GARCH MODEL BTCUSD

Angela Arceo

Data Analysis

The objective of this work is to analyze the volatility of BTCUSD. I download the daily prices from TradingView provided by Coinbase. The minimum date available in this platform was 2015-01-13 so the period of analysis is from 1st January 2015 to 31 December 2020, therefore the total of observations were 2177.

```
btc<-read_csv('BTCUSD_1D_122020.csv')
```

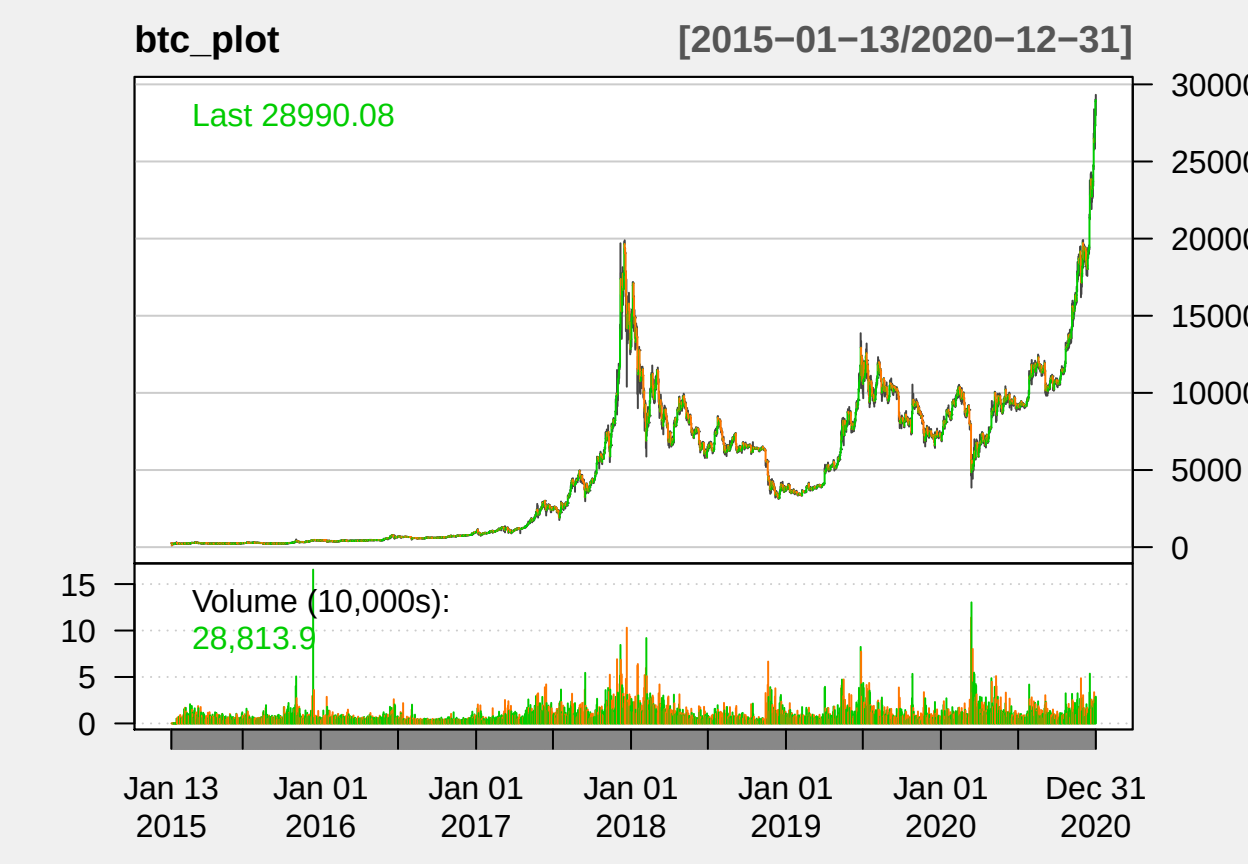
```
## Parsed with column specification:
## cols(
##   time = col_date(format = ""),
##   open = col_double(),
##   high = col_double(),
##   low = col_double(),
##   close = col_double(),
##   Volume = col_double(),
##   `Volume MA` = col_double()
## )
```

```
btc%<>%
  filter(between(time,date('2015-01-13'),date('2020-12-31')))

btc_plot<-timetk::tk_xts(btc%>%select(-`Volume MA`))
```

```
## Warning: Non-numeric columns being dropped: time
```

```
## Using column `time` for date_var.
```

```
quantmod::chartSeries(btc_plot,
          theme=chartTheme('white'))
```



The returns were calculated as the log difference of the close prices. In order to corroborate if the returns are stationary two tests were applied: Dickey-Fuller and KPSS. Dicker-Fuller test for the null hypothesis that returns has a unit root. The result of the test provided evidence that we can reject the null hypothesis with at least 1% of significance level that the returns don't have unit root.

```
btc_lr<-btc%>%
       filter(time > date('2015-01-13'))%>%
       select(time, close)%>%
       mutate(r = (log(close) - log(lag(close))))%>%
       filter(!is.na(r))

  tseries::adf.test(btc_lr$r)
```

```
## Warning in tseries::adf.test(btc_lr$r): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  btc_lr$r
## Dickey-Fuller = -12.329, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary
```

KPSS reverses the null and alternative hypothesis, the returns are stationary against the alternative that the process is integrated of order one. The result of the test provided evidence that we can't reject the null hypothesis with at least 10% of significance level that the returns are stationary.

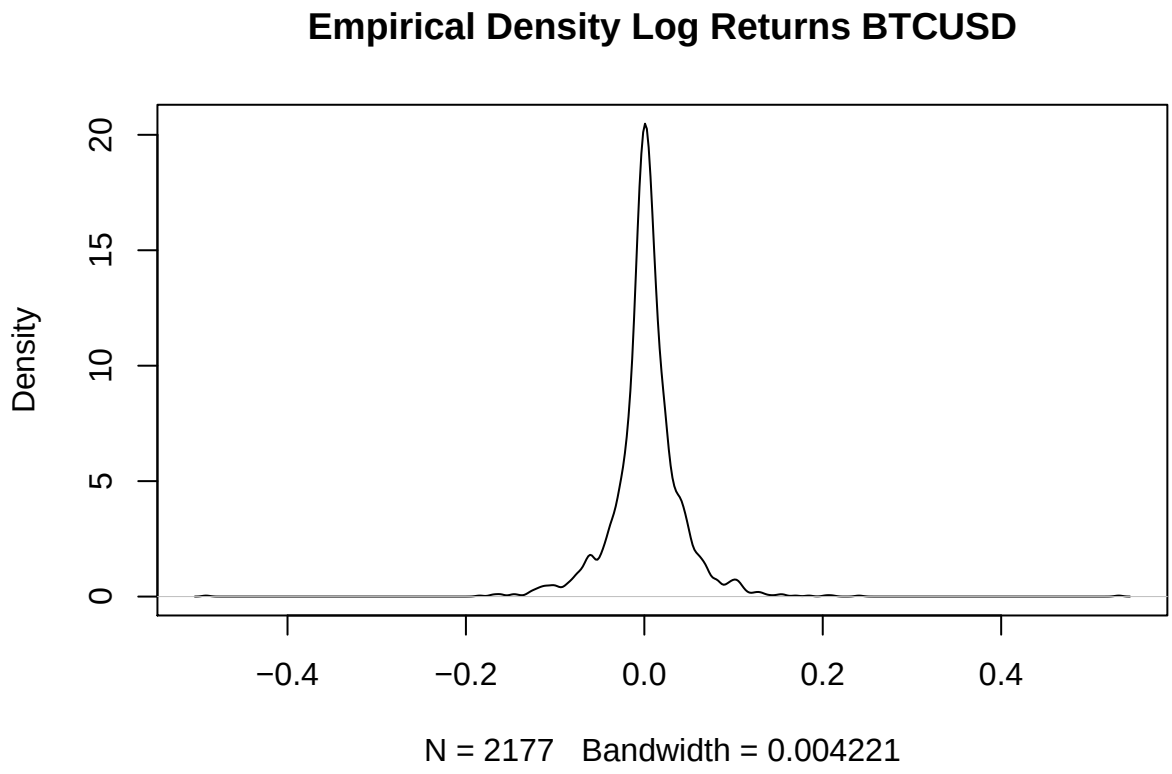
```
   tseries::kpss.test(btc_lr$r)
```

```
## Warning in tseries::kpss.test(btc_lr$r): p-value greater than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  btc_lr$r
## KPSS Level = 0.13085, Truncation lag parameter = 8, p-value = 0.1
```

