Time Series Project

Edoardo Di Rienzo 1873844
Lucrezia Castrichella 2002835
Mattia Zitelli 1999380

May 1, 2022

Contents

1 Abstract

During this project we are taking six variables expressed as prices such as: WTI, Europe Brent, Propane, Kerosene, Heating Oil, Gasoline. Firstly we used some tests to analyse some statistics of the Data, like for example the stationarity of the process. Then we built some models, precisely, sGARCH models with all the distributions, and we applied the Backtesting procedure either at 95% and 99% with the VaR prediction and the DQ test. If the models that we had passed more than two of three tests, we applied the MCS procedure developed by Hansen et al. (2011), an MCS is a set of models that is constructed so that it will contain the best model with a given level of confidence, in our case we applied both the levels. In the other part of the project we applied the Garch Midas models, that are models, that mix different frequencies of dates. In our case we used the Covid deaths as a Midas variable to put in our models. We then created the matrices with all the values we needed: the returns of the variables that we mentioned before, considered as weekly, and the Covid deaths already taken as weekly. We then applied the Garch Midas procedure on all the variables and then we proceded as we did before with the VaR prediction and the DQ test for these kind of models. We saw which one passed two out of the three tests, and then we applied the MCS procedure on these models. In the end we tried to use the MCS procedure for both these models together to see the final models.

2 GARCH model

We employed the GARCH model, which is based on the ARCH models, ARCH is an acronym for Auto-Regressive Conditional Heteroskedasticity, and an ARCH model is a Self-Regressive Model with Conditional Heteroskedasticity. Heteroskedasticity is a property of time series of equity returns in which periods of high volatility are followed by periods of relatively low volatility, and in which some groups or subsets of random variables exhibit variance that differs from the rest. The following is the definition of the GARCH (p,q) model:

$$\sigma_{t^2} = \omega + \sum_{j=1}^{p} \alpha_j \epsilon^2_{(t-j)} + \sum_{j=1}^{q} \beta_j \sigma^2_{(t-j)} \tag{1}$$

The process is stationary when $\omega > 0$ and $\alpha + \beta < 1$. We must enforce that $\alpha \geq 0$ and $\beta \geq 0$ to ensure that the conditional variance is positive. In this paper we considered different distributions errors, the Normal (norm), T-Student (std), Skew Normal (snorm), Skew Student (sstd), GED. The rugarch package implements a rich set of univariate GARCH models, in particular we used the sGarch model (standard Garch), Bollerslev (1986) that may be written as:

$$\sigma^2_t = (\omega + \sum_{j=1}^{m} \zeta_j v_j t) + \sum_{j=1}^{q} \alpha_j \epsilon^2_{(t-j)} + \sum_{j=1}^{p} \beta_j \sigma^2_{(t-j)} \tag{2}$$

with σ^2_t denoting the conditional variance, ω the intercept and ϵ^2_t the residuals from the mean filtration process discussed previously. The GARCH order is defined by (q, p) (ARCH, GARCH), with possibly m external regressors v_j which are passed pre-lagged. The model specifications used are defined in the "ugarchspec".

```
> args(ugarchspec)
R > function (variance.model = list(model = "sGARCH", garchOrder = c(1,1),
submodel = NULL, external.regressors = NULL, variance.targeting = FALSE),
mean.model = list(armaOrder = c(1, 1), include.mean = TRUE, archm = FALSE,
archpow = 1, arfima = FALSE, external.regressors = NULL, archex = FALSE),
distribution.model = "norm", start.pars = list(), fixed.pars = list(), ...)
```

2.1 The outcome of the tests

Table 1: Summary statistics of daily returns. J-B is the Jarque-Bera test.L-B is the Ljung-Box test on squared returns with 20 lags. ARCH LM is the ARCH Lagrange Multiplier test. ADF is the Augmented Dickey-Fuller unit root test.

	Different statistics:							
	MEAN	SD	SKEWNESS	KURTOSIS	JB	LB	ARCH LM	ADF
WTI Returns	0.0014	0.0746	-2.2429	37.0293	15463*	83.102*	56.243*	-6.7417*
Europe Brent Returns	0.0010	0.0693	-3.8981	57.1677	39308*	42.736*	34.576*	-6.6761*
Heating Oil Returns	0.0008	0.0372	-1.6286	13.0169	1456.2*	105.9*	66.842*	-7.5595*
Propane Returns	0.0026	0.0402	-0.4836	8.6236	427.36*	155.42*	80.689*	-6.5653*
Gasoline Returns	0.0012	0.0494	-1.7243	17.2531	2822.5*	207.53*	161.98*	-5.3365*
Kerosene Returns	0.0008	0.0466	-2.0516	19.6395	3854.9*	55.599*	33.503*	-5.8326*

Initially we computed the Dickey Fuller Test for the normal variables, we saw that for each variable we rejected the null, that is regarding the non stationarity of the processes. For making it stationary we had to compute the returns by doing the difference of the logarithm of the variable that we had:

```
R > WTI2 <- diff(log(Data$WTI))
```

As soon as we did this we then computed all the other tests that we were interested in, such as:

1. Mean
2. Standard Deviation
3. Skewness :is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean.
4. Kurtosis: is a measure of the "tailedness" of the probability distribution of a real-valued random variable.
5. Jarque-Bera Test: is a goodness-of-fit test of whether sample data have the skewness and kurtosis matching a normal distribution.
6. Ljung Box: is a type of statistical test of whether any of a group of autocorrelations of a time series are different from zero.
7. ARCH Lagrange Multiplier Test: ARCH-LM test is the standard approach to detect autoregressive conditional heteroscedasticity

8. Augmented Dickey Fuller Test: tests the null hypothesis that a unit root is present in a time series sample. The alternative hypothesis is different depending on which version of the test is used, but is usually stationarity or trend-stationarity.

