = bern_test(0.05,v_GARCH_5_eu);
bern_GARCH_1_eu = bern_test(0.01,v_GARCH_1_eu);

bern_E_5_eu = bern_test(0.05,v_E_5_eu);
bern_E_1_eu = bern_test(0.01,v_E_1_eu);

bern_GJR_5_eu = bern_test(0.05,v_GJR_5_eu);
bern_GJR_1_eu = bern_test(0.01,v_GJR_1_eu);

%% Independence Tests - S&P 500
ind_GARCH_5_sp = ind_test(v_GARCH_5_sp);
ind_GARCH_1_sp = ind_test(v_GARCH_1_sp);

ind_E_5_sp = ind_test(v_E_5_sp);
ind_E_1_sp = ind_test(v_E_1_sp);

ind_GJR_5_sp = ind_test(v_GJR_5_sp);
ind_GJR_1_sp = ind_test(v_GJR_1_sp);
```

```
%% Independence Tests - Euro Stoxx 50

ind_GARCH_5_eu = ind_test(v_GARCH_5_eu);
ind_GARCH_1_eu = ind_test(v_GARCH_1_eu);

ind_E_5_eu = ind_test(v_E_5_eu);
ind_E_1_eu = ind_test(v_E_1_eu);

ind_GJR_5_eu = ind_test(v_GJR_5_eu);
ind_GJR_1_eu = ind_test(v_GJR_1_eu);

%% Tabulate
% S&P 500
bern_matrix_sp(1:length(bern_GARCH_5_sp),1) = bern_GARCH_5_sp;
bern_matrix_sp(1:length(bern_GARCH_1_sp),2) = bern_GARCH_1_sp;

bern_matrix_sp(1:length(bern_E_5_sp),3) = bern_E_5_sp;
bern_matrix_sp(1:length(bern_E_1_sp),4) = bern_E_1_sp;

bern_matrix_sp(1:length(bern_GJR_5_sp),5) = bern_GJR_5_sp;
bern_matrix_sp(1:length(bern_GJR_1_sp),6) = bern_GJR_1_sp;

% Euro Stoxx 50
bern_matrix_eu(1:length(bern_GARCH_5_eu),1) = bern_GARCH_5_eu;
bern_matrix_eu(1:length(bern_GARCH_1_eu),2) = bern_GARCH_1_eu;

bern_matrix_eu(1:length(bern_E_5_eu),3) = bern_E_5_eu;
bern_matrix_eu(1:length(bern_E_1_eu),4) = bern_E_1_eu;

bern_matrix_eu(1:length(bern_GJR_5_eu),5) = bern_GJR_5_eu;
bern_matrix_eu(1:length(bern_GJR_1_eu),6) = bern_GJR_1_eu;

bern_table_sp = array2table(bern_matrix_sp, ...
    'VariableNames',{'bern_GARCH_5_sp','bern_GARCH_1_sp',...
    'bern_E_5_sp','bern_E_1_sp',...
    'bern_GJR_5_sp', 'bern_GJR_1_sp'});

bern_table_eu = array2table(bern_matrix_eu, ...
    'VariableNames',{'bern_GARCH_5_eu','bern_GARCH_1_eu',...
    'bern_E_5_eu','bern_E_1_eu',...
    'bern_GJR_5_eu', 'bern_GJR_1_eu'});

writetable(bern_table_sp, 'bern_table_sp.csv');
writetable(bern_table_eu, 'bern_table_eu.csv');


%% Tabulate
% S&P 500
ind_matrix_sp(1:length(ind_GARCH_5_sp),1) = ind_GARCH_5_sp;
ind_matrix_sp(1:length(ind_GARCH_1_sp),2) = ind_GARCH_1_sp;

ind_matrix_sp(1:length(ind_E_5_sp),3) = ind_E_5_sp;
ind_matrix_sp(1:length(ind_E_1_sp),4) = ind_E_1_sp;

ind_matrix_sp(1:length(ind_GJR_5_sp),5) = ind_GJR_5_sp;
ind_matrix_sp(1:length(ind_GJR_1_sp),6) = ind_GJR_1_sp;

% Euro Stoxx 50
ind_matrix_eu(1:length(ind_GARCH_5_eu),1) = ind_GARCH_5_eu;
```

```
ind_matrix_eu(1:length(ind_GARCH_1_eu),2) = ind_GARCH_1_eu;

ind_matrix_eu(1:length(ind_E_5_eu),3) = ind_E_5_eu;
ind_matrix_eu(1:length(ind_E_1_eu),4) = ind_E_1_eu;

ind_matrix_eu(1:length(ind_GJR_5_eu),5) = ind_GJR_5_eu;
ind_matrix_eu(1:length(ind_GJR_1_eu),6) = ind_GJR_1_eu;

ind_table_sp = array2table(ind_matrix_sp, ...
    'VariableNames',{'ind_GARCH_5_sp','ind_GARCH_1_sp',...
    'ind_E_5_sp','ind_E_1_sp',...
    'ind_GJR_5_sp', 'ind_GJR_sp'});

ind_table_eu = array2table(ind_matrix_eu, ...
    'VariableNames',{'ind_GARCH_5_eu','ind_GARCH_1_eu',...
    'ind_E_5_eu','ind_E_1_eu',...
    'ind_GJR_5_eu', 'ind_GJR_1_eu'});

writetable(ind_table_sp, 'ind_table_sp.csv');
writetable(ind_table_eu, 'ind_table_eu.csv');


%% Joint test
% S&P 500
joint_GARCH_5_sp = bern_GARCH_5_sp + ind_GARCH_5_sp;
joint_GARCH_1_sp = bern_GARCH_1_sp + ind_GARCH_1_sp;

joint_E_5_sp = bern_E_5_sp + ind_E_5_sp;
joint_E_1_sp = bern_E_1_sp + ind_E_1_sp;

joint_GJR_5_sp = bern_GJR_5_sp + ind_GJR_5_sp;
joint_GJR_1_sp = bern_GJR_1_sp + ind_GJR_1_sp;

% Euro Stoxx 50
joint_GARCH_5_eu = bern_GARCH_5_eu + ind_GARCH_5_eu;
joint_GARCH_1_eu = bern_GARCH_1_eu + ind_GARCH_1_eu;

joint_E_5_eu = bern_E_5_eu + ind_E_5_eu;
joint_E_1_eu = bern_E_1_eu + ind_E_1_eu;

joint_GJR_5_eu = bern_GJR_5_eu + ind_GJR_5_eu;
joint_GJR_1_eu = bern_GJR_1_eu + ind_GJR_1_eu;

%% Tabulate
% S&P 500
joint_matrix_sp(1:length(ind_GARCH_5_sp),1) = joint_GARCH_5_sp;
joint_matrix_sp(1:length(ind_GARCH_5_sp),2) = joint_GARCH_1_sp;

joint_matrix_sp(1:length(ind_GARCH_5_sp),3) = joint_E_5_sp;
joint_matrix_sp(1:length(ind_GARCH_5_sp),4) = joint_E_1_sp;

joint_matrix_sp(1:length(ind_GARCH_5_sp),5) = joint_GJR_5_sp;
joint_matrix_sp(1:length(ind_GARCH_5_sp),6) = joint_GJR_1_sp;

% Euro Stoxx 50
joint_matrix_eu(1:length(ind_GARCH_5_sp),1) = joint_GARCH_5_eu;
joint_matrix_eu(1:length(ind_GARCH_5_sp),2) = joint_GARCH_1_eu;

joint_matrix_eu(1:length(ind_GARCH_5_sp),3) = joint_E_5_eu;
joint_matrix_eu(1:length(ind_GARCH_5_sp),4) = joint_E_1_eu;
```

```
joint_matrix_eu(1:length(ind_GARCH_5_sp),5) = joint_GJR_5_eu;
joint_matrix_eu(1:length(ind_GARCH_5_sp),6) = joint_GJR_1_eu;

joint_table_sp = array2table(joint_matrix_sp, ...
    'VariableNames',{'joint_GARCH_5_sp','joint_GARCH_1_sp',...
    'joint_E_5_sp','joint_E_1_sp',...
    'joint_GJR_5_sp', 'joint_GJR_1_sp'});

joint_table_eu = array2table(joint_matrix_eu, ...
    'VariableNames',{'joint_GARCH_5_eu','joint_GARCH_1_eu',...
    'joint_E_5_eu','joint_E_1_eu',...
    'joint_GJR_5_eu', 'joint_GJR_1_eu'});

writetable(joint_table_sp, 'joint_table_sp.csv');
writetable(joint_table_eu, 'joint_table_eu.csv');


%% Violations
exp_vio_5_sp = round(0.05*length(v_GARCH_5_sp))
exp_vio_1_sp = round(0.01*length(v_GARCH_1_sp))

exp_vio_5_eu = round(0.05*length(v_GARCH_5_eu))
exp_vio_1_eu = round(0.01*length(v_GARCH_1_eu))


vio_5_G_sp = sum(v_GARCH_5_sp);
vio_1_G_sp = sum(v_GARCH_1_sp);

vio_5_E_sp = sum(v_E_5_sp);
vio_1_E_sp = sum(v_E_1_sp);

vio_5_GJR_sp = sum(v_GJR_5_sp);
vio_1_GJR_sp = sum(v_GJR_1_sp);


vio_5_G_eu = sum(v_GARCH_5_eu);
vio_1_G_eu = sum(v_GARCH_1_eu);

vio_5_E_eu = sum(v_E_5_eu);
vio_1_E_eu = sum(v_E_1_eu);

vio_5_GJR_eu = sum(v_GJR_5_eu);
vio_1_GJR_eu = sum(v_GJR_1_eu);
```

VaR for Bayes