In addition to statistical test, financial literature mentions that it is a stylized fact that the returns time series present a heavy tails that is a leptokurtic distribution. We can observe that when the empirical density of BTCUSD log returns is plotted. Also, we can prove that the log returns aren't normally distributed through the Jarque-Bera test where it demonstrates that p-value is close to 0 therefore there is evidence that rejects the null hypothesis.

```
plot(density(btc_lr$r),main='Empirical Density Log Returns BTCUSD')
```

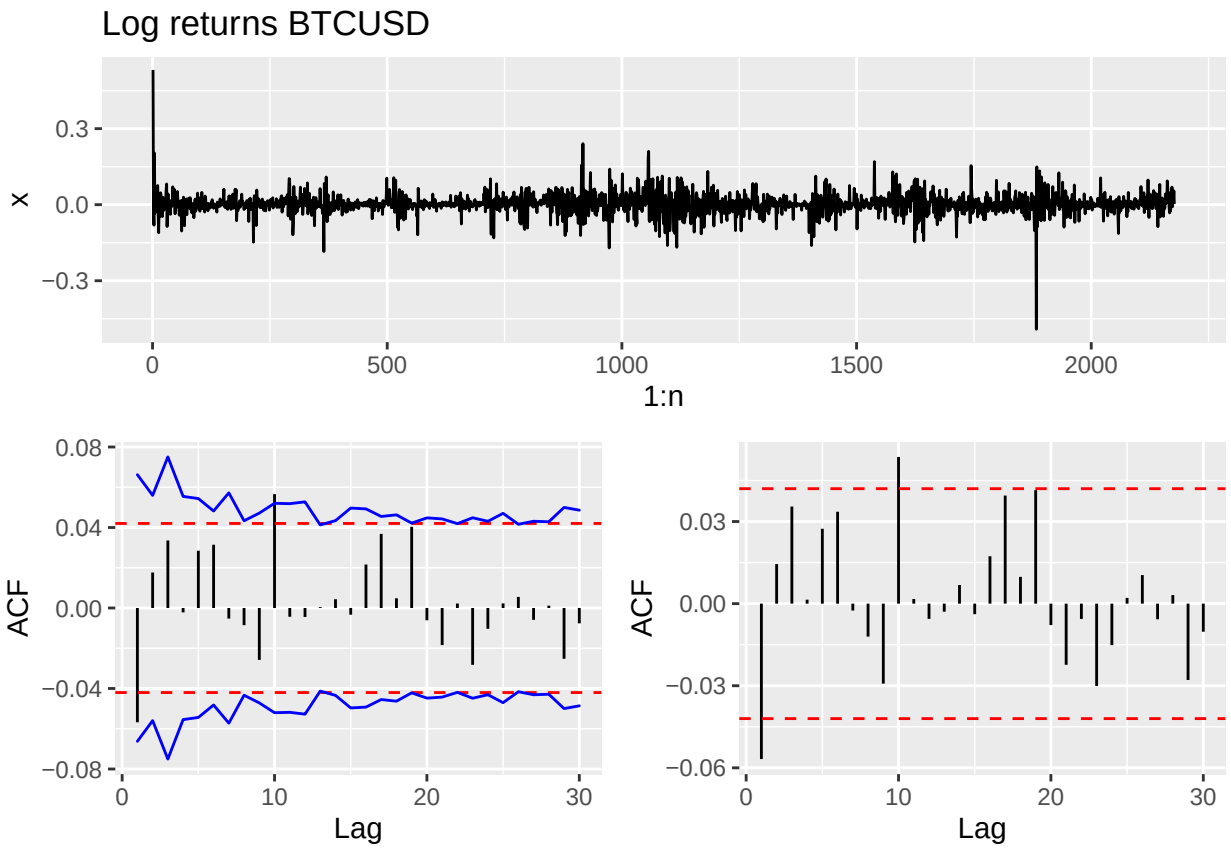



```
tseries::jarque.bera.test(btc_lr$r)
```

```
##
##  Jarque Bera Test
##
## data:  btc_lr$r
## X-squared = 58511, df = 2, p-value < 2.2e-16
```

After the examination of distribution, we can observe that the log returns have been periods of high volatility. The figure below shows the estimated ACF in which the red bands represent the correlation for white noise and the blue ones represent confidence intervals for asymptotic bands proposed by Francq and Zakoïan (2009). The estimated ACF for the log returns are outside of both bands at lag 10. Also, Ljung-Box test was computed to analyze the null hypothesis of independence in log returns which present evidence that the log returns are independent in order four at least 1% of significance level.

```
garch_examine(btc_lr$r,"series")
```

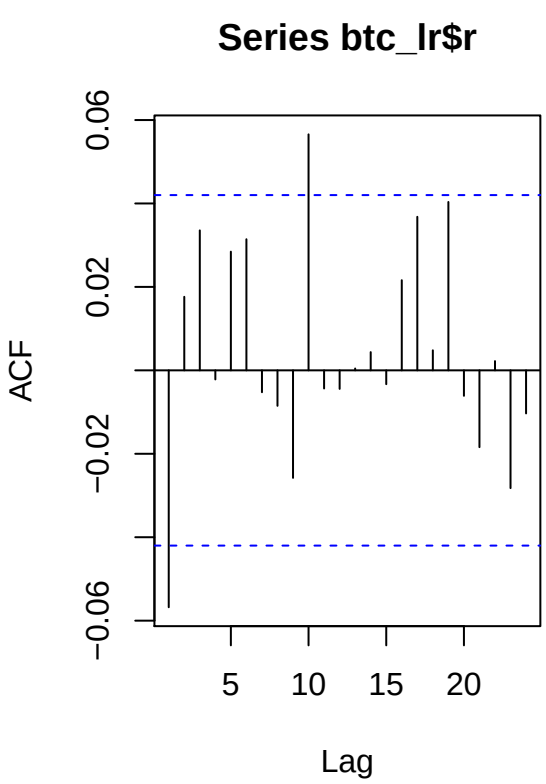


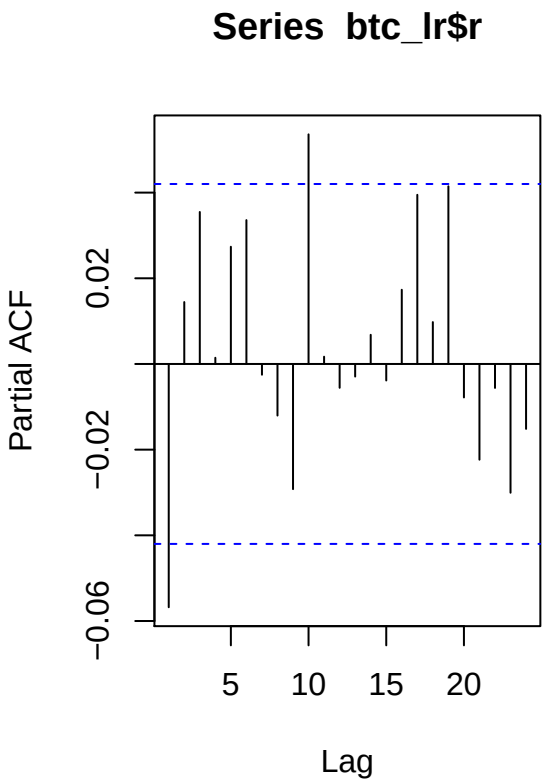

```
Box.test(btc_lr$r,lag=14,type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  btc_lr$r
## X-squared = 22.906, df = 14, p-value = 0.06182
```