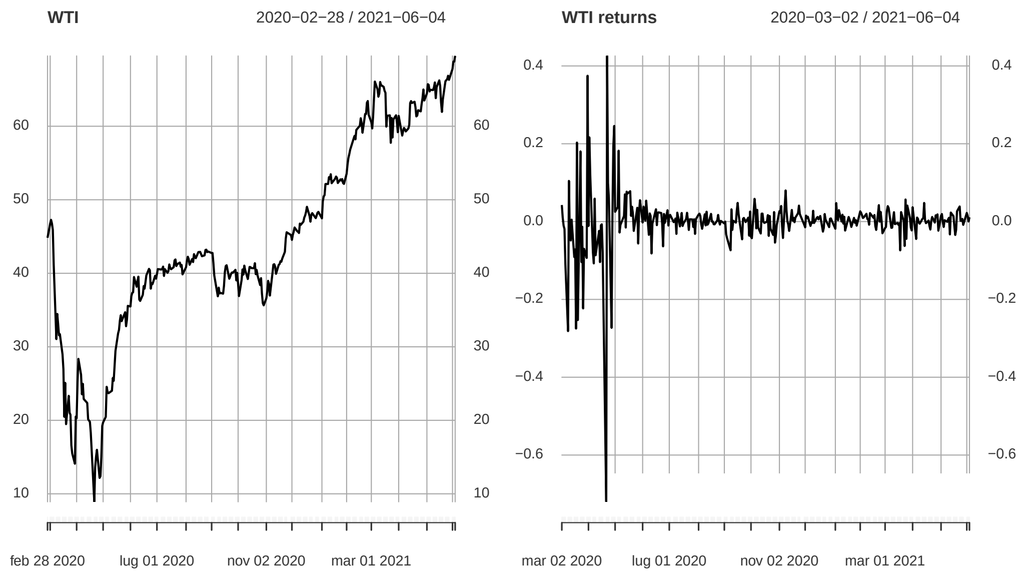

Figure 1: Comparison between prices of WTI

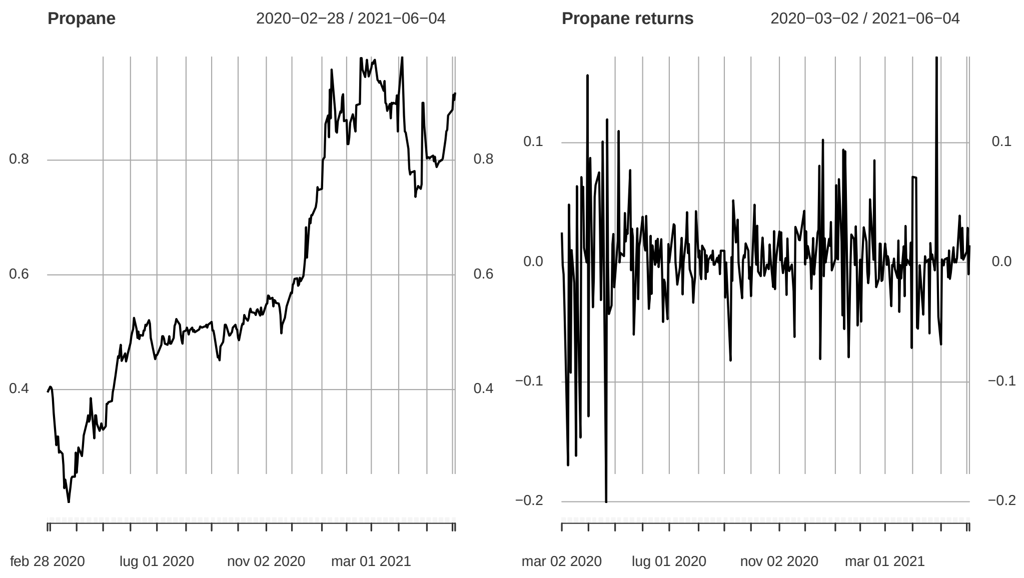

Figure 2: Comparison between prices of Propane

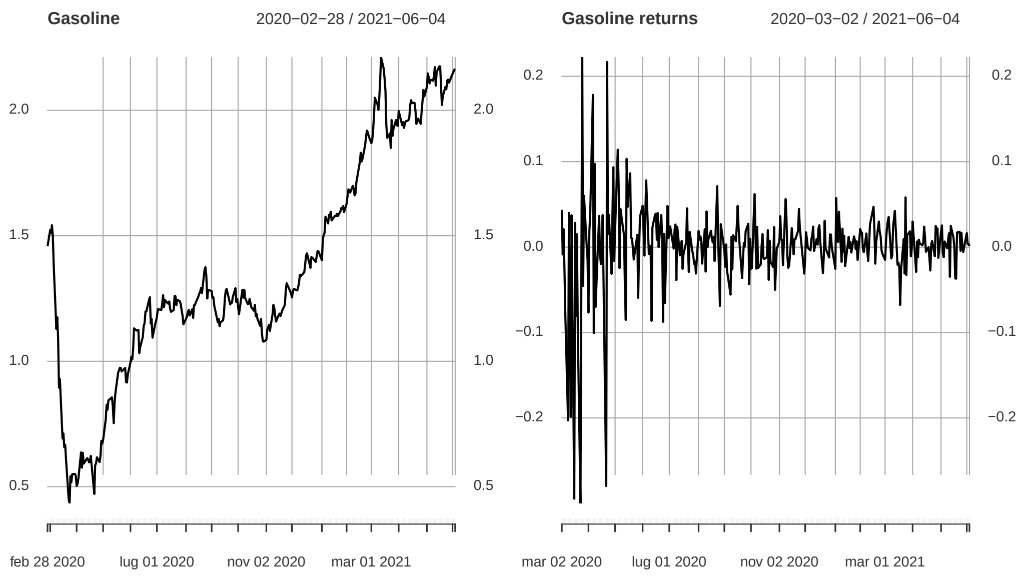

Figure 3: Comparison between prices of Gasoline

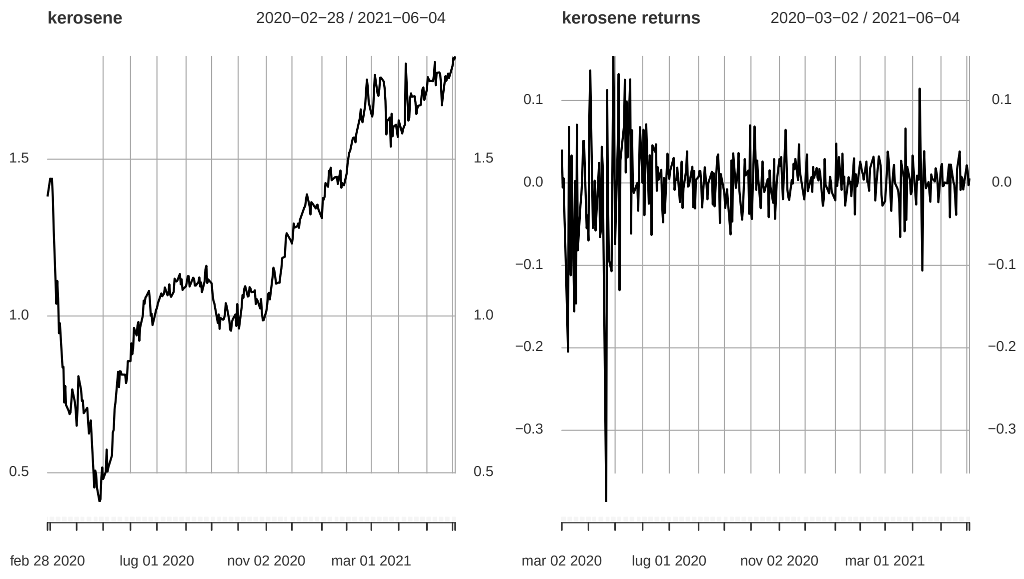

Figure 4: Comparison between prices of Kerosene

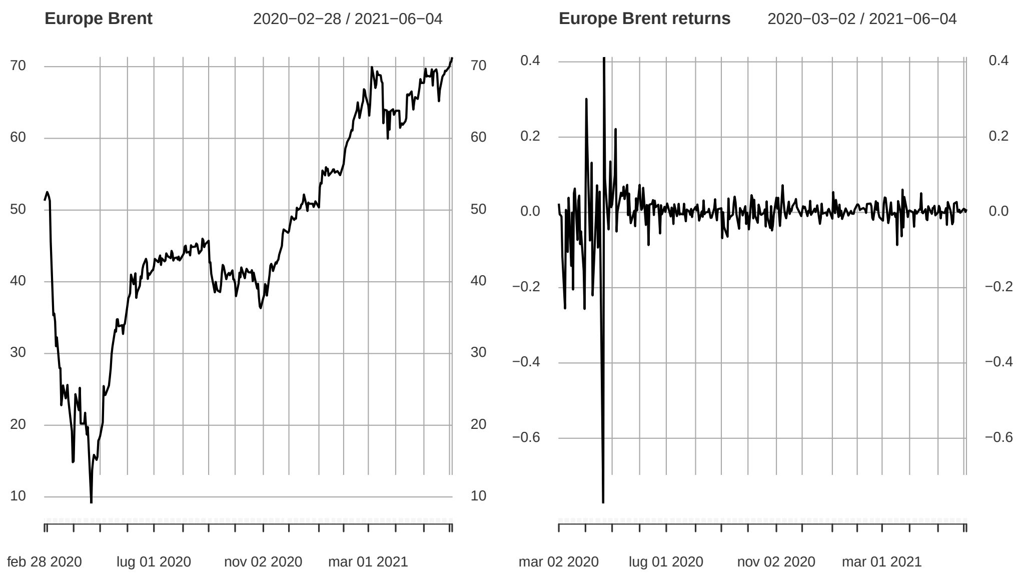

Figure 5: Comparison between prices of Europe Brent

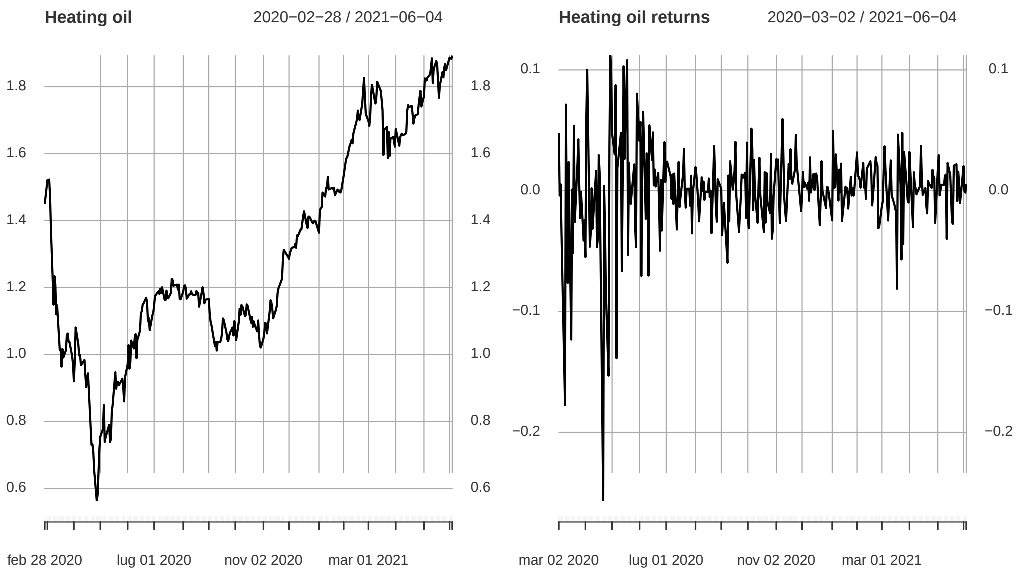

Figure 6: Comparison between prices of Heating oil

3 GARCH Midas models

Ghysels et al. (2006) present MIDAS (Mixed Data Sampling), a regression scheme that allows data from different frequencies to be used in the same model. This allows high-frequency return data to be combined with macroeconomic data that are only available at lower frequencies, such as monthly or quarterly. To evaluate time-varying market volatility, Engle et al. (2009) offer the GARCH-MIDAS model within the MIDAS framework. The conditional variance is separated into long-term and short-term components in this approach. The conditional variance is influenced by low frequency variables through the longterm component. In this paper, we apply the recently proposed methodology, GARCH-MIDAS, to examine a daily time series. Our investigation mainly focuses on the energy data prices. Using GARCH-MIDAS we decompose the return volatility to its short-term and long-term components. We examine a large group of variables which include WTI, Europe.Brent, Heating Oil, Propane, Gasoline, Kerosene and COVID deaths USA. In order to capture the information contained in different economic variables and investigate their combined effect, we perform a principal component analysis. The advantage of this approach is to reduce the number of parameters and increase the computational efficiency. The GARCH-MIDAS model can formally be described as below. Assume the return on day i in month t follows the following process:

$$r_{i,t} = \mu + \sqrt{\tau_t * g_{i,t} * \epsilon_{i,t}} \qquad \forall = i, ..., N_t \tag{3}$$

$$\epsilon_{i,t}|\phi_{i-1,t} \sim N(0,1) \tag{4}$$

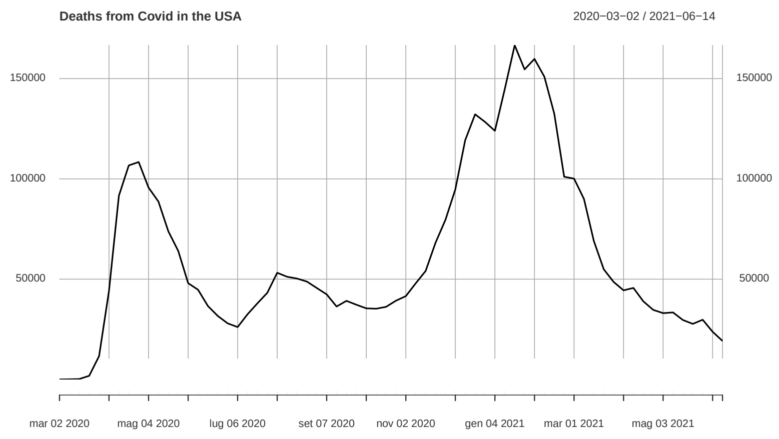

Figure 7: Deaths from covid in the USA from 20/03/2020 to 14/06/2021

4 VaR and Backtesting procedure for GARCH models

4.1 Value at Risk (VaR) function

The VaR (Value-at-risk) is a statistical measure of the riskiness of financial entities or portfolios of assets. For a return series, VaR is defined as the high quantile of the negative value of the returns. With a large enough data set, it may be used the empirical quantile calculated with 'quantile'. If a (right) assumption is made on the return distribution, such as the normal distribution, more efficient VaR estimations are achieved. Cornish-Fisher estimations of VaR may be more suitable if your return series is skewed and/or contains excess kurtosis. There are three key elements to describe the Value at Risk (VaR):

1. A time period
2. The dollar amount of VaR (portfolio, assets, etc.)
3. A given normal market condition (or confidence interval).

The var() function in R calculates a vector's sample variance. It is the measure of how much value is away from the mean.

4.2 The Backtesting Idea

Backtesting is a statistical procedure where actual profits and losses are systematically compared to corresponding VaR estimates. In theory, however, a good VaR model not only produces the 'correct' amount of exceptions but also exceptions that are evenly spread over time i.e. are independent of each other. In most cases, the backtesting process is determined by the type of forecasts provided. VaR predictions are also known as "probability range forecasts" or "interval forecasts," because they define an interval within which the forecast value is predicted to be found, with a probability p. Three separate backtests on the VaR of the commodities chosen were performed in this paper: the unconditional coverage test, the conditional coverage test, and the dynamic quantile test. VaR is thus a quantile of the loss distribution in probabilistic terms. Typical values for α are $\alpha = 0.95$, $\alpha = 0.99$. To backtest a model, we must create a sequence $(I_{t+1})_{t=1}^{T}$ (where T denotes the number of days in the testing period), which reveals when past exceedances occurred. A 1 with probability $1-\alpha$ and a 0 with probability alpha should be expected. Following this logic, we may conclude that a risk model for VaR calculations should look into two possible hypotheses:

1. the unconditional coverage hypothesis: $H_0 = E[I_{t+1}] = \pi = (1-\alpha)$
2. the conditional coverage hypothesis: $H_0 = E_t[I_{t+1}] = \pi_{t+1|t} = (1-\alpha)$

4.3 Unconditional Coverage Tests (Kupiec's Test)

In this case, it would be enough to check if the number of violations follows a binomial distribution: $f(x) = \binom{T}{x} p^x (1-p)^{T-x}$
Since the binomial distribution may be approximated by a normal distribution, we can apply a simple mean test to discuss the null hypothesis:

$$z = \frac{x - pT}{\sqrt{p(1-p)T}} \approx N(0,1) \tag{5}$$

where pT is the expected number of exceptions and p(1-p)T their variance.

We can also use likelihood ratio tests to perform unconditional coverage tests and he most widely popular likelihood ratio tests have been suggested by Kupiec.
The proportion of failure test's null hypothesis is:

$$H_0 : p = \hat{p} = \frac{x}{T} \tag{6}$$

The idea is to see if there is a significant difference between the observed failure rate, p, and the theoretical failure rate, p. Under the null hypothesis, that is under the assumption of a correct model spec- ification, LR_{uc} is asymptotically χ^2_1 (chi-squared distributed) with one degree of freedom.
The null hypothesis will be rejected and the model will be identified as erroneous if the value of the statistic is greater than the critical value of the chi-square distribution. It's important to note that when the sample size grows, the test's power grows as well. The POF-Test also has the issue of ignoring the time when losses occur. As a result, it might not be able to reject a model that generates clustered VaR violations. This is why there is an additional type of coverage test: conditional coverage testing.

4.4 Conditional Coverage Tests (Christoffersen's Test)

Before formulating a conditional coverage test we must first solve the independence problem: in an accurate model, an exception today should not be dependent on whether or not an exception occurred the day before. Examining the autocorrelation function is the simplest technique to test for dynamics in time series analysis. For the violation process, let γ_k be the autocorrelation at lag k. The degree of interconnection between an exceedance in one of the last m trading days and a hit today can be determined by plotting the autocorrelation function (for k=1, ..., m). Then the null hypothesis will be:

$$H_0 : \gamma_k = 0, for k = 1, ..., m \tag{7}$$

For the independence test, we can alternatively use a likelihood technique. Assume that the hit sequence's dependence structure can be represented as a first-order Markov chain with a transition probability matrix.

$$\Pi_1 = \begin{bmatrix} 1 - \pi_{01} & \pi_{01} \\ 1 - \pi_{11} & \pi_{11} \end{bmatrix} \tag{8}$$

We can interpret the numbers in the matrix as follow:

(A) π_{01} is the probability of having a violation tomorrow, conditional on today having no violations;

(B) π_{11} is the probability of tomorrow being a violation given today is also a violation;

(C) $1 - \pi_{01}$ is the probability of a non-violation following a non-violation;

(D) $1 - \pi_{11}$ is the probability of a non-violation succeeding a violation.

If we given a sample of T observations, the probability function of the first-order Markov process can be written as:

$$L(\Pi_1) = (1 - \pi_{01})^{T_{00}} \pi_{01}^{T_{01}} (1 - \pi_{11})^{T_{10}} \pi_{11}^{T_{11}} \tag{9}$$

where $T_{i,j}$, $i, j = 0, 1$ is the number of observations with a j succeeding an i.
Taking the first derivatives with regard to π_{01} and π_{11} and setting them equal to zero yields the Maximum Likelihood (ML) estimates. Finally, the purpose is to evaluate both aspects of a good VaR model at the same time: the correct failure rate and independence of exceptions. The unconditional coverage and independence qualities are included in the joint test, also known as the conditional coverage test. A significant aspect of this is that a rejection of the conditional. coverage may indicate that the VaR model has to be improved in order to minimize clustering behaviour.

$$LR_{cc} = LR_{uc} + LR_{ind} \tag{10}$$

and $LR_{cc} \sim \chi^2_2$
Christoffersen's approach allows to see if the reason for not passing the test was due to erroneous coverage, clustered violations, or a combination of the two.

4.5 DQ Tests (Dynamic Quantile)

Typical VaR tests cannot control for the dependence of violations, i.e., violations may cluster while the overall (unconditional) average of violations is not significantly different from $\alpha = 1 - VaR$. The conditional expectation should also be zero meaning that $Hit_t(\alpha)$ is uncorrelated with its own past and other lagged variables (such as r_t, r_t^2 or the one-step ahead forecast VaR). To test this assumption, the dynamic conditional quantile (DQ) test is used which involves the following statistic:

$$DQ = \frac{Hit^T X (X^T X)^{-1} X^T Hit}{\alpha(1 - \alpha)} \tag{11}$$

where X is the matrix of explanatory variables (e.g., raw and squared past returns) and Hit the vector collecting $Hit_t(\alpha)$. Under the null hypothesis, Engle and Manganelli (2004) show that the proposed statistic DQ follows a χ^2_q where $q = rank(X)$.
Berkowitz notes that running the DQ test with lagged VaR from a GARCH model and lagged violation, produces consistent results (as well as in terms of power) in a practical daily VaR examination.