```
clear;
clc;
format longG

file_sp                     = 'sp500.csv';
file_eu                     = 'stoxx50.csv';

% Imports data and converts to datetime & log returns
[dates_sp,df_sp,close_sp,y_sp]       = process_data(file_sp);
[dates_eu,df_eu,close_eu,y_eu]       = process_data(file_eu);

y_sp = y_sp;
y_eu = y_eu;
```

```
dates_sp_ret = dates_sp(2:end);
close_sp_ret = close_sp(2:end);

dates_eu_ret = dates_eu(2:end);
close_eu_ret = close_eu(2:end);

WT_sp = 1015;
WT_eu = 1007;

VaR_matrix_sp = NaN(1015,6);
VaR_matrix_eu = NaN(1007,6);


%% SP
% Import
varForecastsPython_sp = csvread('vf_sp.csv',1,1);
varForecastsBAYES_sp = csvread('vf_sp_bays.csv',1,1);
% GARCH(1,1)
vfGARCH_sp = varForecastsBAYES_sp(1501:end,1);

% EGARCH(1,1)
vfE_sp = varForecastsPython_sp(1501:end,6);

% GJRGARCH(1,1)
vfGJR_sp = varForecastsPython_sp(1501:end,9);

%% EU

varForecastsPython_eu = csvread('vf_eu.csv',1,1);
varForecastsBAYES_eu = csvread('vf_eu_bays.csv',1,1);
% GARCH(1,1)

vfGARCH_eu = varForecastsBAYES_eu(1501:end,1);

% EGARCH(1,1)
vfE_eu = varForecastsPython_eu(1501:end,6);

% GJRGARCH(1,1)
vfGJR_eu = varForecastsPython_eu(1501:end,9);



%% VAR crap
%% VaR(5)/GARCH_NORM - S&P 500
VaR_norm_5_sp = VaR_skewt(close_sp(2:end),vfGARCH_sp,-1.87720990398151* sqrt( (7.45828-2)/7.45828 ));
VaR_matrix_sp(1:end,1) = VaR_norm_5_sp;

%% VaR(5)/GARCH_NORM - Euro Stoxx 50
VaR_norm_5_eu = VaR_skewt(close_eu(2:end),vfGARCH_eu,-1.90427648250246* sqrt( (6.7707-2)/6.7707 ));
VaR_matrix_eu(1:end,1) = VaR_norm_5_eu;

%% VaR(1)/GARCH_NORM - S&P 500
VaR_norm_1_sp = VaR_skewt(close_sp(2:end),vfGARCH_sp,-2.9473346931348* sqrt( (7.45828-2)/7.45828 ));
VaR_matrix_sp(1:end,2) = VaR_norm_1_sp;

%% VaR(1)/GARCH_NORM - Euro Stoxx 50
VaR_norm_1_eu = VaR_skewt(close_eu(2:end),vfGARCH_eu,-3.02646613140637* sqrt( (6.7707-2)/6.7707 ));
VaR_matrix_eu(1:end,2) = VaR_norm_1_eu;
```

```
%% VaR(5)/EGARCH_NORM - S&P 500
VaR_E_norm_5_sp = VaR_skewt(close_sp(2:end),vfE_sp,-1.60137);
VaR_matrix_sp(1:end,3) = VaR_E_norm_5_sp;

%% VaR(5)/EGARCH_NORM - Euro Stoxx 50
VaR_E_norm_5_eu = VaR_skewt(close_eu(2:end),vfE_eu,-1.61634);
VaR_matrix_eu(1:end,3) = VaR_E_norm_5_eu;

%% VaR(1)/EGARCH_NORM - S&P 500
VaR_E_norm_1_sp = VaR_skewt(close_sp(2:end),vfE_sp,-2.53332);
VaR_matrix_sp(1:end,4) = VaR_E_norm_1_sp;

%% VaR(1)/EGARCH_NORM - Euro Stoxx 50
VaR_E_norm_1_eu = VaR_skewt(close_eu(2:end),vfE_eu,-2.48934);
VaR_matrix_eu(1:end,4) = VaR_E_norm_1_eu;


%% VaR(5)/GJR-GARCH_NORM - S&P 500
VaR_GJR_norm_5_sp = VaR_skewt(close_sp(2:end),vfGJR_sp,-1.60365);
VaR_matrix_sp(1:end,5) = VaR_GJR_norm_5_sp;

%% VaR(5)/GJR-GARCH_NORM - Euro Stoxx 50
VaR_GJR_norm_5_eu = VaR_skewt(close_eu(2:end),vfGJR_eu,-1.61175);
VaR_matrix_eu(1:end,5) = VaR_GJR_norm_5_eu;

%% VaR(1)/GJR-GARCH_NORM - S&P 500
VaR_GJR_norm_1_sp = VaR_skewt(close_sp(2:end),vfGJR_sp,-2.52747);
VaR_matrix_sp(1:end,6) = VaR_GJR_norm_1_sp;

%% VaR(1)/GJR-GARCH_NORM - Euro Stoxx 50
VaR_GJR_norm_1_eu = VaR_skewt(close_eu(2:end),vfGJR_eu,-2.50433);
VaR_matrix_eu(1:end,6) = VaR_GJR_norm_1_eu;


%% Tabulate
VaR_table_sp = array2table(VaR_matrix_sp, ...
    'VariableNames',{'VaR(5) GARCH-NORM','VaR(1) GARCH-NORM',...
    'VaR(5) EGARCH-NORM','VaR(1) EGARCH-NORM',...
    'VaR(5) GJR-GARCH-NORM', 'VaR(1) GJR-GARCH-NORM'});

VaR_table_eu = array2table(VaR_matrix_eu, ...
    'VariableNames',{'VaR(5) GARCH-NORM','VaR(1) GARCH-NORM',...
    'VaR(5) EGARCH-NORM','VaR(1) EGARCH-NORM',...
    'VaR(5) GJR-GARCH-NORM', 'VaR(1) GJR-GARCH-NORM'});

%writetable(VaR_table_sp, 'VaR_table_sp_normal.csv');
%writetable(VaR_table_eu, 'VaR_table_eu_normal.csv');

%% Calculate Violations - S&P 500
v_GARCH_5_sp = violation(VaR_norm_5_sp, y_sp(1502:end));
v_GARCH_1_sp = violation(VaR_norm_1_sp, y_sp(1502:end));

v_E_5_sp = violation(VaR_E_norm_5_sp, y_sp(1502:end));
v_E_1_sp = violation(VaR_E_norm_1_sp, y_sp(1502:end));

v_GJR_5_sp = violation(VaR_GJR_norm_5_sp, y_sp(1502:end));
v_GJR_1_sp = violation(VaR_GJR_norm_1_sp, y_sp(1502:end));


%% Calculate Violations - Euro Stoxx 50
```

```
v_GARCH_5_eu = violation(VaR_norm_5_eu, y_eu(1502:end));
v_GARCH_1_eu = violation(VaR_norm_1_eu, y_eu(1502:end));

v_E_5_eu = violation(VaR_E_norm_5_eu, y_eu(1502:end));
v_E_1_eu = violation(VaR_E_norm_1_eu, y_eu(1502:end));

v_GJR_5_eu = violation(VaR_GJR_norm_5_eu, y_eu(1502:end));
v_GJR_1_eu = violation(VaR_GJR_norm_1_eu, y_eu(1502:end));



%% Bernoulli Coverage Tests - S&P 500
bern_GARCH_5_sp = bern_test(0.05,v_GARCH_5_sp);
bern_GARCH_1_sp = bern_test(0.01,v_GARCH_1_sp);

bern_E_5_sp = bern_test(0.05,v_E_5_sp);
bern_E_1_sp = bern_test(0.01,v_E_1_sp);

bern_GJR_5_sp = bern_test(0.05,v_GJR_5_sp);
bern_GJR_1_sp = bern_test(0.01,v_GJR_1_sp);

%% Bernoulli Coverage Tests - Euro Stoxx 50
bern_GARCH_5_eu = bern_test(0.05,v_GARCH_5_eu);
bern_GARCH_1_eu = bern_test(0.01,v_GARCH_1_eu);

bern_E_5_eu = bern_test(0.05,v_E_5_eu);
bern_E_1_eu = bern_test(0.01,v_E_1_eu);

bern_GJR_5_eu = bern_test(0.05,v_GJR_5_eu);
bern_GJR_1_eu = bern_test(0.01,v_GJR_1_eu);

%% Independence Tests - S&P 500
ind_GARCH_5_sp = ind_test(v_GARCH_5_sp);
ind_GARCH_1_sp = ind_test(v_GARCH_1_sp);

ind_E_5_sp = ind_test(v_E_5_sp);
ind_E_1_sp = ind_test(v_E_1_sp);

ind_GJR_5_sp = ind_test(v_GJR_5_sp);
ind_GJR_1_sp = ind_test(v_GJR_1_sp);

%% Independence Tests - Euro Stoxx 50

ind_GARCH_5_eu = ind_test(v_GARCH_5_eu);
ind_GARCH_1_eu = ind_test(v_GARCH_1_eu);

ind_E_5_eu = ind_test(v_E_5_eu);
ind_E_1_eu = ind_test(v_E_1_eu);

ind_GJR_5_eu = ind_test(v_GJR_5_eu);
ind_GJR_1_eu = ind_test(v_GJR_1_eu);