The ARCH Engle's test is constructed based on the fact that if the residuals are heteroscedastic, the squared residuals are autocorrelated. In order to analyze ARCH effects an ARMA model was fitted to analyze the residuals. The approach to determine the order of the ARMA model was analyzing PACF to AR(p) and ACF for MA(q).

```
## Registered S3 methods overwritten by 'TSA':
##   method       from
##   fitted.Arima forecast
##   plot.Arima   forecast
```



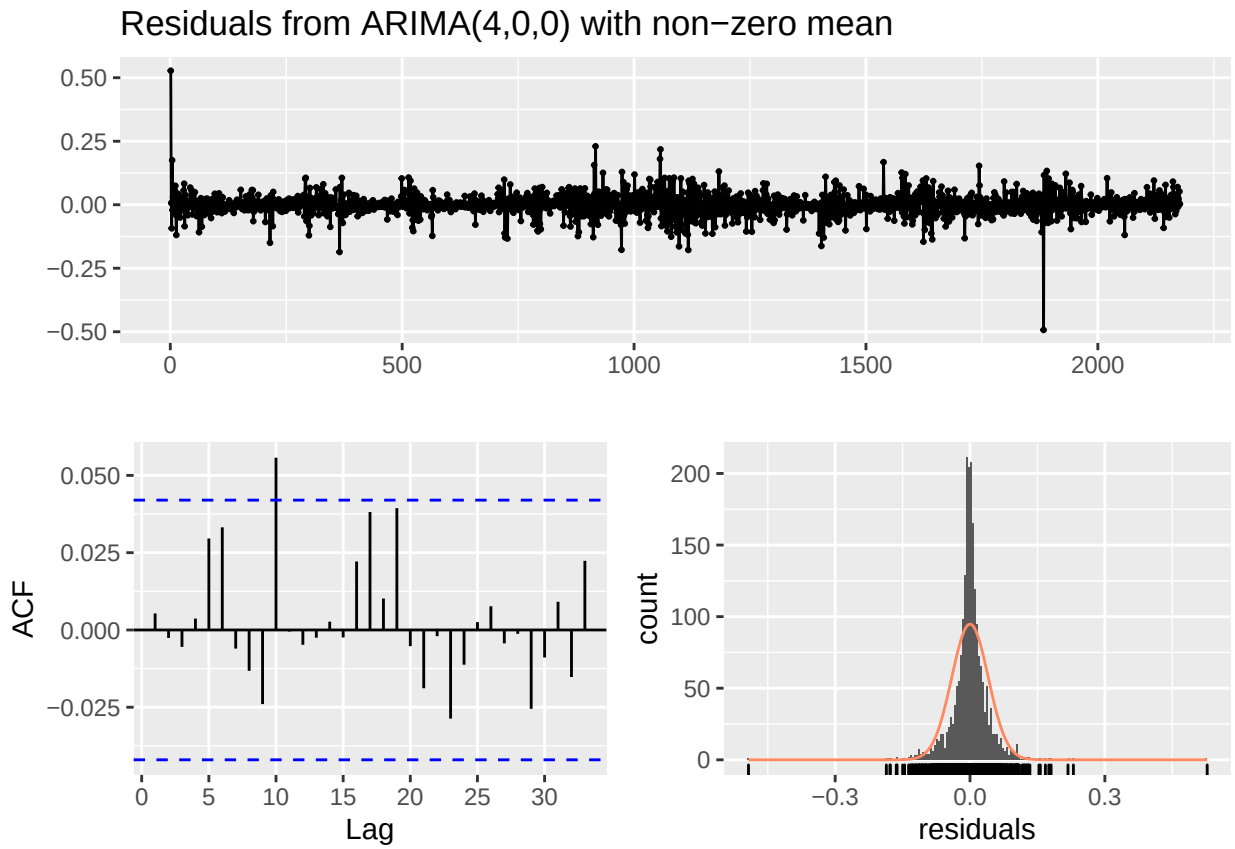


The result of the model was:

$$x = -0.0618 * x_{t-1} + 0.0179x_{t-2} + 0.0394x_{t-3} - 0.0006x_{t-4} + 0.0025$$

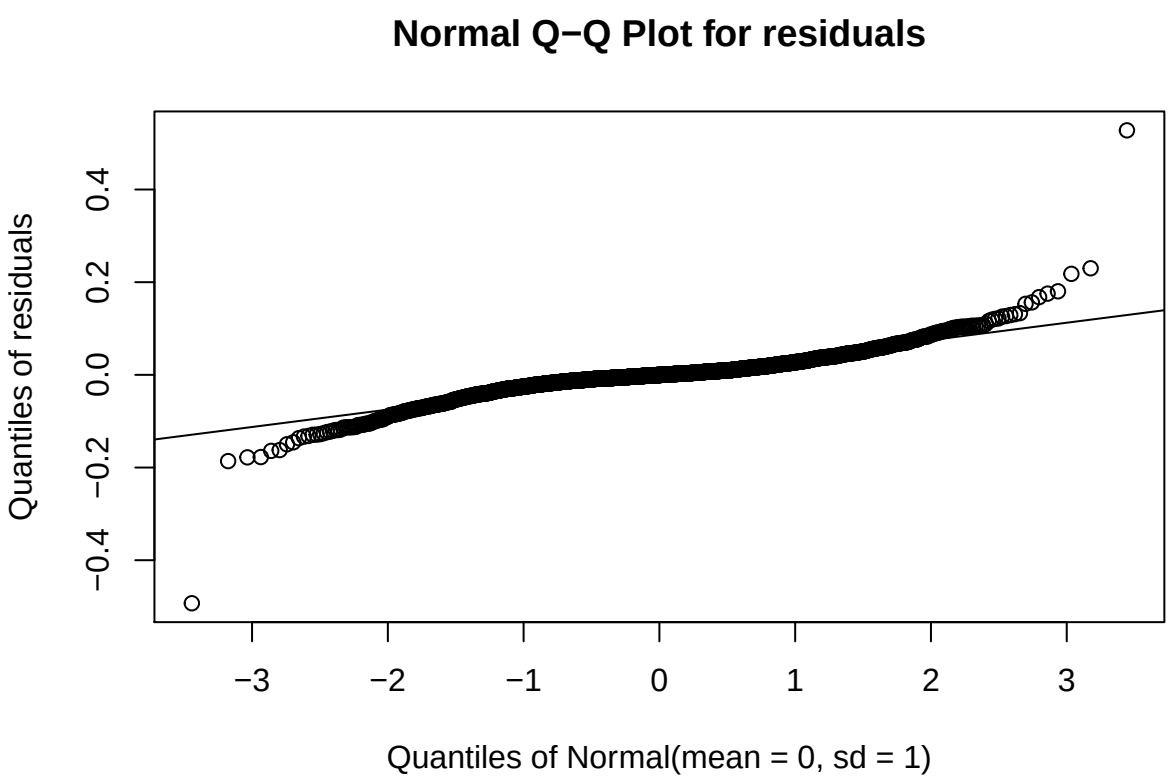
We can observe the empirical distribution and de qqplot of residuals and these ones were not white noise. In addition to the visualizations, the Portmanteau test was performed to test the null hypothesis that the residuals of an ARIMA model are homoscedastic. The test provided evidence at 2.46% significance level and order 16 that the residuals were heteroscedastic.

```
Model.Arima=Arima(btc_lr$r,c(4,0,0))
checkresiduals(Model.Arima)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(4,0,0) with non-zero mean
## Q* = 13.008, df = 5, p-value = 0.0233
##
## Model df: 5.   Total lags used: 10
```

```
residuals<-as.vector(Model.Arima$residuals)
EnvStats::qqPlot(residuals, add.line = TRUE)
```