4.6 Backtesting in R

To evaluate the backtesting in R, we applyied the function "VaRTest" to the rolls of forecasts produced by each distribution. In our script, to allow for a quicker and overall vision of the data, distributions were forecasted all together in a single roll. The "VaRTest" function uses the variable "alpha" to represent the confidence level of its reading. In our case, we ran the test at confidence level 95% and 99% , where 1-alpha equals the confidence level. The results show the values of all the previous backtesting: the Unconditional Coverage test, the Conditional Coverage test and the DQ test.

4.7 Backtesting results

At 95 percent confidence level we observed similar results regarding the unconditional and the conditional coverage tests. All commodities pass the tests in all distributions of errors.Regarding the DQ test, WTI, Europe Brent, Heating Oil and Gasoline showing promising results. Kerosene instead do not qualify for any of their distributions. The commodity Propane shows good results except for the Student-T and the Skew Student-T distributions. At 99 percent confidence level the overall results seem more promising. All the commodities except the variable Propane show excellent results for all tests, in particular it shows value higher than the critical ones for each test around the Skew-Normal distribution. Models that have passed at least two of three tests have been subjected to the mcs procedures.

Table 2: returns backtesting results. Critical values of the test : 95% Kupiec = 3.841, Christoffersen = 5.991; 99% Kupiec = 6.635, Christoffersen = 9.21

	95%:			*99%:*		
	Kupiec	Christoffersen	DQ	Kupiec	Christoffersen	DQ
WTI						
sGarch-Norm	0.429	0.554	0.5786	0.012	0.056	3.7905
sGarch-Skew Norm	0.429	0.554	0.3411	0.012	0.056	2.4371
sGarch-Student-T	0.135	0.327	0.1827	0.012	0.056	1.5629
sGarch-Ged	0.803	0.545	0.434	0.474	0.484	1.1661
sGarch-Skew Student-T	0.429	0.0.554	0.3000373	0.012	0.05	2.3688
Europe Brent						
sGarch-Norm	0.007	0.933	1.6220	0.012	0.056	3.4484
sGarch-Skew Norm	0.627	1.181	2.2229	0.012	0.056	3.621169
sGarch-Student-T	0.329	1.729	2.8377	0.474	0.484	1.2814
sGarch-Ged	0.062	1.212	2.2633	0.474	0.484	1.2230
sGarch-Skew Student-T	0.062	1.212	2.1103	0.474	0.484	1.2121
Heating Oil						
sGarch-Norm	0.186	1.111	5.9628	0.608	0.707	9.5609*
sGarch-Skew Norm	0.186	1.111	5.9628	0.608	0.707	8.6460
sGarch-Ged	0.007	0.605	5.6932	0.608	0.707	9.0329
sGarch-Skew Student-T	0.627	1.975	6.4790	0.608	0.707	7.7688
Gasoline						
sGarch-Norm	0.627	1.181	2.9743	0.608	0.707	9.3374
sGarch-Skew Norm	2.45	2.729	4.0529	0.608	0.707	7.0525
sGarch-Ged	0.627	1.181	2.6708	0.608	0.707	7.8862
sGarch-Skew Student-T	0.627	1.975	5.0529	0.012	0.056	2.8537
Kerosene						
sGarch-Norm	0.329	1.729	16.5510*	0.012	0.056	0.7862
sGarch-Skew Norm	0.062	1.212	17.5619*	0.012	0.056	1.0525
sGarch-Ged	0.062	1.212	15.3315*	0.012	0.056	1.6477
sGarch-Skew Student-T	0.007	0.933	16.1849*	0.012	0.056	1.676853
Propane						
sGarch-Norm	0.007	0.933	3.3840	3.697	3.976	20.2237*
sGarch-Skew Norm	0.007	0.933	6.3673	5.914*	6.318*	31.8703*
sGarch-Student-T	0.7900	2.466	17.7783*	0.608	0.707	9.19798
sGarch-Ged	0.007	0.933	4.5731	0.608	0.707	10.9170
sGarch-Skew Student-T	0.7900	2.466	20.1356*	3.697	3.976	37.5813*

**The null hypothesis is rejected at the 5% significance level.*

5 The Model Confidence Set

5.1 Introduction

In order to select the final models we used the Model Confidence Set method developed by Hansen (2011). The Model Confidence Set approach, from a frequentist point of view, is a tool for summarizing the relative performances of an entire set of models by establishing which models are statistically superior and at what level of significance. The Hansen process consists of a series of statistical tests that allow the construction of a set of "superior" models, known as "Superior Set Models" (SSM), in which the null hypothesis of equal predictive ability (EPA) is not rejected at a given level of confidence α. The EPA statistic tests are assessed for an arbitrary loss function, which essentially implies that depending on the loss function used, it is possible to test models on various aspects. The Model Confidence Set method begins with a set of m competing models M_0 and ends with a smaller set of superior models (i.e. SSM), indicated by $\hat{M}^*_{1-}$. The EPA hypothesis is evaluated at each phase of the iterative approach, and if the null hypothesis is accepted, the process ends and the SSM is generated; otherwise, the EPA hypothesis should be examined again after the exclusion of the worst model.
The MCS procedure is based on an:

1. Equivalence Test , δ_M
2. Elimination Rule , e_M

5.2 The MCS procedure

Formally, let Y_t the observation at time t and $\hat{Y}_{i,t}$ the output of model i at time t, the loss function $\ell_{i,t}$ associated to the $i-th$ model is defined as:

$$\ell_{i,t} = \ell(Y_t, \hat{Y}_{i,t}) \tag{12}$$

and measures the difference between the output $\hat{Y}_{i,t}$ and the a posteriori realisation Y_t.
The procedure starts with an initial set of models $\hat{M}^0$ of dimension m, which includes all of the model characteristics specified, and produces a smaller set, the superior set of models (SSM), $\hat{M}^*_{1-\alpha}$, of dimension $m^* \leq m$, with a given confidence level 1. The best scenario is when the final set consists of a single mode, $m^* = 1$. Let $d_{ij,t}$ denotes the loss differential between models i and j:

$$d_{ij,t} = \ell_{i,t} - \ell_{j,t}, i, j = 1, ..., m and t = 1, ..., n \tag{13}$$

The formula represents the model loss differential defined over the training period. For a given collection of models M, the EPA hypothesis may be expressed in two ways:

$$H_{0,M} : c_{ij} = 0, \forall i, j = 1, 2, ..., m \tag{14}$$

$$H_{A,M} : c_{ij} \neq 0, for some i, j = 1, ..., m, \tag{15}$$

or

$$H_{0,M} : c_{i}. = 0, \forall i = 1, 2, ..., m \tag{16}$$

$$H_{A,M} : c_{i}. \neq 0, for some i = 1, ..., m, \tag{17}$$

According to Hansen (2011), the two hypothesis defined can be tested by constructing the following two statistics:

$$t_{ij} = \frac{\bar{d_{ij}}}{\sqrt{\hat{var}(\bar{d_{ij}})}}, for i, j \in M \tag{18}$$

Where $\hat{d_{ij}}$ measures the relative sample loss between the i-th and j-th models,while the denominator is the bootstrapped estimation of $var(\hat{d_{ij}})$.

$$t_{i.} = \frac{\bar{d_{i,.}}}{\sqrt{v\hat{a}r(\bar{d_{i,.}})}}, for i \in M \tag{19}$$

Where $\hat{d_i}.$ measures the simple loss of the i-th model relative to the averages losses across models in the set M, while the denominator is the bootstrapped estimation of $var(\hat{d_i}.)$.
The MCS technique for obtaining the SSM, summarized, consists of the following steps:

1. set $M = M_0$
2. test for EPA–hypothesis: if EPA is accepted terminate the algorithm and set $M^*_{1-\alpha} = M$, otherwise use the elimination rules (e_M) to determine the worst model.
3. remove the worst model, and go to step 2.

The set $M^*_{1-\alpha} = M$ consists of the set of 'surviving' models (SSM).
The equivalence test and elimination rule must meet three assumptions asymptotically for each potential subset M M0. First, the equivalency test is size-limited, in the sense that it rejects no more than of all cases where the null hypothesis is true. Second, the equivalence tests reject with probability 1 in the alternative. Third, the elimination rule, in the alternative, never chooses a good model.

5.3 MCS procedure in R

MCS is a R package that aims to implement the previously described Model Confidence Set (MCS). The MCS procedure is used to compare different models under an user defined loss function. The loss function evaluates the competing models' "performance" at each time point t throughout the evaluating period. Assuming there are m competing models and an evaluating period of length n, a loss matrix of dimension (m x n) may be constructed using the given loss function. Loss, included in the MCS packages, can be freely chosen by the user. There are three different loss functions available in the MCS package, we used the "LossVaR()" that is used to check the performances associated to VaR (or more generally quantile) forecasts.