%% Tabulate
% S&P 500
bern_matrix_sp(1:length(bern_GARCH_5_sp),1) = bern_GARCH_5_sp;
bern_matrix_sp(1:length(bern_GARCH_1_sp),2) = bern_GARCH_1_sp;

bern_matrix_sp(1:length(bern_E_5_sp),3) = bern_E_5_sp;
bern_matrix_sp(1:length(bern_E_1_sp),4) = bern_E_1_sp;
```

```
bern_matrix_sp(1:length(bern_GJR_5_sp),5) = bern_GJR_5_sp;
bern_matrix_sp(1:length(bern_GJR_1_sp),6) = bern_GJR_1_sp;

% Euro Stoxx 50
bern_matrix_eu(1:length(bern_GARCH_5_eu),1) = bern_GARCH_5_eu;
bern_matrix_eu(1:length(bern_GARCH_1_eu),2) = bern_GARCH_1_eu;

bern_matrix_eu(1:length(bern_E_5_eu),3) = bern_E_5_eu;
bern_matrix_eu(1:length(bern_E_1_eu),4) = bern_E_1_eu;

bern_matrix_eu(1:length(bern_GJR_5_eu),5) = bern_GJR_5_eu;
bern_matrix_eu(1:length(bern_GJR_1_eu),6) = bern_GJR_1_eu;

bern_table_sp = array2table(bern_matrix_sp, ...
    'VariableNames',{'bern_GARCH_5_sp','bern_GARCH_1_sp',...
    'bern_E_5_sp','bern_E_1_sp',...
    'bern_GJR_5_sp', 'bern_GJR_1_sp'});

bern_table_eu = array2table(bern_matrix_eu, ...
    'VariableNames',{'bern_GARCH_5_eu','bern_GARCH_1_eu',...
    'bern_E_5_eu','bern_E_1_eu',...
    'bern_GJR_5_eu', 'bern_GJR_1_eu'});

writetable(bern_table_sp, 'bern_table_sp.csv');
writetable(bern_table_eu, 'bern_table_eu.csv');


%% Tabulate
% S&P 500
ind_matrix_sp(1:length(ind_GARCH_5_sp),1) = ind_GARCH_5_sp;
ind_matrix_sp(1:length(ind_GARCH_1_sp),2) = ind_GARCH_1_sp;

ind_matrix_sp(1:length(ind_E_5_sp),3) = ind_E_5_sp;
ind_matrix_sp(1:length(ind_E_1_sp),4) = ind_E_1_sp;

ind_matrix_sp(1:length(ind_GJR_5_sp),5) = ind_GJR_5_sp;
ind_matrix_sp(1:length(ind_GJR_1_sp),6) = ind_GJR_1_sp;

% Euro Stoxx 50
ind_matrix_eu(1:length(ind_GARCH_5_eu),1) = ind_GARCH_5_eu;
ind_matrix_eu(1:length(ind_GARCH_1_eu),2) = ind_GARCH_1_eu;

ind_matrix_eu(1:length(ind_E_5_eu),3) = ind_E_5_eu;
ind_matrix_eu(1:length(ind_E_1_eu),4) = ind_E_1_eu;

ind_matrix_eu(1:length(ind_GJR_5_eu),5) = ind_GJR_5_eu;
ind_matrix_eu(1:length(ind_GJR_1_eu),6) = ind_GJR_1_eu;

ind_table_sp = array2table(ind_matrix_sp, ...
    'VariableNames',{'ind_GARCH_5_sp','ind_GARCH_1_sp',...
    'ind_E_5_sp','ind_E_1_sp',...
    'ind_GJR_5_sp', 'ind_GJR_sp'});

ind_table_eu = array2table(ind_matrix_eu, ...
    'VariableNames',{'ind_GARCH_5_eu','ind_GARCH_1_eu',...
    'ind_E_5_eu','ind_E_1_eu',...
    'ind_GJR_5_eu', 'ind_GJR_1_eu'});

writetable(ind_table_sp, 'ind_table_sp.csv');
writetable(ind_table_eu, 'ind_table_eu.csv');
```

```
%% Joint test
% S&P 500
joint_GARCH_5_sp = bern_GARCH_5_sp + ind_GARCH_5_sp;
joint_GARCH_1_sp = bern_GARCH_1_sp + ind_GARCH_1_sp;

joint_E_5_sp = bern_E_5_sp + ind_E_5_sp;
joint_E_1_sp = bern_E_1_sp + ind_E_1_sp;

joint_GJR_5_sp = bern_GJR_5_sp + ind_GJR_5_sp;
joint_GJR_1_sp = bern_GJR_1_sp + ind_GJR_1_sp;

% Euro Stoxx 50
joint_GARCH_5_eu = bern_GARCH_5_eu + ind_GARCH_5_eu;
joint_GARCH_1_eu = bern_GARCH_1_eu + ind_GARCH_1_eu;

joint_E_5_eu = bern_E_5_eu + ind_E_5_eu;
joint_E_1_eu = bern_E_1_eu + ind_E_1_eu;

joint_GJR_5_eu = bern_GJR_5_eu + ind_GJR_5_eu;
joint_GJR_1_eu = bern_GJR_1_eu + ind_GJR_1_eu;

%% Tabulate
% S&P 500
joint_matrix_sp(1:length(ind_GARCH_5_sp),1) = joint_GARCH_5_sp;
joint_matrix_sp(1:length(ind_GARCH_5_sp),2) = joint_GARCH_1_sp;

joint_matrix_sp(1:length(ind_GARCH_5_sp),3) = joint_E_5_sp;
joint_matrix_sp(1:length(ind_GARCH_5_sp),4) = joint_E_1_sp;

joint_matrix_sp(1:length(ind_GARCH_5_sp),5) = joint_GJR_5_sp;
joint_matrix_sp(1:length(ind_GARCH_5_sp),6) = joint_GJR_1_sp;

% Euro Stoxx 50
joint_matrix_eu(1:length(ind_GARCH_5_sp),1) = joint_GARCH_5_eu;
joint_matrix_eu(1:length(ind_GARCH_5_sp),2) = joint_GARCH_1_eu;

joint_matrix_eu(1:length(ind_GARCH_5_sp),3) = joint_E_5_eu;
joint_matrix_eu(1:length(ind_GARCH_5_sp),4) = joint_E_1_eu;

joint_matrix_eu(1:length(ind_GARCH_5_sp),5) = joint_GJR_5_eu;
joint_matrix_eu(1:length(ind_GARCH_5_sp),6) = joint_GJR_1_eu;

joint_table_sp = array2table(joint_matrix_sp, ...
    'VariableNames',{'joint_GARCH_5_sp','joint_GARCH_1_sp',...
    'joint_E_5_sp','joint_E_1_sp',...
    'joint_GJR_5_sp', 'joint_GJR_1_sp'});

joint_table_eu = array2table(joint_matrix_eu, ...
    'VariableNames',{'joint_GARCH_5_eu','joint_GARCH_1_eu',...
    'joint_E_5_eu','joint_E_1_eu',...
    'joint_GJR_5_eu', 'joint_GJR_1_eu'});

writetable(joint_table_sp, 'joint_table_sp.csv');
writetable(joint_table_eu, 'joint_table_eu.csv');


%% Violations
exp_vio_5_sp = round(0.05*length(v_GARCH_5_sp));
```

```
exp_vio_1_sp = round(0.01*length(v_GARCH_1_sp));

exp_vio_5_eu = round(0.05*length(v_GARCH_5_eu));
exp_vio_1_eu = round(0.01*length(v_GARCH_1_eu));


vio_5_G_sp = sum(v_GARCH_5_sp);
vio_1_G_sp = sum(v_GARCH_1_sp);

vio_5_E_sp = sum(v_E_5_sp);
vio_1_E_sp = sum(v_E_1_sp);

vio_5_GJR_sp = sum(v_GJR_5_sp);
vio_1_GJR_sp = sum(v_GJR_1_sp);


vio_5_G_eu = sum(v_GARCH_5_eu);
vio_1_G_eu = sum(v_GARCH_1_eu);

vio_5_E_eu = sum(v_E_5_eu);
vio_1_E_eu = sum(v_E_1_eu);

vio_5_GJR_eu = sum(v_GJR_5_eu);
vio_1_GJR_eu = sum(v_GJR_1_eu);
```

Misc functions

```
function [dates,df,closePrices,y]    = process_data(file)

df                      = importdata(file);
dates                   = datetime(datestr(df.textdata));
closePrices             = df.data(:,5);
y                       = diff(log(closePrices));
end


function [v] = violation(VaR_forecast_vector, returns)

v = NaN(length(VaR_forecast_vector),1);
for i = 1:length(VaR_forecast_vector)
    if returns(i) <= VaR_forecast_vector(i)
        v(i,1) = 1;
    else
        v(i,1) = 0;
    end
end

end


function [VaR_norm] = VaR_skewt(close,variance,q)

T = length(close);
WE = 1500;
WT = T-WE;

VaR_norm = NaN(WT,1);
for i = 2:WT
    VaR_norm(i,1) = q*sqrt(variance(i));
end
```

```
VaR_norm = VaR_norm(2:end);


function [VaR_norm] = VaR_norm(close,variance,p)

T = length(close);
WE = 1500;
WT = T-WE;

VaR_norm = NaN(WT,1);
for i = 1:WT
    VaR_norm(i,1) = norminv(p)*sqrt(variance(i));
end