The evidence of ARCH effects in residuals and leptokurtic distribution in log returns suggest that we can fit a GARCH model for the returns of BTCUSD.

Volatily Model

The volatility of financial asset returns changes over time, with periods when volatility is exceptionally high alternated with periods when volatility is unusually low. The volatility clustering will be easier to identify in intraday data and the generalized autoregressive conditional heteroscedasticity (GARCH) models were designed to capture this phenomenon. Also, GARCH models estimate the conditional variance which depends on the history of returns. The simple ARMA(p,q)-GARCH(m,r) model is:

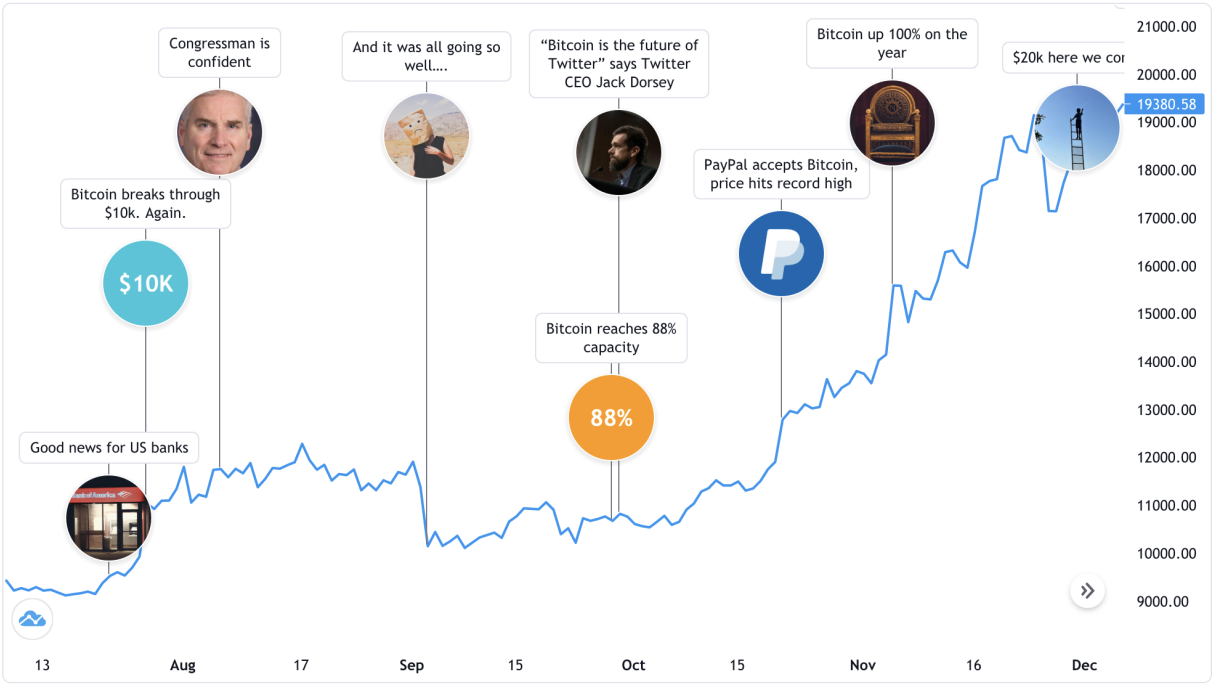
$$Y_t - \mu = \sum_{h=1}^{p} \phi_h (Y_{t-h} - \mu) + \sum_{l=1}^{q} \theta_t V_{t-1} + V_t, V_t = \sigma_t \epsilon_t$$

$$\sigma^2 = \omega + \sum_{i=1}^{m} \alpha_i * V_{t-i}^2 + \sum_{j=1}^{r} \beta_j * \sigma_{t-j}^2$$

Frequently we assume a normal distribution to analyze returns but we previously observed that the log returns have heavy tails. Therefore, we alternated normal distributions, student-t (STD) and generalized (GED) distributions to analyze what is the best fitting for the models.

We observed in the figure below that BTCUSD has been influenced by news about hacks, globally economic, governments acceptance or investor feelings. Also, it is known that news impacts on the volatility of the cryptocurrency as we can see in the below image.The standard GARCH model does not distinguish between positive and negative prediction errors. In reality, the sign of the prediction error matters. The model GJR

GARCH consider an adicional parameter gamma to describe the variance reaction after a negative shock in the returns.



The package *rugarch* was used to fit the GARCH models. The statistical significance of coefficients were tested to evaluate if all the model variables were relevant.Also, we evaluated if the squared residuals autocorrelation were captured by the models through Ljung-Box test. The test provided evidence that the p-value is less than 1% of significance level when the ARMA process wasn't included in the model and a heavy tail distribution was used (see Table 3.)

```
#ARMA-GARCH norm ------------------------------------------------------
btc_lr_m<-btc_lr$r
means= list(list(armaOrder = c(1, 0), include.mean=TRUE),
            list(armaOrder = c(0, 0), include.mean=TRUE))

varianza=list(model = "sGARCH", garchOrder = c(1, 1))


specs<-lapply(means,function(m) ugarchspec(varianza,m,distribution.model = "norm"))
fits<-lapply(specs,function(spec) ugarchfit(spec,btc_lr_m))

# ARMA-GARCH-STD-GED ---------------------------------------------------
mean0= list(armaOrder = c(0, 0), include.mean=TRUE)
var1=list(model = "sGARCH", garchOrder = c(1, 1))
dist<-c("std","ged")

specs_d<-lapply(dist,function(d) ugarchspec(var1,mean0,distribution.model = d,fixed.pars=list(omega=0)))
fit_d<-lapply(specs_d,function(spec) ugarchfit(spec,btc_lr_m))

# GJR-GARCH-STD-GED ----------------------------------------------------
mean0= list(armaOrder = c(0, 0), include.mean=TRUE)
var2=list(model = "gjrGARCH", garchOrder = c(1, 1))
dist<-c("std","ged")
```

Table 1: ARMA(1,0)-GARCH(1,1) norm

	Estimate	Std. Error	t value	Pr(>|t|)
mu	0.0026314	0.0006525	4.033107	0.0000550
ar1	-0.0482376	0.0276485	-1.744677	0.0810411
omega	0.0000783	0.0000119	6.568489	0.0000000
alpha1	0.1630354	0.0196596	8.292904	0.0000000
beta1	0.8061678	0.0181880	44.324162	0.0000000