R > LossVaR(realized, evaluated, which = 'asymmetricLoss', type = 'normal', delta = 25, tau)

The MCSprocedure() function may then be used to create a collection of better models. The main inputs of the function are: loss, alpha, B, cluster, statistic. The construction of the Superior Set of Models can be done using the following portion of code:

R> library(MCS)
R> data(Loss)
R> SSM <- MCSprocedure(Loss = Loss, alpha = 0.2, B = 5000, statistic = "Tmax")

The outcome of the MCS technique for our time series is shown in the next section. In our application we only evaluate models that have passed at least two of the three tests.

Table 3: Estimated sGARCH models with MCS procedure at 95%

95%	Rank_M	v_M	MCS_M	Rank_R	v_R	MCS_R	Loss
WTI							
sGARCH-norm	1	− 2.246	1	1	− 1.151	1	0.003
sGARCH-std	3	1.175	0.418	4	1.965	0.157	0.003
sGARCH-ged	2	− 1.860	1	2	1.151	0.632	0.003
sGARCH-sstd	4	1.504	0.230	3	1.874	0.191	0.003
Europe Brent							
sGARCH-norm	1	− 1.115	1	1	− 0.657	1	0.003
sGARCH-std	4	0.470	0.911	3	0.932	0.801	0.003
sGARCH-ged	3	0.461	0.916	4	0.953	0.788	0.003
sGARCH-snorm	5	0.515	0.888	5	1.700	0.280	0.003
sGARCH-sstd	2	0.028	1	2	0.657	0.925	0.003
Gasoline							
sGARCH-norm	2	− 0.407	1	4	2.083	0.132	0.002
sGARCH-ged	1	− 2.048	1	1	− 1.573	1	0.002
sGARCH-snorm	4	0.978	0.565	3	1.794	0.241	0.002
sGARCH-sstd	3	0.943	0.590	2	1.573	0.351	0.002
Propane							
sGARCH-norm	2	− 0.729	1	5	2.428	0.063	0.003
sGARCH-std	3	0.334	0.897	2	0.993	0.763	0.004
sGARCH-ged	1	− 1.669	1	1	− 0.993	1	0.003
sGARCH-snorm	4	0.545	0.757	4	2.058	0.139	0.004
sGARCH-sstd	5	0.855	0.573	3	1.921	0.181	0.004
Kerosene							
sGARCH-norm	2	0.625	0.689	2	1.150	0.455	0.003
sGARCH-ged	1	− 1.763	1	1	− 1.150	1	0.003
sGARCH-sstd	3	1.275	0.268	3	2.445	0.038	0.003
Heating Oil							
sGARCH-ged	1	− 2.643	1	1	− 2.643	1	0.002

The models that passed two out of three tests of the VaR backtesting procedure at 95%, were then subjected to the MCS. One of the most interesting results of this table is the Heating Oil variable and the models related to it. They MCS procedure did cut all the models except from the ged one, that remained.

Table 4: Estimated sGARCH models with MCS procedure at 99%

99%	Rank_M	v_M	MCS_M	Rank_R	v_R	MCS_R	Loss
WTI							
sGARCH-norm	1	− 1.974	1	1	− 1.477	1	0.001
sGARCH-ged	2	− 0.465	1	3	6.412	0	0.001
sGARCH-snorm	3	1.224	0.257	2	1.477	0.189	0.001
Europe Brent							
sGARCH-norm	1	− 2.912	1	1	− 2.912	1	0.001
Gasoline							
sGARCH-norm	2	0.388	0.572	2	0.388	0.572	0.001
sGARCH-snorm	1	− 0.388	1	1	− 0.388	1	0.001
Propane							
sGARCH-norm	4	0.343	0.965	3	0.928	0.853	0.001
sGARCH-std	1	− 1.420	1	1	− 0.365	1	0.001
sGARCH-ged	2	− 1.285	1	2	0.365	0.996	0.001
sGARCH-snorm	5	1.183	0.434	4	1.423	0.538	0.001
sGARCH-sstd	3	0.275	0.985	5	1.622	0.400	0.001
Kerosene							
sGARCH-norm	1	− 3.438	1	1	− 3.438	1	0.001
Heating Oil							
sGARCH-norm	1	− 0.833	1	1	− 0.462	1	0.001
sGARCH-ged	2	− 0.323	1	2	0.462	0.920	0.001
sGARCH-snorm	3	0.746	0.748	3	0.853	0.718	0.001
sGARCH-sstd	4	1.255	0.399	4	1.209	0.481	0.001

For 99% instead, Heating Oil had only few models eliminated from it, but Kerosene instead got all models eliminated, but not the one with the normal distribution.

6 VaR and Backtesting procedure for GARCH MIDAS models

The results for Garch-Midas models are very interesting. As indicated in the table (see table 5), none of the commodities has passed all tests under all distributions. The variables WTI and Propane are intriguing, as you can see, in fact, only the Garch-Midas with standard t distribution and without the skew parameter for the Propane and the Garch-Midas with standard t distributiion and with the skew parameter for the WTI have not passed the Kupiec's test and the DQ test respectively, unlike the other models that showed excellent results for all tests. Another interesting thing is that we did not obtain results for any model regarding the Gasoline variable because the system itself failed to produce them. We observe the same thing for the Europe Brent but only under the skew and not skew student-t distributions.

Table 5: backtesting results for Midas. Critical values of the test: 95% Kupiec = 3.841, Christoffersen = 5.991; 99% Kupiec = 6.635, Christoffersen = 9.21

	95%:			*99%:*		
	Kupiec	Christoffersen	DQ	Kupiec	Christoffersen	DQ
WTI						
GMnormYES	0.6266953	1.180512	2.953496	0.4735675	0.4844965	0.9112693
GMnormNO	2.450099	2.729465	2.90965	0.4735675	0.4844965	0.9277523
GMstdYES	2.450099	2.729465	2.778836	0.4735675	0.4844965	0.7890147
GMstdNO	3.948656*	4.126448	3.594531	0.4735675	0.4844965	0.8105726
Europe Brent						
GMnormYES	2.450099	2.729465	2.779574	0.4735675	0.4844965	0.7883884
GMnormNO	5.963605*	6.063057*	4.974881	0.4735675	0.4844965	0.7770511
Heating Oil						
GMnormYES	0.1859573	1.110704	5.786948	0.6077959	0.707248	4.216081
GMnormNO	8.66914*	8.713097*	6.183525	0.01196905	0.05592598	1.223984
GMstdYES	2.450099	2.729465	3.768574	0.6077959	0.707248	3.372346
GMstdNO	8.66914*	8.713097*	6.035356	0.4735675	0.4844965	0.4261432
Gasoline						
GMnormYES	-	-	-	-	-	
GMnormNO	-	-	-	-	-	-
GMstdYES	-	-	-	-	-	-
GMstdNO	-	-	-	-	-	-
Kerosene						
GMnormYES	0.007173915	0.9332967	14.7732*	0.01196905	0.05592598	54.41285*
GMnormNO	1.365364	1.769935	12.08837*	0.4735675	0.4844965	0.7197736
GMstdYES	1.365364	1.769935	13.104744*	0.01196905	0.05592598	54.70696*
GMstdNO	3.948656*	4.126448	12.84762*	-	-	-
Propane						
GMnormYES	0.007173915	0.9332967	2.790013	0.6077959	0.707248	2.835834
GMnormNO	0.6266953	1.180512	3.294767	0.6077959	0.707248	3.056768
GMstdYES	0.007173915	0.9332967	13.58922*	0.01196905	0.05592598	3.35396
GMstdNO	2.450099	2.729465	4.78798	0.6077959	0.707248	7.756845

**The null hypothesis is rejected at the 5% significance level.*

7 MCS for MIDAS

We utilized the same technique for Garch-MIDAS MCS as we did for Garch MCS. Because of the relationship with "death weekly," the R code has been revised in this situation.