VaR_norm = VaR_norm(2:end);


% Independence coverage test in Matlab
function res=ind_test(V)
  T=length(V);
  J=zeros(T,4);
  for i = 2:T
    J(i,1)=V(i-1)==0 & V(i)==0;
    J(i,2)=V(i-1)==0 & V(i)==1;
    J(i,3)=V(i-1)==1 & V(i)==0;
    J(i,4)=V(i-1)==1 & V(i)==1;
  end
  V_00=sum(J(:,1));
  V_01=sum(J(:,2));
  V_10=sum(J(:,3));
  V_11=sum(J(:,4));
  p_00=V_00/(V_00+V_01);
  p_01=V_01/(V_00+V_01);
  p_10=V_10/(V_10+V_11);
  p_11=V_11/(V_10+V_11);
  hat_p=(V_01+V_11)/(V_00+V_01+V_10+V_11);
  a=(1-hat_p)^(V_00+V_10)*(hat_p)^(V_01+V_11);
  b=(p_00)^(V_00)*(p_01)^(V_01)*(p_10)^(V_10)*p_11^(V_11);
  res= -2*log(a/b);
end



% Bernoulli coverage test in Matlab
function res=bern_test(p,v)
  a=p^(sum(v))*(1-p)^(length(v)-sum(v));
  b=(sum(v)/length(v))^(sum(v))*(1-(sum(v)/length(v)))^(length(v)-sum(v));
  res=-2*log(a/b);
end
```

Bayes estimation of the variances, based on R code

```
import scipy as sc
import numpy as np
import pandas as pd
import datetime as dt
import matplotlib.pyplot as plt

from arch import arch_model
```

```
filename1 = '^STOXX50E.csv'
filename2 = '^GSPC.csv'

scaling = 100

df1 = pd.read_csv(filename1, index_col = 0) #stoxx
df2 = pd.read_csv(filename2, index_col = 0) #sp500

adjClose1 = df1['Adj Close'] #stoxx
adjClose2 = df2['Adj Close'] #sp500

y1 = np.diff(np.log(adjClose1))*scaling  #stoxx
y2 = np.diff(np.log(adjClose2))*scaling #sp500


def forecast_stoxx(y1):
    mdl1 = arch_model(y1, vol = 'GARCH', p=1, q=1, dist = 't', rescale = False)
    estMdl1 = mdl1.fix([0, 0.07714, 0.10801, 0.85755, 6.7707])
    forecast1 = estMdl1.forecast(horizon=1, start=1500)
    vf_stoxx = np.square(((np.sqrt(forecast1.variance))/scaling))
    data1 = pd.DataFrame({'vf_stoxx':[vf_stoxx]})
    return data1, vf_stoxx

data1, vf_stoxx = forecast_stoxx(y1)

def forecast_sp500(y2):
    mdl2 = arch_model(y2, vol = 'GARCH', p=1, q=1, dist = 't', rescale = False)
    estMdl2 = mdl2.fix([0, 0.0488, 0.1556, 0.8030, 7.45828])
    forecast2 = estMdl2.forecast(horizon=1, start=1500)
    vf_sp500 = np.square(((np.sqrt(forecast2.variance))/scaling))
    data2 = pd.DataFrame({'vf_sp500':[vf_sp500]})
    return data2, vf_sp500

data2, vf_sp500 = forecast_sp500(y2)

#MAE Bayesian Estimation
Scaling = 1
y1_u = np.diff(np.log(adjClose1))*Scaling  #stoxx
y2_u = np.diff(np.log(adjClose2))*Scaling #sp500

y1_u_sq = y1_u ** 2
y2_u_sq = y2_u ** 2

def mae_stoxx(vf_stoxx):
    mae_stoxx = np.subtract(vf_stoxx['h.1'][1500:2508], y1_u_sq[1500:2508])
    mae_stoxx = np.abs(mae_stoxx)
    mae_stoxx = np.mean(mae_stoxx)
    return mae_stoxx

mae_stoxx = mae_stoxx(vf_stoxx)

def mae_sp500(vf_stoxx):
    mae_sp500= np.subtract(vf_sp500['h.1'][1500:2516], y2_u_sq[1500:2516])
    mae_sp500 = np.abs(mae_sp500)
    mae_sp500 = np.mean(mae_sp500)
    return mae_sp500

mae_sp500= mae_sp500(vf_sp500)

vf_sp500.to_csv('vf_sp_bays.csv', index=True)
```

```
vf_stoxx.to_csv('vf_eu_bays.csv', index=True)



plt.plot(np.sqrt(vf_sp500))


# -*- coding: utf-8 -*-
"""
Created on Fri Jan 31 15:24:03 2020

@author: Maurits
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from arch.univariate import arch_model
import arch
import scipy as sc
# Data Sets:
sp          = '^GSPC.csv'
eu          = '^STOXX50E.csv'

### Constant Parameters and the data set you want: ###
FILE = eu
SCALING = 100

#-------------------------------------------------------------------------------------------------------------------

def main(y, model, distr, SCALING):

    y = y[1500:] #just for checking leverage effect structural breaks, usually not there
    split_date = y.size #just for checking leverage effect structural breaks, used to be 1500
    #random comment

    if model == 'garch':
        am = arch_model(y,mean='Zero', dist = distr)
        res = am.fit(last_obs=split_date)
        forecasts = res.forecast(horizon=1, start = split_date)

        std_error = res.std_err
        std_error[0] = std_error[0]/SCALING**2

        params = res.params
        params[0] = params[0]/SCALING**2

        likelihood = res.loglikelihood
        aic_crit = 2*len(params) - 2*likelihood
        bic_crit = np.log(len(forecasts.variance)) * len(params) - 2*likelihood

    #random comment 2
    elif model == 'egarch':
        am = arch_model(y,mean='Zero', vol='EGARCH',p=1, o=1, q=1, dist = distr)
        res = am.fit(last_obs=split_date)
        forecasts = res.forecast(horizon=1, start = split_date)

        std_error = res.std_err
        std_error[0] = std_error[0]/SCALING**2
```

```
        params = res.params
        params[0] = params[0]/SCALING**2

        likelihood = res.loglikelihood
        aic_crit = 2*len(params) - 2*likelihood
        bic_crit = np.log(len(forecasts.variance)) * len(params) - 2*likelihood


    elif model == 'gjr-garch':
        am = arch_model(y,mean='Zero', vol='GARCH', p=1, o=1, q=1, dist = distr)
        res = am.fit(last_obs=split_date)
        forecasts = res.forecast(horizon=1, start = split_date)

        std_error = res.std_err
        std_error[0] = std_error[0]/SCALING**2

        params = res.params
        params[0] = params[0]/SCALING**2


        likelihood = res.loglikelihood
        aic_crit = 2*len(params) - 2*likelihood
        bic_crit = np.log(len(forecasts.variance)) * len(params) - 2*likelihood
        print(res.summary())




    if distr == 'gaussian':
        q = am.distribution.ppf([0.01, 0.05], None)

    elif distr == 't':
        q = am.distribution.ppf([0.01, 0.05], res.params[-1:])

    elif distr == 'skewt':
        q = am.distribution.ppf([0.01, 0.05], res.params[-2:])



    vf = forecasts.variance
    vf = np.square( (np.sqrt(forecasts.variance) / SCALING) )



    return vf, params, aic_crit, bic_crit, likelihood,q

#-----------------------------------------------------------------------------------------------------------------------


def dmtest(e1, e2):
    d = e1 - e2
    n = len(d)
    dmean = np.mean(d)
    s2d = 0
    for i in range(0,n):
        s2d = s2d + (d[i]-dmean)**2
    s2d = s2d*(1/(n-1))
    sd = np.sqrt(s2d)
    teststat = np.sqrt(n)*((dmean-0)/sd);
    return teststat
```

```
#-------------------------------------------------------------------------------------------------------------------

def forecast_stoxx(y):
    mdl1 = arch_model(y, vol = 'GARCH', p=1, q=1, dist = 't', rescale = False)
    estMdl1 = mdl1.fix([0, 0.07714, 0.10801, 0.85755, 6.7707])  #ask walter how he changed it xdd
    forecast1 = estMdl1.forecast(horizon=1, start=1500)
    vf_stoxx = np.square(((np.sqrt(forecast1.variance))/SCALING))
    data1 = pd.DataFrame({'vf_stoxx':[vf_stoxx]})
    return data1, vf_stoxx

def forecast_sp500(y):
    mdl2 = arch_model(y, vol = 'GARCH', p=1, q=1, dist = 't', rescale = False)
    estMdl2 = mdl2.fix([0, 0.0488, 0.1556, 0.8030, 7.45828])     #ask walter how he changed it xdd
    forecast2 = estMdl2.forecast(horizon=1, start=1500)
    vf_sp500 = np.square(((np.sqrt(forecast2.variance))/SCALING))
    data2 = pd.DataFrame({'vf_sp500':[vf_sp500]})
    return data2, vf_sp500


# Possible models: 'GARCH' (default), 'ARCH', 'EGARCH', 'FIARCH' and 'HARCH'

# Possible distributions: Normal: 'normal', 'gaussian' (default)
# Students's t: 't', 'studentst'
# Skewed Student's t: 'skewstudent', 'skewt'
# Generalized Error Distribution: 'ged', 'generalized error"

###### Main Parameters ######



# Create data frame with dates
df              = pd.read_csv(FILE)
df['y']         = np.log(df['Adj Close']).diff()

y = SCALING*df['y'].dropna().to_numpy()