Table 2: ARMA(0,0)-GARCH(1,1) norm

	Estimate	Std. Error	t value	Pr(>|t|)
mu	0.0026710	0.0006812	3.921118	8.81e-05
omega	0.0000781	0.0000120	6.534814	0.00e+00
alpha1	0.1624553	0.0194924	8.334271	0.00e+00
beta1	0.8067755	0.0180753	44.634124	0.00e+00

```
specs_d_gjr<-lapply(dist,function(d) ugarchspec(var2,mean0,distribution.model = d,fixed.pars=list(omega=
fit_d_gjr<-lapply(specs_d_gjr,function(spec) ugarchfit(spec,btc_lr_m))
```

We can observe in Table 1 that the parameter of ARMA(1,0) was not significantly relevant at 5% of significance level and the coefficient *omega* is close to zero but it was significant so it could be omitted for parsimony principle as result the other parameters weren't affected (Table 2).

```
### Coef
table1<-as.matrix(fits[[1]]@fit$matcoef)
table2<-as.matrix(fits[[2]]@fit$matcoef)

kable(table1,caption = "ARMA(1,0)-GARCH(1,1) norm")%>%
  kable_styling(full_width = F, font_size = 8)
```

```
kable(table2,caption = "ARMA(0,0)-GARCH(1,1) norm")%>%
  kable_styling(full_width = F, font_size = 8)
```

On the other hand, if we analyze the distribution of the residuals , it still presents heavy tails but squared residuals weren't autocorrelated. The order of the GARCH(1,1) was well specified. However, two more models were fitted with distributions like t-student (std) and generalized error distribution (ged) that have heavier tails. Additionally, GJR-GARCH(1,1) model was fitted in order to capture the highest shocks in consequence of the news that provokes high volatility.

```
sqr_f1<-as.numeric((fits[[1]]@fit$residuals)^2)
sqr_f2<-as.numeric((fits[[2]]@fit$residuals)^2)
sqr_f1_d<-as.numeric((fit_d[[1]]@fit$residuals)^2)
sqr_f2_d<-as.numeric((fit_d[[2]]@fit$residuals)^2)
sqr_f1_gjr<-as.numeric((fit_d_gjr[[1]]@fit$residuals)^2)
sqr_f2_gjr<-as.numeric((fit_d_gjr[[2]]@fit$residuals)^2)

auto_corr_sr_pvalues<-matrix(c(Box.test(sqr_f1,type="Ljung-Box")$p.value,
Box.test(sqr_f2,type="Ljung-Box")$p.value,
Box.test(sqr_f1_d,type="Ljung-Box")$p.value,
Box.test(sqr_f2_d,type="Ljung-Box")$p.value,
```

Table 3: Squared Residuals Autocorrelation

	p-value
ARMA(1,0)-GARCH(1,1) norm	0.0343214
ARMA(0,0)-GARCH(1,1) norm	0.0076123
ARMA(0,0)-GARCH(1,1) std	0.0075358
ARMA(0,0)-GARCH(1,1) ged	0.0075293
ARMA(0,0)-GJR-GARCH(1,1) std	0.0075286
ARMA(0,0)-GJR-GARCH(1,1) ged	0.0075177

```
Box.test(sqr_f1_gjr,type="Ljung-Box")$p.value,
Box.test(sqr_f2_gjr,type="Ljung-Box")$p.value
),
dimnames=list(c("ARMA(1,0)-GARCH(1,1) norm",
       "ARMA(0,0)-GARCH(1,1) norm",
       "ARMA(0,0)-GARCH(1,1) std",
       "ARMA(0,0)-GARCH(1,1) ged",
       "ARMA(0,0)-GJR-GARCH(1,1) std",
       "ARMA(0,0)-GJR-GARCH(1,1) ged"),
     c("p-value")

))

kable(auto_corr_sr_pvalues,caption='Squared Residuals Autocorrelation')%>%
  kable_styling()
```

The previous table presented evidence that the order of models were well specified especially when the ARMA process wasn't included. However, the p-values that are minor than 1% and very close between them just like MSE (see annex 3) so these metrics wasn't useful to specify the model with the best performance. The quality of the GARCH prediction was evaluated through the likelihood of the returns that uses densities to measure how likely it is that the observed returns come from the estimated GARCH model, greater is this value is better.

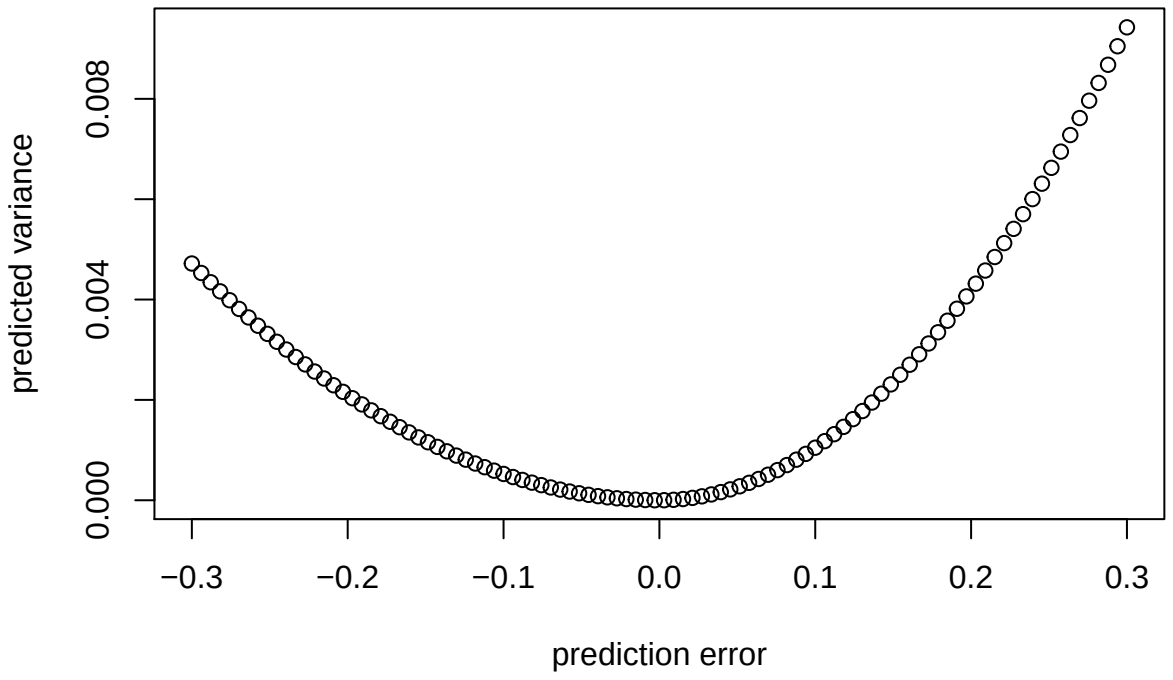
```
l1<-sapply(fits,rugarch::likelihood)
l2<-sapply(fit_d,rugarch::likelihood)
l3<-sapply(fit_d_gjr,rugarch::likelihood)

t_l<-matrix(c(l1,l2,l3))
dimnames(t_l) =list(c("ARMA(1,0)-GARCH(1,1) norm",
                      "ARMA(0,0)-GARCH(1,1) norm",
                      "ARMA(0,0)-GARCH(1,1) std",
                      "ARMA(0,0)-GARCH(1,1) ged",
                      "ARMA(0,0)-GJR-GARCH(1,1) std",
                      "ARMA(0,0)-GJR-GARCH(1,1) ged"),
                    c("Likelihood"))

kable(t_l,caption ='Likelihood')%>%
  kable_styling()
```

The likelihood table show evidence that the GJR-GARCH(1,1) has the highest likelihood. Also, if we review the news impact curve we can observed that the log returns of BTCUSD are more sensitive to positive shocks than negatives, that is, good news provoke high volatility.