Table 6: MCS procedure for Garch Midas of WTI returns

95%	Rank_M	v_M	MCS_M	Rank_R	v_R	MCS_R	Loss
WTI							
GMnormYES	2	− 0.065	1	2	0.767	0.651	0.003
GMnormNO	3	1.290	0.272	3	1.551	0.201	0.003
GMstdYES	1	− 0.990	1	1	− 0.767	1	0.003
Europe Brent							
GMnormYES	1	− 1.542	1	1	− 1.542	1	0.003
Kerosene							
GMstdYES	1	− 0.303	1	1	− 0.303	1	0.003
GMnormNO	2	0.303	0.735	2	0.303	0.735	0.003
Propane							
GMnormYES	1	− 0.867	1	1	− 0.867	1	0.003
GMnormNO	2	0.867	0.386	2	0.867	0.386	0.003
Heating Oil							
GMstdYES	1	− 1.300	1	1	− 1.300	1	0.003
GMnormNO	2	1.300	0.205	2	1.300	0.205	0.003

We applied the MCS procedure at 95% in a different R way then we did before. We computed a for loop with all the models in this way:

```
models1<- c("GM")

specifications1<- c("norm", "std")

Skew <- c("YES", "NO")

spec.compM<- c()

for(m in models1){
  for(s in specifications1){
    for (k in Skew){
      spec.compM[[paste( m, s, k, sep = "" )]] <-
        ugmfit(model=m,skew=k,distribution=s,
               WTI2_weekly['2020-04-06/'],
               mv_matWTI2a,K=4, out_of_sample = 185)
    }
  }
}
```

The "mv_matWTI2a" is a matrix related to the weekly Covid deaths and the WTI returns considered as weekly frequencies. We then applied the command we have up here and then did almost the same procedure we did before with the GARCH models, but without the ugarchroll, but with this command:

```
specifications1 <- names( spec.compM )
roll.compM_95 <- list()
for(s in specifications1){
  roll.compM\_95[[s]]<- qnorm(.05)*spec.compM[[s]]\$est_vol_oos
}
```

and then we proceded with the same way we did before either at 95% and 99$.

Table 7: MCS procedure for Garch Midas of returns

99%	Rank_M	v_M	MCS_M	Rank_R	v_R	MCS_R	Loss
WTI							
GMnormYES	2	− 0.051	1	3	0.670	0.854	0.001
GMnormNO	4	0.755	0.696	4	1.469	0.304	0.001
GMstdYES	1	− 0.975	1	1	− 0.368	1	0.001
GMstdNO	3	0.076	0.993	2	0.368	0.948	0.001
Europe Brent							
GMnormYES	1	− 2.689	1	1	− 2.689	1	0.001
Kerosene							
GMstdYES	2	0.581	0.558	2	0.581	0.558	0.001
GMnormNO	1	− 0.581	1	1	− 0.581	1	0.001
Propane							
GMnormYES	2	− 0.597	1	2	0.042	1	0.001
GMnormNO	3	− 0.265	1	3	1.370	0.454	0.001
GMstdYES	4	0.962	0.499	4	1.412	0.420	0.001
GMstdNO	1	− 0.789	1	1	− 0.042	1	0.001
Heating Oil							
GMstdYES	1	− 0.714	1	1	− 0.714	1	0.003
GMnormYES	2	0.714	0.485	2	0.714	0.485	0.003

8 Mixed MCS procedure

In the end we tried to apply the same procedure for both the sGARCH and GARCH Midas models that passed their MCS, combined. We saw that most of the times the models that were cleaned were the Midas ones. In R we applied the procedure not with the for loop, but instead, we broke down models into singular parts and we chose only the one that passed their MCS procedure. We then applied the same procedure as we did before for the different kind of models, and we eliminated the ones that were not good with the same procedure we did before.

Table 8: Mixed MCS procedure for Midas and sGARCH at 95%

95%	Rank_M	v_M	MCS_M	Rank_R	v_R	MCS_R	Loss
WTI							
GMstdYES	5	1.355	0.388	5	1.838	0.261	0.003
sGARCHnorm	1	− 3.048	1	1	− 1.173	1	0.003
sGARCHstd	3	− 0.132	1	4	1.818	0.273	0.003
sGARCHged	2	− 2.836	1	2	1.173	0.723	0.003
sGARCHsstd	4	0.465	0.913	3	1.790	0.286	0.003
Europe Brent							
GMnormYES	6	1.247	0.570	5	1.548	0.419	0.003
sGARCHnorm	1	− 2.233	1	1	− 0.654	1	0.003
sGARCHsnorm	2	− 0.867	1	6	1.716	0.312	0.003
sGARCHstd	5	− 0.414	1	3	0.933	0.874	0.003
sGARCHged	4	− 0.771	1	4	0.961	0.858	0.003
sGARCHsstd	3	− 0.843	1	2	0.654	0.971	0.003
Propane							
GMnormYES	3	− 0.477	1	2	0.599	0.985	0.003
GMnormNO	4	− 0.421	1	3	0.709	0.970	0.003
sGARCHnorm	2	− 0.512	1	7	2.089	0.201	0.003
sGARCHsnorm	7	0.855	0.677	6	1.910	0.285	0.004
sGARCHstd	5	0.413	0.921	4	0.917	0.919	0.004
sGARCHged	1	− 1.530	1	1	− 0.599	1	0.003
sGARCHsstd	6	0.818	0.701	5	1.862	0.311	0.004
Gasoline							
sGARCHnorm	2	− 0.423	1	4	2.083	0.127	0.002
sGARCHsnorm	4	0.978	0.565	3	1.800	0.233	0.002
sGARCHged	1	− 2.053	1	1	− 1.583	1	0.002
sGARCHsstd	3	0.952	0.583	2	1.583	0.357	0.002
Heating Oil							
GMstdYES	2	1.288	0.202	2	1.288	0.202	0.003
sGARCHged	1	− 1.288	1	1	− 1.288	1	0.002
Kerosene							
GMstdYES	4	0.932	0.651	3	1.385	0.562	0.003
GMnormNO	5	1.367	0.349	4	1.734	0.327	0.003
sGARCHnorm	3	− 1.258	1	2	1.066	0.804	0.003
sGARCHged	1	− 2.011	1	1	− 1.066	1	0.003
sGARCHsstd	2	− 1.486	1	5	2.039	0.184	0.003

Table 9: Mixed MCS procedure for Midas and sGARCH at 99%

99%	Rank_M	v_M	MCS_M	Rank_R	v_R	MCS_R	Loss
WTI							
GMstdYES	5	1.221	0.420	4	3.191	0.009	0.001
GMstdNO	4	1.135	0.506	3	2.578	0.047	0.001
sGARCHnorm	1	− 3.443	1	1	− 1.483	1	0.001
sGARCHsnorm	3	0.637	0.786	2	1.483	0.388	0.001
sGARCHged	2	− 1.846	1	5	6.407	0	0.001
Europe Brent							
sGARCHnorm	1	− 3.082	1	1	− 3.082	1	0.001
Propane							
GMstdYES	9	1.217	0.496	8	1.811	0.459	0.001
GMnormYES	7	0.063	1	9	2.017	0.334	0.001
GMstdNO	4	− 0.545	1	4	0.666	0.994	0.001
GMnormNO	3	− 0.889	1	2	0.062	1	0.001
sGARCHnorm	6	− 0.068	1	5	0.902	0.966	0.001
sGARCHsnorm	8	0.565	0.916	6	1.206	0.853	0.001
sGARCHstd	1	− 0.909	1	1	− 0.062	1	0.001
sGARCHged	2	− 0.903	1	3	0.249	1	0.001
sGARCHsstd	5	− 0.127	1	7	1.581	0.609	0.001
Gasoline							
sGARCHnorm	2	0.379	0.572	2	0.379	0.572	0.001
sGARCHsnorm	1	− 0.379	1	1	− 0.379	1	0.001
Heating Oil							
GMstdYES	4	− 0.494	1	2	0.057	1.000	0.001
GMnormYES	6	1.653	0.235	6	1.700	0.383	0.001
sGARCHnorm	1	− 1.155	1	1	− 0.057	1	0.001
sGARCHsnorm	3	− 0.630	1	4	0.883	0.878	0.001
sGARCHged	2	− 0.721	1	3	0.477	0.984	0.001
sGARCHsstd	5	0.113	0.999	5	1.221	0.693	0.001
Kerosene							
GMstdYES	3	0.954	0.519	2	1.299	0.400	0.001
GMnormNO	2	0.141	0.926	3	2.907	0.002	0.001
sGARCHnorm	1	− 1.877	1	1	− 1.299	1	0.001