# if scaling is off:
#vf = np.square((np.sqrt(forecasts.variance)/1000))


# Volatility Models
garch               = 'garch'
e                   = 'egarch'
gjr                 = 'gjr-garch'

# Distributions:
normal              = 'gaussian'
student_t           = 't'
skew_student_t      = 'skewt'
ged                 = 'ged'




### Main Function: create a df of all the vfs ###

# GARCH model:
vf, garch_norm_params,\
```

```
 garch_norm_aic_crit, \
 garch_norm_bic_crit, \
 garch_norm_likelihood,_ = main(y, garch, normal, SCALING)
vf.columns = ['garch_norm']

vf['garch_t'], \
garch_t_params, \
garch_t_aic_crit, \
garch_t_bic_crit, \
garch_t_likelihood,_ = main(y, garch, student_t, SCALING)

vf['garch_skewt'], \
garch_skewt_params, \
garch_skewt_aic_crit, \
garch_skewt_bic_crit, \
garch_skewt_likelihood,_ = main(y, garch, skew_student_t, SCALING)



# EGARCH model:
vf['egarch_norm'], \
egarch_norm_params, \
egarch_norm_aic_crit, \
egarch_norm_bic_crit, \
egarch_norm_likelihood,_ = main(y, e, normal, SCALING)

vf['egarch_t'], \
egarch_t_params, \
egarch_t_aic_crit, \
egarch_t_bic_crit, \
egarch_t_likelihood,_ = main(y, e, student_t, SCALING)

vf['egarch_skewt'], \
egarch_skewt_params, \
egarch_skewt_aic_crit, \
egarch_skewt_bic_crit, \
egarch_skewt_likelihood,_ = main(y, e, skew_student_t, SCALING)



# GJR-GARCH model:
vf['gjr_norm'], \
gjr_norm_params, \
gjr_norm_aic_crit, \
gjr_norm_bic_crit, \
gjr_norm_likelihood,_ = main(y, gjr, normal, SCALING)

vf['gjr_t'], \
gjr_t_params, \
gjr_t_aic_crit, \
gjr_t_bic_crit, \
gjr_t_likelihood,_ = main(y, gjr, student_t, SCALING)

vf['gjr_skewt'], \
gjr_skewt_params, \
gjr_skewt_aic_crit, \
gjr_skewt_bic_crit, \
gjr_skewt_likelihood,_ = main(y, gjr, skew_student_t, SCALING)
```

```
# Date
vf['Date'] = df['Date'].shift(-1)
vf['Date'] = pd.to_datetime(vf['Date'])
vf.set_index('Date', inplace=True)

#%%

#garch
y = y/1000
ymean = np.mean(y)
eps = y
#scale all params to the right versions

#%% make all log L vectors
G11vectorSPnormal = arch.univariate.Normal.loglikelihood('Normal', parameters=garch_norm_params, resids=eps[15
G11vectorSPstudent = arch.univariate.StudentsT.loglikelihood('Standardized Student\'s t', parameters=np.array(
G11vectorSPskewt = arch.univariate.SkewStudent.loglikelihood(arch.univariate.SkewStudent(), parameters=np.arra

#egarch
GEvectorSPnormal = arch.univariate.Normal.loglikelihood('Normal', parameters=egarch_norm_params, resids=eps[15
GEvectorSPstudent = arch.univariate.StudentsT.loglikelihood('Standardized Student\'s t', parameters=np.array([
GEvectorSPskewt = arch.univariate.SkewStudent.loglikelihood(arch.univariate.SkewStudent(), parameters=np.array


##gjr
GJRvectorSPnormal = arch.univariate.Normal.loglikelihood('Normal', parameters=gjr_norm_params, resids=eps[1500
GJRvectorSPstudent = arch.univariate.StudentsT.loglikelihood('Standardized Student\'s t', parameters=np.array(
GJRvectorSPskewt = arch.univariate.SkewStudent.loglikelihood(arch.univariate.SkewStudent(), parameters=np.arra


#which vector 2 use
if FILE == eu:
    data1, vf_stoxx = forecast_stoxx(y)
    Baysian_vector = arch.univariate.StudentsT.loglikelihood('Standardized Student\'s t', parameters=np.array(
    Baysian_vector['Date'] = df['Date'].shift(-1)
    Baysian_vector['Date'] = pd.to_datetime(Baysian_vector['Date'])
    Baysian_vector.set_index('Date', inplace=True)
else:
    data1, vf_sp500 = forecast_sp500(y)
    Baysian_vector = arch.univariate.StudentsT.loglikelihood('Standardized Student\'s t', parameters=np.array(
    Baysian_vector['Date'] = df['Date'].shift(-1)
    Baysian_vector['Date'] = pd.to_datetime(Baysian_vector['Date'])
    Baysian_vector.set_index('Date', inplace=True)


#%%
## some logl dm testing-main
#dm = []
##row one
#dm = np.append(dm,dmtest(G11vectorSPnormal, G11vectorSPstudent))
#dm = np.append(dm,dmtest(G11vectorSPnormal, G11vectorSPskewt))
#dm = np.append(dm,dmtest(G11vectorSPnormal, GEvectorSPnormal))
#dm = np.append(dm,dmtest(G11vectorSPnormal, GEvectorSPstudent))
#dm = np.append(dm,dmtest(G11vectorSPnormal, GEvectorSPskewt))
#dm = np.append(dm,dmtest(G11vectorSPnormal, GJRvectorSPnormal))
#dm = np.append(dm,dmtest(G11vectorSPnormal, GJRvectorSPstudent))
#dm = np.append(dm,dmtest(G11vectorSPnormal, GJRvectorSPskewt))
```

```
#
##row two
#dm = np.append(dm,dmtest(G11vectorSPstudent, G11vectorSPskewt))
#dm = np.append(dm,dmtest(G11vectorSPstudent, GEvectorSPnormal))
#dm = np.append(dm,dmtest(G11vectorSPstudent, GEvectorSPstudent))
#dm = np.append(dm,dmtest(G11vectorSPstudent, GEvectorSPskewt))
#dm = np.append(dm,dmtest(G11vectorSPstudent, GJRvectorSPnormal))
#dm = np.append(dm,dmtest(G11vectorSPstudent, GJRvectorSPstudent))
#dm = np.append(dm,dmtest(G11vectorSPstudent, GJRvectorSPskewt))
##
###row three
#dm = np.append(dm,dmtest(G11vectorSPskewt, GEvectorSPnormal))
#dm = np.append(dm,dmtest(G11vectorSPskewt, GEvectorSPstudent))
#dm = np.append(dm,dmtest(G11vectorSPskewt, GEvectorSPskewt))
#dm = np.append(dm,dmtest(G11vectorSPskewt, GJRvectorSPnormal))
#dm = np.append(dm,dmtest(G11vectorSPskewt, GJRvectorSPstudent))
#dm = np.append(dm,dmtest(G11vectorSPskewt, GJRvectorSPskewt))
##
###row four
#dm = np.append(dm,dmtest(GEvectorSPnormal, GEvectorSPstudent))
#dm = np.append(dm,dmtest(GEvectorSPnormal, GEvectorSPskewt))
#dm = np.append(dm,dmtest(GEvectorSPnormal, GJRvectorSPnormal))
#dm = np.append(dm,dmtest(GEvectorSPnormal, GJRvectorSPstudent))
#dm = np.append(dm,dmtest(GEvectorSPnormal, GJRvectorSPskewt))
##
###row five
#dm = np.append(dm,dmtest(GEvectorSPstudent, GEvectorSPskewt))
#dm = np.append(dm,dmtest(GEvectorSPstudent, GJRvectorSPnormal))
#dm = np.append(dm,dmtest(GEvectorSPstudent, GJRvectorSPstudent))
#dm = np.append(dm,dmtest(GEvectorSPstudent, GJRvectorSPskewt))
##
###row six
#dm = np.append(dm,dmtest(GEvectorSPskewt, GJRvectorSPnormal))
#dm = np.append(dm,dmtest(GEvectorSPskewt, GJRvectorSPstudent))
#dm = np.append(dm,dmtest(GEvectorSPskewt, GJRvectorSPskewt))
##
###row seven
#dm = np.append(dm,dmtest(GJRvectorSPnormal, GJRvectorSPstudent))
#dm = np.append(dm,dmtest(GJRvectorSPnormal, GJRvectorSPskewt))
##
###row eight
#dm = np.append(dm,dmtest(GJRvectorSPstudent, GJRvectorSPskewt))
##
#
#yeet = len(dm)
#
#yeet2 = 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1


#-------------------------------------------------------------------------------------------------------------------------
#logl dm testing for bayes
dm = []
dm = np.append(dm,dmtest(Baysian_vector['h.1'], G11vectorSPnormal))
dm = np.append(dm,dmtest(Baysian_vector['h.1'], G11vectorSPstudent))
dm = np.append(dm,dmtest(Baysian_vector['h.1'], G11vectorSPskewt))
dm = np.append(dm,dmtest(Baysian_vector['h.1'], GEvectorSPnormal))
dm = np.append(dm,dmtest(Baysian_vector['h.1'], GEvectorSPstudent))
dm = np.append(dm,dmtest(Baysian_vector['h.1'], GEvectorSPskewt))
dm = np.append(dm,dmtest(Baysian_vector['h.1'], GJRvectorSPnormal))
dm = np.append(dm,dmtest(Baysian_vector['h.1'], GJRvectorSPstudent))
```