Table 4: Likelihood

	Likelihood
ARMA(1,0)-GARCH(1,1) norm	4092.583
ARMA(0,0)-GARCH(1,1) norm	4091.065
ARMA(0,0)-GARCH(1,1) std	4497.729
ARMA(0,0)-GARCH(1,1) ged	4485.924
ARMA(0,0)-GJR-GARCH(1,1) std	4504.974
ARMA(0,0)-GJR-GARCH(1,1) ged	4490.388

```
out<-newsimpact(fit_d_gjr[[1]])
plot(out$zx,out$zy,xlab='prediction error',ylab='predicted variance')
```




```
criteria_fits<-sapply(fits,infocriteria)
criteria_fitd<-sapply(fit_d,infocriteria)
criteria_fit_gjr<-sapply(fit_d_gjr,infocriteria)

table_criteria<-matrix(c(criteria_fits,criteria_fitd,criteria_fit_gjr),nrow=4,ncol=6,byrow=TRUE)
dimnames(table_criteria) =list(c("Akaike","Bayes","Shibata","Hannan-Quinn"),
                               c("ARMA(1,0)-GARCH(1,1) norm",
                                 "ARMA(0,0)-GARCH(1,1) norm",
                                 "ARMA(0,0)-GARCH(1,1) std",
                                 "ARMA(0,0)-GARCH(1,1) ged",
                                 "ARMA(0,0)-GJR-GARCH(1,1) std",
```

Table 5: Information Criteria

	ARMA(1,0)-GARCH(1,1) norm	ARMA(0,0)-GARCH(1,1) norm	ARMA(0,0)-GARCH(1,1) std	ARMA(0,0)-GARCH(1
Akaike	-3.755244	-3.742185	-3.755254	-3
Bayes	-3.754775	-3.750949	-4.128369	-4
Shibata	-4.117523	-4.107077	-4.117530	-4
Hannan-Quinn	-4.134116	-4.129331	-4.120706	-4

```
                                    "ARMA(0,0)-GJR-GARCH(1,1) ged"))

kable(table_criteria, caption="Information Criteria")%>%
kable_styling(full_width = F, font_size = 8)
```

```
btc_plot<-timetk::tk_xts(btc%>%select(-`Volume MA`))
```

```
## Warning: Non-numeric columns being dropped: time
```

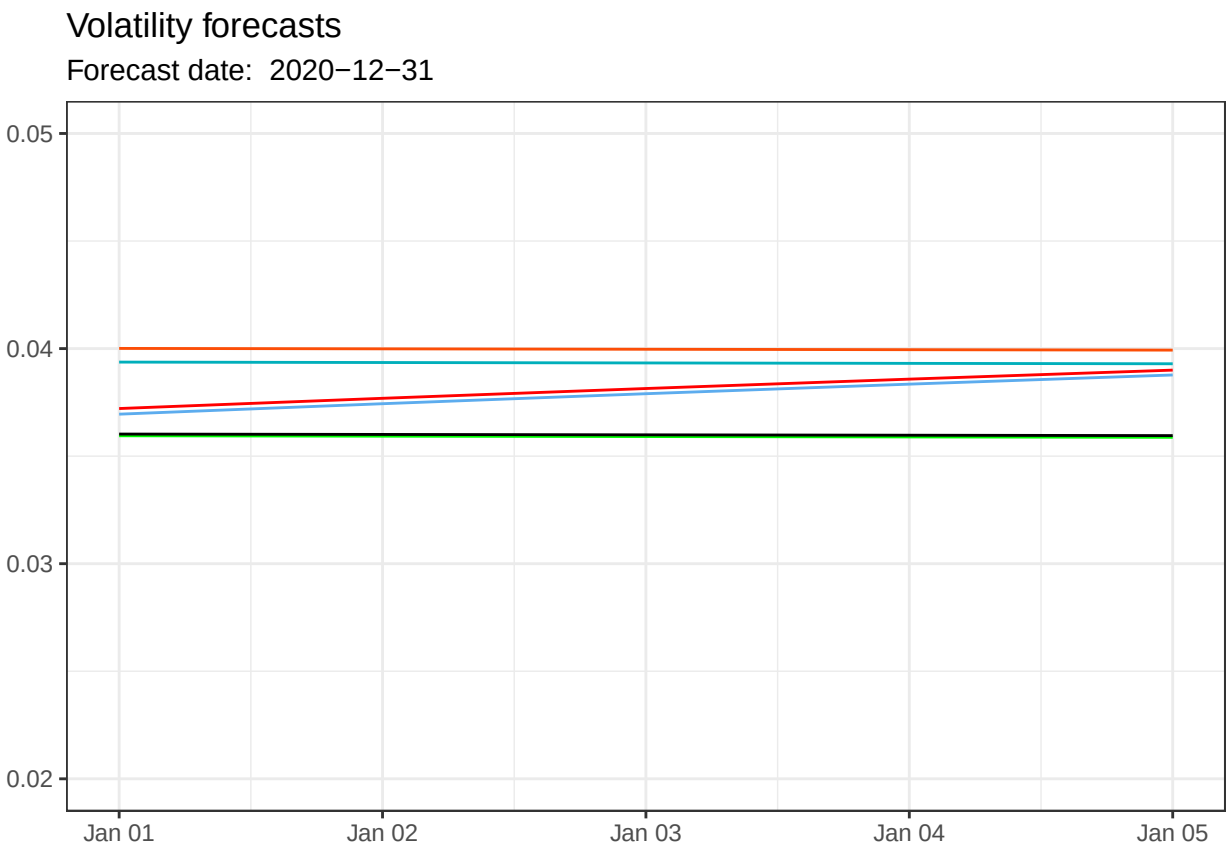
```
## Using column 'time' for date_var.
```

```
nforecast=5

f_fits_1 <- ugarchforecast(fits[[1]], n.ahead = nforecast)
f_fits_2 <- ugarchforecast(fits[[2]], n.ahead = nforecast)
f_fit_d_1 <- ugarchforecast(fit_d[[1]], n.ahead = nforecast)
f_fit_d_2 <- ugarchforecast(fit_d[[2]], n.ahead = nforecast)
f_fit_gjr_1 <- ugarchforecast(fit_d_gjr[[1]], n.ahead = nforecast)
f_fit_gjr_2 <- ugarchforecast(fit_d_gjr[[2]], n.ahead = nforecast)


temp <- data.frame(Date = end(btc_plot) + 1:nforecast,
                   arma_garch_norm = as.numeric(sigma(f_fits_1)),
                   garch_norm = as.numeric(sigma(f_fits_2)),
                   garch_std = as.numeric(sigma(f_fit_d_1)),
                   garch_ged = as.numeric(sigma(f_fit_d_2)),
                   gjr_garch_std = as.numeric(sigma(f_fit_gjr_1)),
                   gjr_garch_ged = as.numeric(sigma(f_fit_gjr_2))
                   )
ggplot(temp) + geom_line(aes(Date, arma_garch_norm), color="red") +
  geom_line(aes(Date, garch_norm), color="steelblue2") +
  geom_line(aes(Date, garch_std), color="green")+
  geom_line(aes(Date, garch_ged), color="black")+
  geom_line(aes(Date, gjr_garch_std), color="#FC4E07")+
  geom_line(aes(Date, gjr_garch_ged), color="#00AFBB")+

  geom_hline(yintercept = sd(btc_plot), color="seagreen4", linetype="dashed") +
  labs(x=NULL, y=NULL, title="Volatility forecasts",
       subtitle=paste("Forecast date: ", end(btc_plot))) + theme_bw()+ ylim(.02, .05)+theme(legend.posi
```

```
## Warning: Removed 1 rows containing missing values (geom_hline).
```