9 Appendix

```
######## First let's analize if we can use the data as we have them by
#analyzing the stationarity ###

adf.test(data\$WTI)

#From the result we see that is non stationary so
we compute the returns
#of the WTI by applying the difference of the logarithm,
but first let's remove the
#value in the 36th row because it's negative and we
can't do the logarithm of something negative

data <- data[-36,]

WTI2<-diff(log(data\$WTI))

adf.test(WTI2) #we see that p-value is <0.05 so
we refuse the H0 that is the non stationarity hypothesis

# Let's compute all the other tests of our variable

summary<-function(data){
  list (
    Mean=mean(data) ,
    skewness=skewness(data) ,
    kurtosis=kurtosis(data) ,
    Min=min(data) ,
    JB=jarque.bera.test(data) ,
    ST=shapiro.test(as.vector(data)) ,
    LB=Box.test ( (data^2) ,lag=20, type="Ljung-Box" ) ,
    ARCHLM=ArchTest(data, lags=20) ,
    ADF=adf.test(data))
}

summary(WTI2)

######## Let's plot all the variables that we have ###########


par(mfrow=c(1,2))

plot(WTI_weekly, main="WTI") ; plot(WTI2_weekly, main="WTI returns")

##################################### spec model #####################

spec1<-ugarchspec(variance.model = list(model="sGARCH",
                                        garchOrder=c(1,1)), mean.model =
                    list(armaOrder=c(0,0)),
                  distribution.model = "norm")

spec2<-ugarchspec(variance.model = list(model="sGARCH",
                                         garchOrder=c(1,1)), mean.model =
                    list(armaOrder=c(0,0)),
                  distribution.model = "snorm")

spec3<- ugarchspec(variance.model = list(model="sGARCH",
                                          garchOrder=c(1,1)), mean.model =
                     list(armaOrder=c(0,0)),
                   distribution.model = "std")

spec4 <- ugarchspec(variance.model = list(model="sGARCH",
                                           garchOrder=c(1,1)), mean.model =
                      list(armaOrder=c(0,0)),
                    distribution.model = "ged")

spec5 <-ugarchspec(variance.model = list(model="sGARCH",
                                          garchOrder=c(1,1)), mean.model =
                     list(armaOrder=c(0,0)),
                   distribution.model = "sstd")

############### GARCH model #################

garchmodel <-function(data, spec){
  ugarchfit(data, spec=spec, out.sample = 20)
}

Wti_mod1<-garchmodel(WTI2, spec1)
Wti_mod2<-garchmodel(WTI2, spec2)
Wti_mod3<-garchmodel(WTI2, spec3)
Wti_mod4<-garchmodel(WTI2, spec4)
Wti_mod5<-garchmodel(WTI2, spec5)

########## VaR function at 95\% ###############

var_f<-function(Spec, data){
  var_t<- ugarchroll(Spec, data=data, n.ahead=1, n.start=130,
                     refit.every = 5, refit.window = "rolling",
                     calculate.VaR = TRUE, VaR.alpha = c(0.05),
                     keep.coef = TRUE)
  plot(var_t, which= 4, VaR.alpha=0.05)
  report(var_t, type="VaR", VaR.alpha=0.05, conf.level=0.95)
  x=var_t@forecast\$VaR
  Alpha_5 =x[, 1]
  DQtest(data, Alpha_5, VaR_level = 0.95)
}

var_f(spec1, WTI2)
var_f(spec2, WTI2)
var_f(spec3, WTI2)
var_f(spec4, WTI2)
var_f(spec5, WTI2)

########### VaR function at 99\% ###############

var_f2<-function(Spec, data){
  var_t<- ugarchroll(Spec, data=data, n.ahead=1, n.start=130,
                     refit.every = 5, refit.window = "rolling",
                     calculate.VaR = TRUE, VaR.alpha = c(0.01))
  plot(var_t, which= 4, VaR.alpha=0.01)
  report(var_t, type="VaR", VaR.alpha=0.01, conf.level=0.99)
  print(class(var_t))
  x=var_t@forecast\$VaR
  Alpha_1 =x[, 1]
  DQtest(data, Alpha_1, VaR_level = 0.99)
}

var_f2(spec1, WTI2)
var_f2(spec2, WTI2)
var_f2(spec3, WTI2)
var_f2(spec4, WTI2)
var_f2(spec5, WTI2)


######### Let's make a for loop of all the type of sGARCH at 95\% ###########


models <- c("sGARCH")
distributions <- c("norm", "std", "ged", "snorm", "sstd")
spec.comp <- list()
for( m in models ) {
  for( d in distributions ) {
    spec.comp[[paste( m, d, sep = "-" )]] <-
      ugarchspec(mean.model = list(armaOrder = c(0, 0)),
                 variance.model = list(model = m, garchOrder = c(1, 1)),
                 distribution.model=d)
  }
}

############################ WTI returns ###########################

specifications <- names( spec.comp )
roll.comp <- list()
for( s in specifications ){
  roll.comp[[s]] <- ugarchroll(spec = spec.comp[[s]], data = WTI2,
                              n.ahead = 1, n.start=130, refit.every=5,
                              calculate.VaR = TRUE,
                              refit.window = "rolling",
                               VaR.alpha = c(0.05,0.01))
}

############### Let's compute as we did before,
#but this time with a for loop the VaR test ############

Var.test1<-list()
for(s in specifications){
  Var.test1[[s]] =VaRTest(alpha=0.05,roll.comp[[s]]@forecast[["VaR"]]
                          [["realized"]],
                          roll.comp[[s]]@forecast[["VaR"]][["alpha(5\%)"]])
}
show(Var.test1)

Var.test2<-list()
for(s in specifications){
  Var.test2[[s]] =VaRTest(alpha= 0.01,roll.comp[[s]]@forecast[["VaR"]]
                          [["realized"]],
                          roll.comp[[s]]@forecast[["VaR"]][["alpha(1\%)"]])
}
show(Var.test2)


VaR.comp=list()
for( s in specifications ) {
  VaR.comp[[s]] <- as.data.frame(roll.comp[[s]], which = "VaR")[, 1]
}

Loss <- do.call(cbind,lapply(specifications,
                             function(s) LossVaR(tau=0.05,
                                                  realized=tail(WTI2, 185),
                                                  evaluated=VaR.comp[[s]])))

colnames(Loss) <- specifications

VaR.comp2=list()
for( s in specifications ) {
  VaR.comp2[[s]] <- as.data.frame(roll.comp[[s]], which = "VaR")[, 2]
}

Loss2 <- do.call(cbind,lapply(specifications,
                              function(s) LossVaR(tau=0.01,
                                                   realized=tail(WTI2, 185),
                                                   evaluated=VaR.comp2[[s]])))

colnames(Loss2) <- specifications

SSMWTI2_95 <- MCSprocedure(Loss = Loss, alpha = 0.2, B = 5000,
                           statistic = "Tmax")
```

```
SSMWTI2_99 <- MCSprocedure(Loss = Loss2, alpha = 0.2, B = 5000,
                           statistic = "Tmax")

stargazer(SSMWTI2_95@show, type="latex")

stargazer(SSMWTI2_95@show, type="latex")

##########################################################
#################### GARCH Midas ####################
##########################################################

d=data2\$`Deaths USA`

Y2<-ts(d)

date2<-as.Date(data2\$...1)

Death_weekly_<- apply.weekly(Y2, sum)

Death_weekly <- as.xts(coredata(Death_weekly_),order.by = date2,

                       by = "week", length.out = length(Death_weekly_))

################## Remove one date 'cause we
#are talking about returns ################################

d<- data\$...1

d2<- d[-1]

date_ <- as.Date(d)

date <- as.Date(d2)

######## Turn all the variables to time series #############

WTI_TS <- ts(data\$WTI)

WTI2_TS<- ts(WTI2)

#################### apply weekly #####################

WTI_weeklysum<- apply.weekly(WTI_TS, sum)

####################apply weekly for returns #################

WTI2_weeklysum<- apply.weekly(WTI2_TS, sum)

###################### xts ##################################
```

```
xtsfunction_ <- function(x){
  as.xts(coredata(x), order.by=date_,
         by = "week", length.out = length(x))
}

WTI_weekly <- xtsfunction_(WTI_weeklysum)

####################xts for returns #######################

xtsfunction <- function(x){
  as.xts(coredata(x), order.by=date,
         by = "week", length.out = length(x))
}

WTI2_weekly <- xtsfunction(WTI2_weeklysum)

############### Let's write the matrices #################

mv_mat_ <- function (data2){
  mv_into_mat(data2['2020-04-06/'],
              diff(Death_weekly),
              K=4,type="weekly")
}

mv_matWTI2a<-mv_mat_(WTI2_weekly)

############### Let's write the models manually ##############
###############K=4 STD DISTRIBUTION ###################

fit<-function(data,mvmat_){
  ugmfit(model="GM",skew="YES",distribution="std",
         data['2020-04-06/'],
         mvmat_,K=4, out_of_sample = 185)
}

WTI2_m3<-fit(WTI2_weekly, mv_matWTI2a)