```
dm = np.append(dm,dmtest(Baysian_vector['h.1'], GJRvectorSPskewt))

# Python file to calculate MAE's

import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
import datetime as dt
import pandas_datareader.data as web
import scipy.stats
import scipy

# Load in the data.

# S&P500.
vf1 = pd.read_csv('vf_sp2.csv', index_col = 0)
start1 = dt.datetime(2010,1,5)
end1 = dt.datetime(2020,1,3)
sp500 = web.get_data_yahoo('^GSPC', start=start1, end=end1)
y1 = np.diff(np.log(sp500['Adj Close']))

# Euro Stoxx 50.
vf2 = pd.read_csv('vf_eu2.csv', index_col = 0)
start2 = dt.datetime(2010,1,6)
end2 = dt.datetime(2020,1,3)
stoxx50 = pd.read_csv('^STOXX50E.csv', index_col = 0)
y2 = np.diff(np.log(stoxx50['Adj Close']))


# MAE calculation
MAEdf = pd.DataFrame(columns=['MAE S&P500','MAE Euro Stoxx 50'])
for i in vf1.columns:
    MAE1 = 0
    MAE1 = np.subtract(vf1[i],np.power(y1,2))
    MAE1 = abs(MAE1)
    MAE1 = np.mean(MAE1)

    MAE2 = 0
    MAE2 = np.subtract(vf2[i],np.power(y2,2))
    MAE2 = abs(MAE2)
    MAE2 = np.mean(MAE2)

    MAEdf.loc[i] = [MAE1, MAE2]
    pd.DataFrame.to_latex(MAEdf, float_format="{:0.4e}".format)


# DM-statistic calculator
def DMtestT(vfmodel1, vfmodel2, y):
    y2 = y**2
    T = len(y)

    absErrorMdl1 = np.abs(vfmodel1 - y2[1500:T]);
    absErrorMdl2 = np.abs(vfmodel2 - y2[1500:T]);

    n = len(absErrorMdl1);
    dj = absErrorMdl1 - absErrorMdl2;

    s2d = 1/(n-1)*sum(np.power((dj - np.mean(dj)),2));
    Tcalc = np.sqrt(n)*((np.mean(dj)-0)/np.sqrt(s2d));
```

```
    Pvalue = scipy.stats.t.pdf(Tcalc,n-1);

    return Tcalc

def DMtestP(vfmodel1, vfmodel2, y):
    y2 = y**2
    T = len(y)

    absErrorMdl1 = np.abs(vfmodel1 - y2[1500:T]);
    absErrorMdl2 = np.abs(vfmodel2 - y2[1500:T]);

    n = len(absErrorMdl1);
    dj = absErrorMdl1 - absErrorMdl2;

    s2d = 1/(n-1)*sum(np.power((dj - np.mean(dj)),2));
    Tcalc = np.sqrt(n)*((np.mean(dj)-0)/np.sqrt(s2d));
    Pvalue = scipy.stats.t.pdf(Tcalc,n-1);

    return Pvalue

DM = list([0,0,0,0,0,0,0,0,0])

def DMappendT(mdl):
    for i in range(9):
        DM[i] = DMtestT(vf2[vf2.columns[mdl]][1500:len(vf2)], vf2[vf2.columns[i]][1500:len(vf2)], y2)

    return DM

def DMappendP(mdl):
    for i in range(9):
        DM[i] = DMtestP(vf2[vf2.columns[mdl]][1500:len(vf2)], vf2[vf2.columns[i]][1500:len(vf2)], y2)

    return DM

DMdf = pd.DataFrame(columns=range(18))


for i in range(9):
    DMdf[i*2] = DMappendT(i)

for i in range(18):
    if i%2==0:
        DMdf[i+1] = DMappendP(i/2)

DMdftrans = pd.DataFrame.transpose(DMdf)
DMdftrans = DMdftrans.round(decimals=4)
pd.DataFrame.to_latex(DMdftrans)

    # Matlab file S&P500

%% Import data.
clear
filename = 'sp500.csv';
dataStruct = importdata(filename,',',0);

AdjClose = dataStruct.data(:,5);

Date = dataStruct.textdata(:,1);
Date = datetime(Date);
y = diff(log(AdjClose));
```

```
%% Returns analysis
AdjClose = dataStruct.data(:,5);

Date = dataStruct.textdata(:,1);
Date = datetime(Date);

R = price2ret(AdjClose);
y = diff(log(AdjClose));

T = length(y);

% Calculate summary statistics
mean = mean(y, 'omitnan');
min = min(y);
max = max(y);
std = std(y, 'omitnan');
skewness = skewness(y);
kurtosis = kurtosis(y);

summaryStatistics = table(mean,min,max,std,skewness,kurtosis);
clear mean min max std skewness kurtosis

% Plot the returns
f1 = figure;

subplot(1,2,1);
plot(Date(2:end),y,'Color',[0,0,0]);
title('Log returns plot');
xlabel('Date');
ylabel('Log return');

subplot(1,2,2);
histogram(y);
title('Distribtion of returns');
xlabel('Log return');
ylabel('Probability');

savefig f1

% Compare the distibution of the returns to the Normal
f2 = figure;

subplot(1,2,1);
ksdensity(y); hold on
mu = summaryStatistics{1,1};
sigma = summaryStatistics{1,4};
size = length(y);
normDist = normrnd(mu,sigma,size,1);
title('Kernel density plot');
ksdensity(normDist); hold off

subplot(1,2,2);
qqplot(y,normDist);

savefig f2



%% Squared returns analysis
```

```
y2 = y.^2;

autocorr(y2,50);


%% Forecast Variances

% Use the function forecastVar in order to get the forcasted conditional
% variances.
%
% Input parameters:
% y = log returns timeseries.
% model: 1 = GARCH(1,1), 2 = GJR-GARCH(1,1), 3 = E-GARCH(1,1).
% distribution: 'Gaussian' or 't'.
varForecastsPython = csvread('vf_sp2.csv',1,1);

% GARCH(1,1)
varForecastG11N = varForecastsPython(1501:end,1);
varForecastG11T = varForecastsPython(1501:end,2);
varForecastG11ST = varForecastsPython(1501:end,3);

% EGARCH(1,1)
varForecastEG11N = varForecastsPython(1501:end,4);
varForecastEG11T = varForecastsPython(1501:end,5);
varForecastEG11ST = varForecastsPython(1501:end,6);

% GJRGARCH(1,1)
varForecastGJRG11N = varForecastsPython(1501:end,7);
varForecastGJRG11T = varForecastsPython(1501:end,8);
varForecastGJRG11ST = varForecastsPython(1501:end,9);

varForecastVECTOR(:,1) = varForecastG11N(1:end,1);
varForecastVECTOR(:,2) = varForecastG11T(1:end,1);
varForecastVECTOR(:,3) = varForecastG11ST(1:end,1);
varForecastVECTOR(:,4) = varForecastEG11N(1:end,1);
varForecastVECTOR(:,5) = varForecastEG11T(1:end,1);
varForecastVECTOR(:,6) = varForecastEG11ST(1:end,1);
varForecastVECTOR(:,7) = varForecastGJRG11N(1:end,1);
varForecastVECTOR(:,8) = varForecastGJRG11T(1:end,1);
varForecastVECTOR(:,9) = varForecastGJRG11ST(1:end,1);
%% Plot all forecasted volatility
plot(Date(1502:end), sqrt(varForecastG11N)); hold on
plot(Date(1502:end), sqrt(varForecastG11T));
plot(Date(1502:end), sqrt(varForecastG11ST));

plot(Date(1502:end), sqrt(varForecastEG11N));
plot(Date(1502:end), sqrt(varForecastEG11T));
plot(Date(1502:end), sqrt(varForecastEG11ST));

plot(Date(1502:end), sqrt(varForecastGJRG11N));
plot(Date(1502:end), sqrt(varForecastGJRG11T));
plot(Date(1502:end), sqrt(varForecastGJRG11ST));

plot(Date(1502:end), y(1501:T)); hold off
%% Plot all forecasted variances
plot(varForecastG11N); hold on
plot(varForecastG11T);

plot(varForecastGJRG11N);
plot(varForecastGJRG11T);
```

```
plot(varForecastEG11N);
plot(varForecastEG11T);

plot(y2(1501:T).*100^2); hold off

%% Out of sample measures

% MAE
MAEG11N = mean(abs(varForecastG11N - y2(1501:T)));
MAEG11T = mean(abs(varForecastG11T - y2(1501:T)));
MAEG11ST = mean(abs(varForecastG11ST - y2(1501:T)));

MAEEG11N = mean(abs(varForecastEG11N - y2(1501:T)));
MAEEG11T = mean(abs(varForecastEG11T - y2(1501:T)));
MAEEG11ST = mean(abs(varForecastEG11ST - y2(1501:T)));

MAEGJRG11N = mean(abs(varForecastGJRG11N - y2(1501:T)));
MAEGJRG11T = mean(abs(varForecastGJRG11T - y2(1501:T)));
MAEGJRG11ST = mean(abs(varForecastGJRG11ST - y2(1501:T)));

MAEVECTOR = ([MAEG11N, MAEG11T, MAEG11ST, MAEEG11N, MAEEG11T, MAEEG11ST, MAEGJRG11N, MAEGJRG11T, MAEGJRG11ST])
% MSE
%MSEG11N = mean(varForecastG11N - y2(1501:T).^2);
%MSEG11T = mean(varForecastG11T - y2(1501:T).^2);