Table 6: Forecast BTCUSD 5 steps ahead

Date	arma_garch_norm	garch_norm	garch_std	garch_ged	gjr_garch_std	gjr_garch_ged
2021-01-01	0.0372133	0.0369517	0.0359469	0.0360265	0.0400083	0.0393723
2021-01-02	0.0376889	0.0374372	0.0359289	0.0360085	0.0399883	0.0393526
2021-01-03	0.0381443	0.0379017	0.0359110	0.0359904	0.0399683	0.0393329
2021-01-04	0.0385805	0.0383466	0.0358930	0.0359724	0.0399483	0.0393132
2021-01-05	0.0389986	0.0387730	0.0358751	0.0359544	0.0399283	0.0392936




```
kable(temp, caption='Forecast BTCUSD 5 steps ahead')%>%
kable_styling(full_width = F, font_size = 8)
```

Conclusions

In conclusion, after studying the bitcoin timeline we can observe that news about the performance of blockchain and the reliability of bitcoin as commodity or investor's feelings affect the volatility of the crypto currency over time. The model with the best performance was GRJ-GARCH(1,1) with a t-student distribution which captures the shocks provoked for microstructure of the BTCUSD market. Further investigations could include external regressors, for example, the halving day or a proxy variable that captures the investor's feelings as tweets could improve the forecast of the volatility.

Annex

Table 7: ARMA(0,0)-GARCH(1,1) std

	Estimate	Std. Error	t value	Pr(>|t|)
mu	0.0018202	0.0004205	4.328391	1.5e-05
omega	0.0000000	NA	NA	NA
alpha1	0.0839782	0.0096740	8.680828	0.0e+00
beta1	0.9150218	0.0090887	100.676332	0.0e+00
shape	3.3959169	0.1695336	20.030940	0.0e+00

Table 8: ARMA(0,0)-GARCH(1,1) ged

	Estimate	Std. Error	t value	Pr(>|t|)
mu	0.0017373	0.0001456	11.936030	0
omega	0.0000000	NA	NA	NA
alpha1	0.0733925	0.0094230	7.788655	0
beta1	0.9256064	0.0086936	106.470229	0
shape	0.8567981	0.0263013	32.576317	0

1. COEFFICIENTS GARCH MODELS

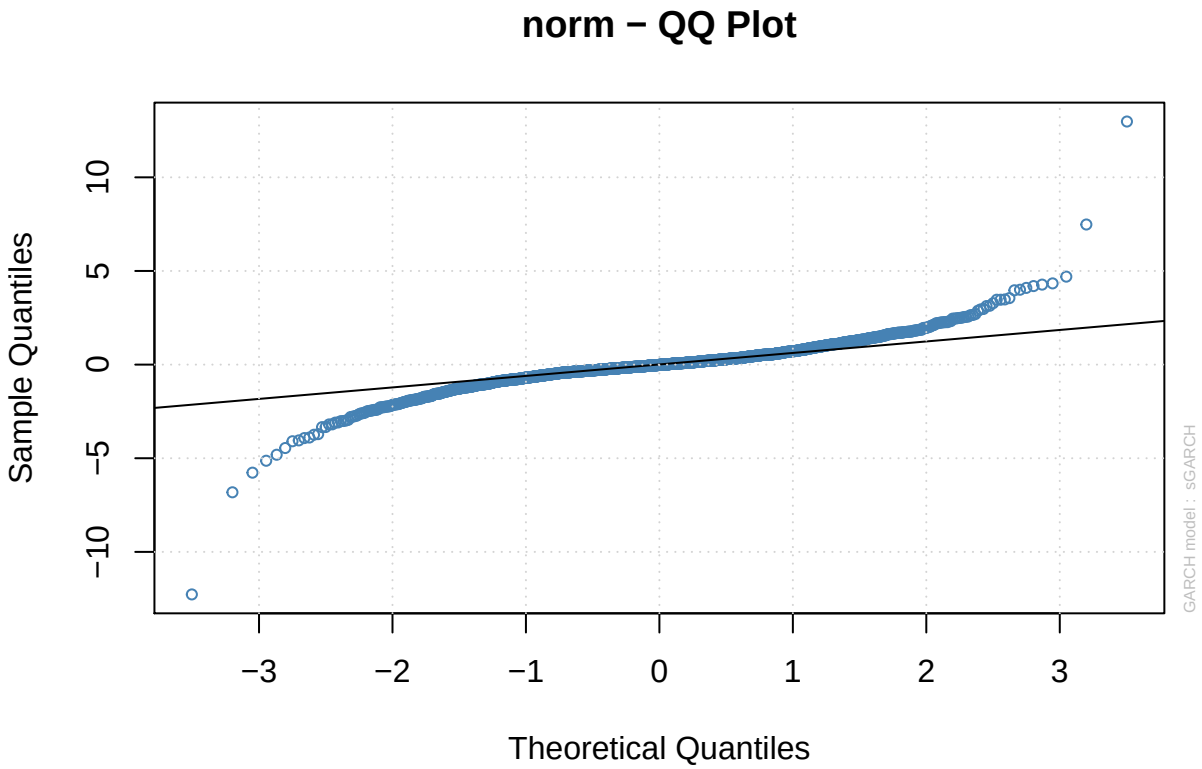
Table 9: ARMA(0,0)-GJR-GARCH(1,1) std

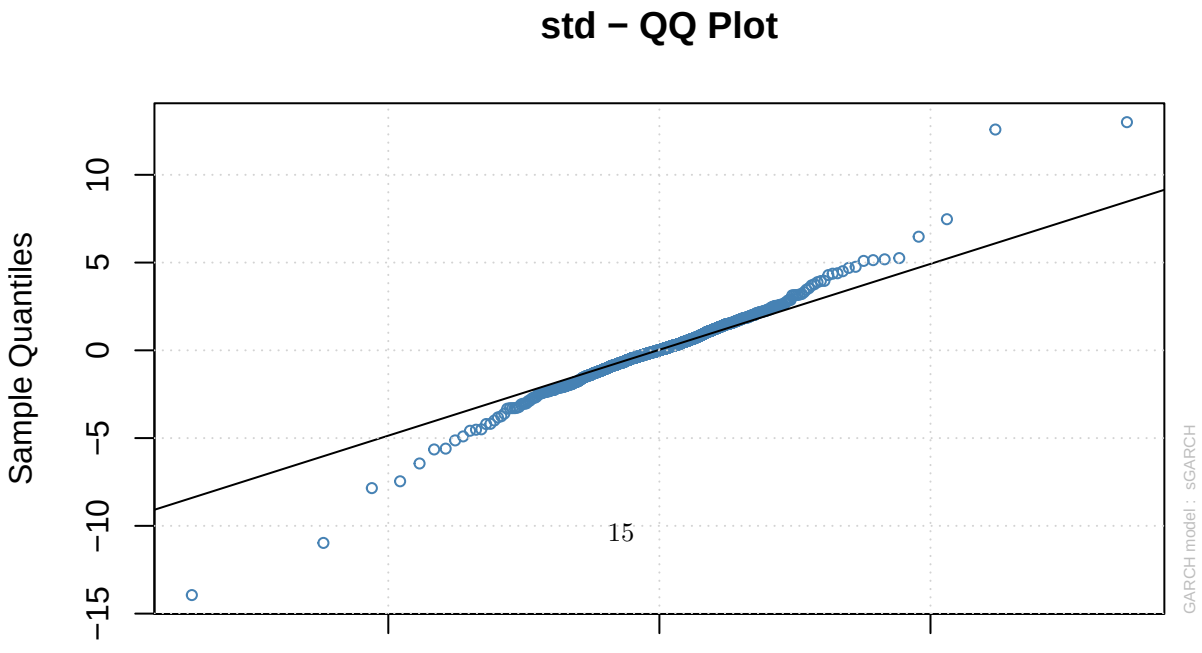
	Estimate	Std. Error	t value	Pr(>|t|)
mu	0.0018202	0.0004205	4.328391	1.5e-05
omega	0.0000000	NA	NA	NA
alpha1	0.0839782	0.0096740	8.680828	0.0e+00
beta1	0.9150218	0.0090887	100.676332	0.0e+00
shape	3.3959169	0.1695336	20.030940	0.0e+00