#############K=4 NORM DISTRIBUTION #########

fit2<-function(data, mvmat_){
  ugmfit(model="GM",skew="YES",distribution="norm",
         data['2020-04-06/'],
         mvmat_,K=4, out_of_sample = 185)
}

WTI2_m4<-fit2(WTI2_weekly, mv_matWTI2a)

############ K=4 STD DISTRIBUTION no skew ##############

fit3<-function(data,mvmat_){
   ugmfit(model="GM",skew="NO",distribution="std",
              data['2020-04-06/'],
              mvmat_,K=4, out_of_sample = 185)
}


WTI2_m5<-fit3(WTI2_weekly, mv_matWTI2a)

############## K=4 NORM DISTRIBUTION no skew ####################

fit4<-function(data, mvmat_){
   ugmfit(model="GM",skew="NO",distribution="norm",
              data['2020-04-06/'],
              mvmat_,K=4, out_of_sample = 185)
}

WTI2_m6<-fit4(WTI2_weekly, mv_matWTI2a)


################# As we did before let's do a for
# loop for the MCS procedure ###############

models1<- c("GM")

specifications1<- c("norm", "std")

Skew <- c("YES", "NO")

spec.compM<- c()

for(m in models1){
   for(s in specifications1){
      for (k in Skew){
         spec.compM[[paste( m, s, k, sep = "" )]] <-
            ugmfit(model=m,skew=k,distribution=s,
                     WTI2_weekly['2020-04-06/'],
                     mv_matWTI2a,K=4, out_of_sample = 185)
      }
   }
}

View(spec.compM)

specifications1 <- names( spec.compM )
roll.compM_95 <- list()
for(s in specifications1){
   roll.compM_95[[s]]<- qnorm(.05)*spec.compM[[s]]\$est_vol_oos
}

DQtestMWTI2_95 <- list()
```

```
Var.testMWTI2_95<-list()
for(s in specifications1){
  Var.testMWTI2_95[[s]]<-VaRTest(alpha = .05, actual = WTI2_TS[131:315],
                                VaR = as.numeric(roll.compM_95[[s]]),
                                conf.level = .95)
  DQtestMWTI2_95[[s]]<-DQtest(WTI2_TS, as.numeric(roll.compM_95[[s]]),
                             0.95)
}



specifications1 <- names( spec.compM )
roll.compM_99 <- list()
for(s in specifications1){
  roll.compM_99[[s]]<- qnorm(.01)*spec.compM[[s]]\$est_vol_oos
}

DQtestMWTI2_99<- list()
Var.testMWTI2_99<-list()
for(s in specifications1){
  Var.testMWTI2_99[[s]]<-VaRTest(alpha = .01, actual = WTI2_TS[131:315],
                                VaR = as.numeric(roll.compM_99[[s]]),
                                conf.level = .99)
  DQtestMWTI2_99[[s]]<-DQtest(WTI2_TS, as.numeric(roll.compM_99[[s]]),
                             0.99)
}


VaR.compMWTI2_95=list()
for( s in specifications1 ) {
  VaR.compMWTI2_95[[s]] <- as.data.frame(roll.compM_95[[s]],
                                        which = "VaR")[, 1]
}

VaR.compMWTI2_99=list()
for( s in specifications1 ) {
  VaR.compMWTI2_99[[s]] <- as.data.frame(roll.compM_99[[s]],
                                        which = "VaR")[, 1]
}


LossM95 <- do.call(cbind,lapply(specifications1,
                                function(s) LossVaR(tau=0.05,
                                        realized=tail(WTI2, 185),
                                    evaluated=VaR.compMWTI2_95[[s]]))))
LossM99 <- do.call(cbind,lapply(specifications1,
                                function(s) LossVaR(tau=0.01,
                                     realized=tail(WTI2, 185),
                                evaluated=VaR.compMWTI2_99[[s]])))
```

```
colnames(LossM95) <- specifications1

colnames(LossM99) <- specifications1

SSMWTI2M95<- MCSprocedure(Loss = LossM95, alpha = 0.2, B = 5000,
                          statistic = "Tmax")

SSMWTI2M99<- MCSprocedure(Loss = LossM99, alpha = 0.2, B = 5000,
                          statistic = "Tmax")

stargazer(SSMWTI2M95@show, type="latex")

stargazer(SSMWTI2M99@show, type="latex")

############### MIXED MCS PROCEDURE ###############

specifications_<-c( "GMstdYES","GMnormYES",  "GMnormNO", "sGARCHnorm",
                   "sGARCHstd", "sGARCHged", "sGARCHsstd")

var.t1<-ugarchroll(spec = spec1, data = WTI2,
                   n.ahead = 1, n.start=130, refit.every=5,
                   calculate.VaR = TRUE,
                   refit.window = "rolling", VaR.alpha = c(0.05))
var.t3<-ugarchroll(spec = spec3, data = WTI2,
                   n.ahead = 1, n.start=130, refit.every=5,
                   calculate.VaR = TRUE,
                   refit.window = "rolling", VaR.alpha = c(0.05))
var.t4<-ugarchroll(spec = spec4, data = WTI2,
                   n.ahead = 1, n.start=130, refit.every=5,
                   calculate.VaR = TRUE,
                   refit.window = "rolling", VaR.alpha = c(0.05))
var.t5<-ugarchroll(spec = spec5, data = WTI2,
                   n.ahead = 1, n.start=130, refit.every=5,
                   calculate.VaR = TRUE,
                   refit.window = "rolling", VaR.alpha = c(0.05))


v1 <-qnorm(.05)*WTI2_m3\$est_vol_oos
v2<- qnorm(.05)*WTI2_m4\$est_vol_oos
v4 <- qnorm(.05)*WTI2_m6\$est_vol_oos
v5 <- var.t1@forecast\$VaR\$`alpha(5\%)`
v7 <- var.t3@forecast\$VaR\$`alpha(5\%)`
v8 <- var.t4@forecast\$VaR\$`alpha(5\%)`
v9 <- var.t5@forecast\$VaR\$`alpha(5\%)`

VaR.comp_ <- list( GMstdYES=v1, GMnormYES=v2, GMnormNO=v4,
                   sGARCHnorm=v5, sGARCHstd=v7,
                   sGARCHged=v8, sGARCHsstd=v9)

Loss1<- do.call(cbind,lapply(specifications_,
                                   function(s)
                                     LossVaR(tau=0.05, realized=tail(WTI2, 185),
                                             evaluated=VaR.comp_[[s]])))


colnames(Loss1) <- specifications_

Mix_Mcs_WTI295<- MCSprocedure(Loss = Loss1, alpha = 0.2, B = 5000,
                              statistic = "Tmax")

stargazer(Mix_Mcs_WTI295@show, type="latex")

############ we applied the same at 99\% #############

specifications_2<-c( "GMstdYES","GMnormYES", "GMstdNO", "GMnormNO",
"sGARCHnorm","sGARCHsnorm","sGARCHged")

var.t1<-ugarchroll(spec = spec1, data = WTI2,
                   n.ahead = 1, n.start=130, refit.every=5,
                   calculate.VaR = TRUE,
                   refit.window = "rolling", VaR.alpha = c(0.01))
var.t2<-ugarchroll(spec = spec2, data = WTI2,
                   n.ahead = 1, n.start=130, refit.every=5,
                   calculate.VaR = TRUE,
                   refit.window = "rolling", VaR.alpha = c(0.01))
var.t4<-ugarchroll(spec = spec4, data = WTI2,
                   n.ahead = 1, n.start=130, refit.every=5,
                   calculate.VaR = TRUE,
                   refit.window = "rolling", VaR.alpha = c(0.01))


v1 <-qnorm(.01)*WTI2_m3\$est_vol_oos
v2<- qnorm(.01)*WTI2_m4\$est_vol_oos
v3<- qnorm(.01)*WTI2_m5\$est_vol_oos
v4 <- qnorm(.01)*WTI2_m6\$est_vol_oos
v5 <- var.t1@forecast\$VaR\$`alpha(1\%)`
v6 <- var.t2@forecast\$VaR\$`alpha(1\%)`
v8 <- var.t4@forecast\$VaR\$`alpha(1\%)`


VaR.comp_2 <- list( GMstdYES=v1, GMnormYES=v2, GMstdNO=v3, GMnormNO=v4,
                   sGARCHnorm=v5,sGARCHsnorm=v6,
                   sGARCHged=v8)

Loss2<- do.call(cbind,lapply(specifications_2,
                                   function(s)
                                     LossVaR(tau=0.01, realized=tail(WTI2, 185),
                                             evaluated=VaR.comp_2[[s]])))
```

```
colnames(Loss2) <- specifications_2

Mix_Mcs_WTI299<- MCSprocedure(Loss = Loss2, alpha = 0.2, B = 5000,
                              statistic = "Tmax")

stargazer(Mix_Mcs_WTI299@show, type="latex")
```

References

[1] Bernardi M., Catania L. (2014), "The Model Confidence Set package for R"

[2] Bernardi M., Catania L. (2018), "The Model Confidence Set package for R"

[3] Candila V. (2021)"Univariate GARCH-MIDAS, Double-Asymmetric GARCH-MIDAS and MEM-MIDAS"

[4] Compute Variance and Standard Deviation of a value in R Programming – var() and sd() Function" (2020).

[5] Ghalanos A. (2020), "Introduction to the rugarch package (Version 1.4-3)"

[6] Hossein Asgharian, Ai Jun Hou, Farrukh Javed, "Importance of the macroeconomic variables for variance prediction: A GARCH-MIDAS approach"

[7] Jon D.Samuels, Rodrigo M.Sekke (2017), "Model Confidence Sets and forecast combination", Elsevier

[8] Laporta A. G., Merlo L., Petrella L. (2018), "Selection of Value at Risk Models for Energy Commodities", Elsevier

[9] Roccioletti S. (2015), "Backtesting Value at Risk and Expected Shortfall", Springer Gabler

[10] VaR: calculate various Value at Risk (VaR) measures", RDocumentation.org

[11] Weiqian Li (2016), "Value at Risk (VaR) and its calculations: an overview"