%MSEGJRG11N = mean(varForecastGJRG11N - y2(1501:T).^2);
%MSEGJRG11T = mean(varForecastGJRG11T - y2(1501:T).^2);

%MSEEG11N = mean(varForecastEG11N - y2(1501:T).^2);
%MSEEG11T = mean(varForecastEG11T - y2(1501:T).^2);

%% Diebold-Mariano MAE Tests

% 1. G(1,1)N and G(1,1)T
[Tcalc,Pvalue] = DMTest(varForecastG11N, varForecastG11T,y)

% 2. GJR-G(1,1)N and GJR-G(1,1)T
[Tcalc,Pvalue] = DMTest(varForecastGJRG11N, varForecastGJRG11T,y)

% 3. E-G(1,1)N and E-G(1,1)T
[Tcalc,Pvalue] = DMTest(varForecastEG11N, varForecastEG11T,y)

% 4. G(1,1)N and GJR-G(1,1)N
[Tcalc,Pvalue] = DMTest(varForecastG11N, varForecastGJRG11N,y)

% 5. G(1,1)N and E-G(1,1)N
[Tcalc,Pvalue] = DMTest(varForecastG11N, varForecastEG11N,y)

% 6. GJR-G(1,1)N and E-G(1,1)N
[Tcalc,Pvalue] = DMTest(varForecastGJRG11N, varForecastEG11N,y)

% 7. G(1,1)T and GJR-G(1,1)T
[Tcalc,Pvalue] = DMTest(varForecastG11T, varForecastGJRG11T,y)

% 7. G(1,1)T and E-G(1,1)T
[Tcalc,Pvalue] = DMTest(varForecastG11T, varForecastEG11T,y)

% 8. GJR-G(1,1)T and E-G(1,1)T
[Tcalc,Pvalue] = DMTest(varForecastGJRG11T, varForecastEG11T,y)
```

```
clear DMVECTOR
for i=1:18
    % if i=even
    if mod(i,2) == 0
        disp even
        [~,DMVECTOR(1,i)] = DMTest(varForecastVECTOR(1:end,1),varForecastVECTOR(1:end,(i/2)),y);
    else
        disp odd
        DMVECTOR(1,i) = 38;
    end
end
DMVECTOR = num2str(DMVECTOR);


for i=1:length(DMVECTOR)
    disp(DMVECTOR(i))
    if(DMVECTOR(i) == '3' && DMVECTOR(i+1) == '8' && DMVECTOR(i+2) == ' ')
        DMVECTOR(i) = '&'
        DMVECTOR(i+1) = ' '
    end
    if(DMVECTOR(i) ~= ' ' && DMVECTOR(i+1) ~= ' ' && DMVECTOR(i+2) ~= ' ' && DMVECTOR(i+3) ~= ' ')
        if(DMVECTOR(i) == '-')
            DMVECTOR(i+9) = ' ';
            DMVECTOR(i+8) = ' ';
            DMVECTOR(i+7) = ' ';
        else
            DMVECTOR(i+8) = ' ';
            DMVECTOR(i+7) = ' ';
            DMVECTOR(i+6) = ' ';
        end
    end
end

%% Residual analysis

% GARCH(1,1)
mdlG11 = arima(0,0,0);
mdlG11.Variance = garch(1,1);
estMdlG11 = estimate(mdlG11, y(1:1500));
estMdlG11.Constant = 0;
[resG11, varG11, loglG11] = infer(estMdlG11, y(1:1500));

stdResG11 = resG11 ./ sqrt(varG11);
randN = randn(1500,1);

GARCHRESF1 = plot(stdResG11);
title('Standardized residuals GARCH(1,1) model');
xlabel('In sample window');
ylabel('Standardized residuals');
savefig GARCHRESF1;

GARCHRESF2 = figure;
subplot(1,2,1);
autocorr(stdResG11,50);
title('Autocorrelation of standardized residuals GARCH(1,1) model');
subplot(1,2,2);
autocorr(stdResG11.^2,50);
title('Autocorrelation of squared standardized residuals GARCH(1,1) model');
savefig GARCHRESF2;
```

```
% LB-test for autocorrelation
[h,pValue,stat,cValue] = lbqtest(stdResG11, 'Lags', 50)
[h,pValue,stat,cValue] = lbqtest(stdResG11.^2, 'Lags', 50)

% JB-test for normality
[h,pValue,stat,cValue] = jbtest(stdResG11)
[h,pValue,stat,cValue] = jbtest(stdResG11.^2)

% Ksdensity
%ksdensity(stdResG11); hold on
%ksdensity(randn(10000,1)); hold off

%% Summary statistics in and out of sample

obs1 = length(y(1501:end));
mean1 = mean(y(1501:end), 'omitnan');
median1 = median(y(1501:end), 'omitnan');
std1 = std(y(1501:end), 'omitnan');
min1 = min(y(1501:end));
max1 = max(y(1501:end));
skewness1 = skewness(y(1501:end));
kurtosis1 = kurtosis(y(1501:end));
[H,P,JBSTAT] = jbtest(y(1501:end));

    # Matlab file Stoxx50

%% Import data.
clear
filename = 'stoxx50.csv';
dataStruct = importdata(filename,',',0);


%% Returns analysis
AdjClose = dataStruct.data(:,5);

Date = dataStruct.textdata(:,1);
Date = datetime(Date);

R = price2ret(AdjClose);
y = diff(log(AdjClose));

T = length(y);

% Calculate summary statistics
mean = mean(y, 'omitnan');
min = min(y);
max = max(y);
std = std(y, 'omitnan');
skewness = skewness(y);
kurtosis = kurtosis(y);

summaryStatistics = table(mean,min,max,std,skewness,kurtosis);
clear mean min max std skewness kurtosis

% Plot the returns
f1 = figure;

subplot(1,2,1);
plot(Date(2:end),y,'Color',[0,0,0]);
```

```
title('Log returns plot');
xlabel('Date');
ylabel('Log return');

subplot(1,2,2);
histogram(y);
title('Distribtion of returns');
xlabel('Log return');
ylabel('Probability');

savefig f1

% Compare the distibution of the returns to the Normal
f2 = figure;

subplot(1,2,1);
ksdensity(y); hold on
mu = summaryStatistics{1,1};
sigma = summaryStatistics{1,4};
size = length(y);
normDist = normrnd(mu,sigma,size,1);
title('Kernel density plot');
ksdensity(normDist); hold off

subplot(1,2,2);
qqplot(y,normDist);

savefig f2


%% Squared returns analysis
y2 = y.^2;

autocorr(y2,50);


%% Forecast Variances

% Use the function forecastVar in order to get the forcasted conditional
% variances.
%
% Input parameters:
% y = log returns timeseries.
% model: 1 = GARCH(1,1), 2 = GJR-GARCH(1,1), 3 = E-GARCH(1,1).
% distribution: 'Gaussian' or 't'.
varForecastsPython = csvread('vf_eu2.csv',1,1);

% GARCH(1,1)
varForecastG11N = varForecastsPython(1501:end,1);
varForecastG11T = varForecastsPython(1501:end,2);
varForecastG11ST = varForecastsPython(1501:end,3);

% EGARCH(1,1)
varForecastEG11N = varForecastsPython(1501:end,4);
varForecastEG11T = varForecastsPython(1501:end,5);
varForecastEG11ST = varForecastsPython(1501:end,6);

% GJRGARCH(1,1)
varForecastGJRG11N = varForecastsPython(1501:end,7);
varForecastGJRG11T = varForecastsPython(1501:end,8);
```

```
varForecastGJRG11ST = varForecastsPython(1501:end,9);

varForecastVECTOR(:,1) = varForecastG11N(1:end,1);
varForecastVECTOR(:,2) = varForecastG11T(1:end,1);
varForecastVECTOR(:,3) = varForecastG11ST(1:end,1);
varForecastVECTOR(:,4) = varForecastEG11N(1:end,1);
varForecastVECTOR(:,5) = varForecastEG11T(1:end,1);
varForecastVECTOR(:,6) = varForecastEG11ST(1:end,1);
varForecastVECTOR(:,7) = varForecastGJRG11N(1:end,1);
varForecastVECTOR(:,8) = varForecastGJRG11T(1:end,1);
varForecastVECTOR(:,9) = varForecastGJRG11ST(1:end,1);

%% Plot all forecasted volatility
plot(Date(1502:end), sqrt(varForecastG11N)); hold on
plot(Date(1502:end), sqrt(varForecastG11T));
plot(Date(1502:end), sqrt(varForecastG11ST));

plot(Date(1502:end), sqrt(varForecastEG11N));
plot(Date(1502:end), sqrt(varForecastEG11T));
plot(Date(1502:end), sqrt(varForecastEG11ST));

plot(Date(1502:end), sqrt(varForecastGJRG11N));
plot(Date(1502:end), sqrt(varForecastGJRG11T));
plot(Date(1502:end), sqrt(varForecastGJRG11ST));

plot(Date(1502:end), y(1501:T)); hold off

%% Plot all forecasted volatility
plot(Date(1502:end), varForecastG11N); hold on
plot(Date(1502:end), varForecastG11T);
plot(Date(1502:end), varForecastG11ST);

plot(Date(1502:end), varForecastEG11N);
plot(Date(1502:end), varForecastEG11T);
plot(Date(1502:end), varForecastEG11ST);

plot(Date(1502:end), varForecastGJRG11N);
plot(Date(1502:end), varForecastGJRG11T);
plot(Date(1502:end), varForecastGJRG11ST);

plot(Date(1502:end), y2(1501:T)); hold off

%% Plot all forecasted variances
plot(sqrt(varForecastG11N)); hold on
plot(sqrt(varForecastG11T));
plot(sqrt(varForecastG11ST));

plot(sqrt(varForecastGJRG11N));
plot(sqrt(varForecastGJRG11T));
plot(sqrt(varForecastGJRG11ST));

plot(sqrt(varForecastEG11N));
plot(sqrt(varForecastEG11T));
plot(sqrt(varForecastEG11ST));

plot(diff(AdjClose)); hold off
%% Out of sample measures

% MAE
MAEG11N = mean(abs(varForecastG11N - y2(1501:T)));
```

```
MAEG11T = mean(abs(varForecastG11T - y2(1501:T)));
MAEG11ST = mean(abs(varForecastG11ST - y2(1501:T)));

MAEEG11N = mean(abs(varForecastEG11N - y2(1501:T)));
MAEEG11T = mean(abs(varForecastEG11T - y2(1501:T)));
MAEEG11ST = mean(abs(varForecastEG11ST - y2(1501:T)));

MAEGJRG11N = mean(abs(varForecastGJRG11N - y2(1501:T)));
MAEGJRG11T = mean(abs(varForecastGJRG11T - y2(1501:T)));
MAEGJRG11ST = mean(abs(varForecastGJRG11ST - y2(1501:T)));

MAEVECTOR = ([MAEG11N, MAEG11T, MAEG11ST, MAEEG11N, MAEEG11T, MAEEG11ST, MAEGJRG11N, MAEGJRG11T, MAEGJRG11ST])
% MSE
%MSEG11N = mean(varForecastG11N - y2(1501:T).^2);
%MSEG11T = mean(varForecastG11T - y2(1501:T).^2);

%MSEGJRG11N = mean(varForecastGJRG11N - y2(1501:T).^2);
%MSEGJRG11T = mean(varForecastGJRG11T - y2(1501:T).^2);

%MSEEG11N = mean(varForecastEG11N - y2(1501:T).^2);
%MSEEG11T = mean(varForecastEG11T - y2(1501:T).^2);

%% Diebold-Mariano MAE Tests

% 1. G(1,1)N and G(1,1)T
[Tcalc,Pvalue] = DMTest(varForecastG11N, varForecastG11T,y)

% 2. GJR-G(1,1)N and GJR-G(1,1)T
[Tcalc,Pvalue] = DMTest(varForecastGJRG11N, varForecastGJRG11T,y)

% 3. E-G(1,1)N and E-G(1,1)T
[Tcalc,Pvalue] = DMTest(varForecastEG11N, varForecastEG11T,y)

% 4. G(1,1)N and GJR-G(1,1)N
[Tcalc,Pvalue] = DMTest(varForecastG11N, varForecastGJRG11N,y)

% 5. G(1,1)N and E-G(1,1)N
[Tcalc,Pvalue] = DMTest(varForecastG11N, varForecastEG11N,y)

% 6. GJR-G(1,1)N and E-G(1,1)N
[Tcalc,Pvalue] = DMTest(varForecastGJRG11N, varForecastEG11N,y)

% 7. G(1,1)T and GJR-G(1,1)T
[Tcalc,Pvalue] = DMTest(varForecastG11T, varForecastGJRG11T,y)

% 7. G(1,1)T and E-G(1,1)T
[Tcalc,Pvalue] = DMTest(varForecastG11T, varForecastEG11T,y)

% 8. GJR-G(1,1)T and E-G(1,1)T
[Tcalc,Pvalue] = DMTest(varForecastGJRG11T, varForecastEG11T,y)


clear DMVECTOR
for i=1:18
    % if i=even
    if(mod(i,2) == 0)
        disp even
        DMVECTOR(1,i) = DMTest(varForecastVECTOR(1:end,1),varForecastVECTOR(1:end,(i/2)),y);
    else
        disp odd
```

```
        DMVECTOR(1,i) = 38;
    end
end
DMVECTOR = num2str(DMVECTOR);


for i=1:length(DMVECTOR)
    disp(DMVECTOR(i))
    if(DMVECTOR(i) == '3' && DMVECTOR(i+1) == '8' && DMVECTOR(i+2) == ' ')
        DMVECTOR(i) = '&'
        DMVECTOR(i+1) = ' '
    end
    if(DMVECTOR(i) ~= ' ' && DMVECTOR(i+1) ~= ' ' && DMVECTOR(i+2) ~= ' ' && DMVECTOR(i+3) ~= ' ')
        if(DMVECTOR(i) == '-')
            DMVECTOR(i+9) = ' ';
            DMVECTOR(i+8) = ' ';
            DMVECTOR(i+7) = ' ';
        else
            DMVECTOR(i+8) = ' ';
            DMVECTOR(i+7) = ' ';
            DMVECTOR(i+6) = ' ';
        end
    end
end

%% Residual analysis

% GARCH(1,1)
mdlG11 = arima(0,0,0);
mdlG11.Variance = garch(1,1);
estMdlG11 = estimate(mdlG11, y(1:1500));
[resG11, varG11, loglG11] = infer(estMdlG11, y(1:1500));

stdResG11 = resG11 ./ sqrt(varG11);

GARCHRESF1 = figure;
plot(stdResG11);
title('Standardized residuals AR(0)-GARCH(1,1) model');
xlabel('Time');
ylabel('Standardized residuals');
savefig GARCHRESF1;

GARCHRESF2 = figure;
subplot(2,2,1);
autocorr(y,50);
title('Autocorrelation of log returns');
subplot(2,2,2);
autocorr(stdResG11,50);
title('Autocorrelation of standardized residuals');
subplot(2,2,3);
autocorr(y.^2,50);
title('Autocorrelation of squared log returns');
subplot(2,2,4);
autocorr(stdResG11.^2,50);
title('Autocorrelation of squared standardized residuals');
savefig GARCHRESF2;

    install.packages("bayesGARCH")
    require("bayesGARCH")
```

```
    y_sp500 = diff(log(GSPC$Adj.Close)) * 100
    y_sp500 = y_sp500[1:1500]

    MCMC_sp500 = bayesGARCH(y_sp500)
    smpl_sp500 = formSmpl(MCMC_sp500)
    summary(smpl_sp500)

    y_stoxx = diff(log(STOXX50E$Adj.Close)) * 100
    y_stoxx = y_stoxx[1:1500]

    MCMC_stoxx = bayesGARCH(y_stoxx)
    smpl_stoxx = formSmpl(MCMC_stoxx)
    summary(smpl_stoxx)
```

```
    import arch
import scipy as sc
import numpy as np
import pandas as pd
import datetime as dt
import matplotlib.pyplot as plt
import pandas_datareader.data as web

from arch import arch_model
from arch.univariate import arch_model

filename1 = '^STOXX50E.csv'
filename2 = '^GSPC.csv'

scaling = 100

df1 = pd.read_csv(filename1, index_col = 0) #stoxx
df2 = pd.read_csv(filename2, index_col = 0) #sp500

adjClose1 = df1['Adj Close'] #stoxx
adjClose2 = df2['Adj Close'] #sp500

y1 = np.diff(np.log(adjClose1))*scaling  #stoxx
y2 = np.diff(np.log(adjClose2))*scaling  #sp500


def forecast_stoxx(y1):
    mdl1 = arch_model(y1, vol = 'GARCH', p=1, q=1, dist = 't', rescale = False)
    estMdl1 = mdl1.fix([0, 0.07714/100**2, 0.10801, 0.85755, 6.7707])
    forecast1 = estMdl1.forecast(horizon=1, start=1500)
    vf_stoxx = np.square(((np.sqrt(forecast1.variance))/scaling))
    data1 = pd.DataFrame({'vf_stoxx':[vf_stoxx]})
    return data1, vf_stoxx

data1, vf_stoxx = forecast_stoxx(y1)

def forecast_sp500(y2):
    mdl2 = arch_model(y2, vol = 'GARCH', p=1, q=1, dist = 't', rescale = False)
    estMdl2 = mdl2.fix([0, 0.0488/100**2, 0.1556, 0.8030, 7.45828])
    forecast2 = estMdl2.forecast(horizon=1, start=1500)
    vf_sp500 = np.square(((np.sqrt(forecast2.variance))/scaling))
    data2 = pd.DataFrame({'vf_sp500':[vf_sp500]})
    return data2, vf_sp500

data2, vf_sp500 = forecast_sp500(y2)
```

```
#MAE Bayesian Estimation
Scaling = 1
y1_u = np.diff(np.log(adjClose1))*Scaling  #stoxx
y2_u = np.diff(np.log(adjClose2))*Scaling #sp500

y1_u_sq = y1_u ** 2
y2_u_sq = y2_u ** 2

def mae_stoxx(vf_stoxx):
    mae_stoxx = np.subtract(vf_stoxx['h.1'][1500:2508], y1_u_sq[1500:2508])
    mae_stoxx = np.abs(mae_stoxx)
    mae_stoxx = np.mean(mae_stoxx)
    return mae_stoxx

mae_stoxx = mae_stoxx(vf_stoxx)

def mae_sp500(vf_stoxx):
    mae_sp500= np.subtract(vf_sp500['h.1'][1500:2516], y2_u_sq[1500:2516])
    mae_sp500 = np.abs(mae_sp500)
    mae_sp500 = np.mean(mae_sp500)
    return mae_sp500

mae_sp500= mae_sp500(vf_sp500)

#y1 = stoxx
#y2 = SP
```