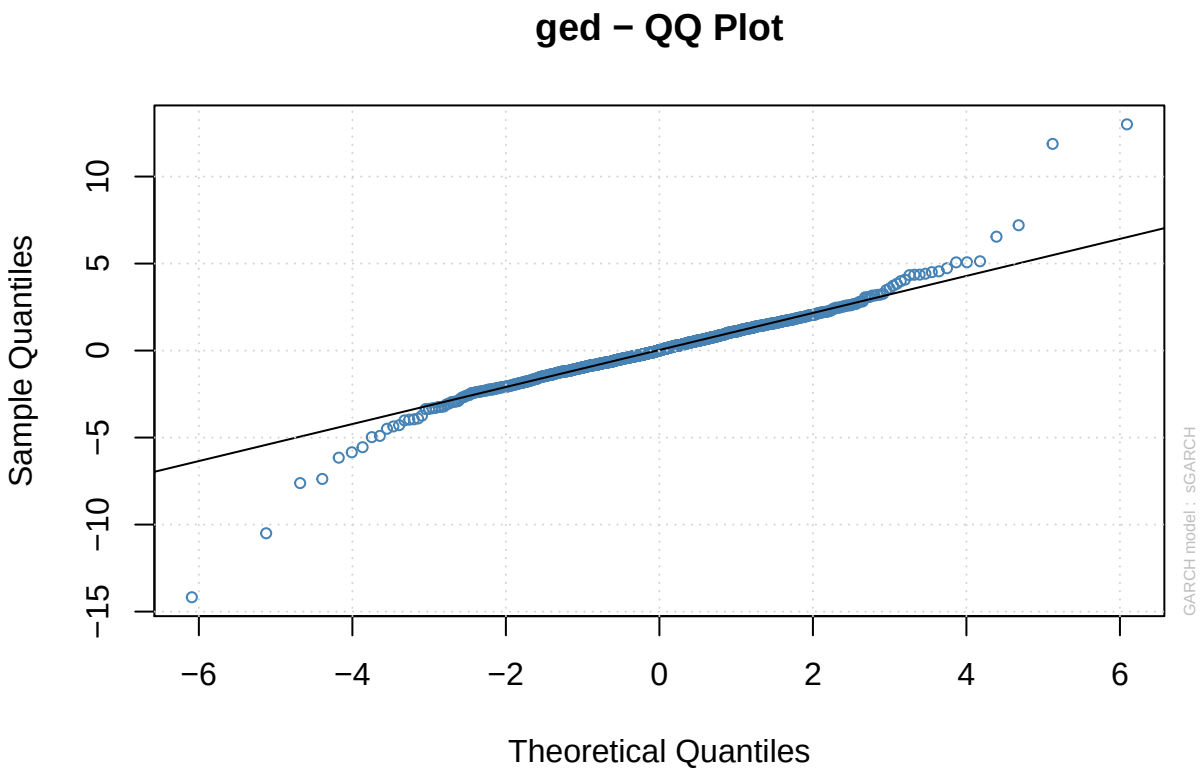
Table 10: ARMA(0,0)-GJR-GARCH(1,1) ged

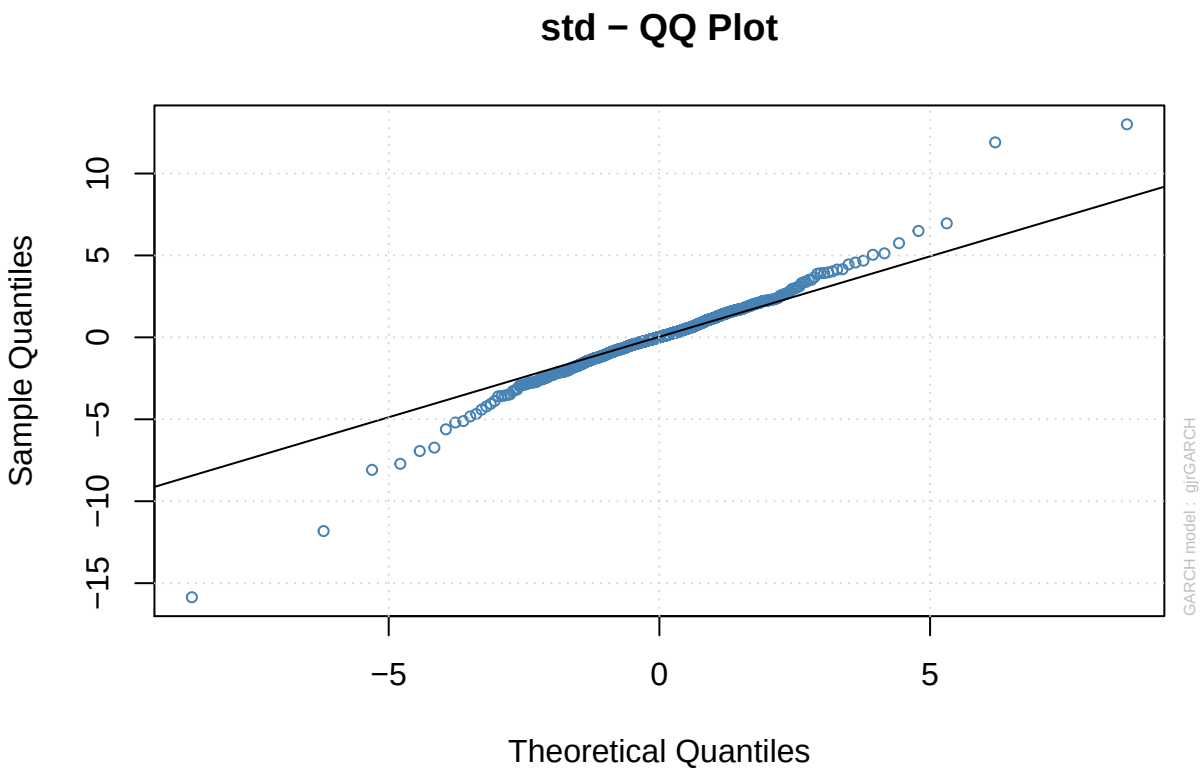
	Estimate	Std. Error	t value	Pr(>|t|)
mu	0.0018202	0.0004205	4.328391	1.5e-05
omega	0.0000000	NA	NA	NA
alpha1	0.0839782	0.0096740	8.680828	0.0e+00
beta1	0.9150218	0.0090887	100.676332	0.0e+00
shape	3.3959169	0.1695336	20.030940	0.0e+00

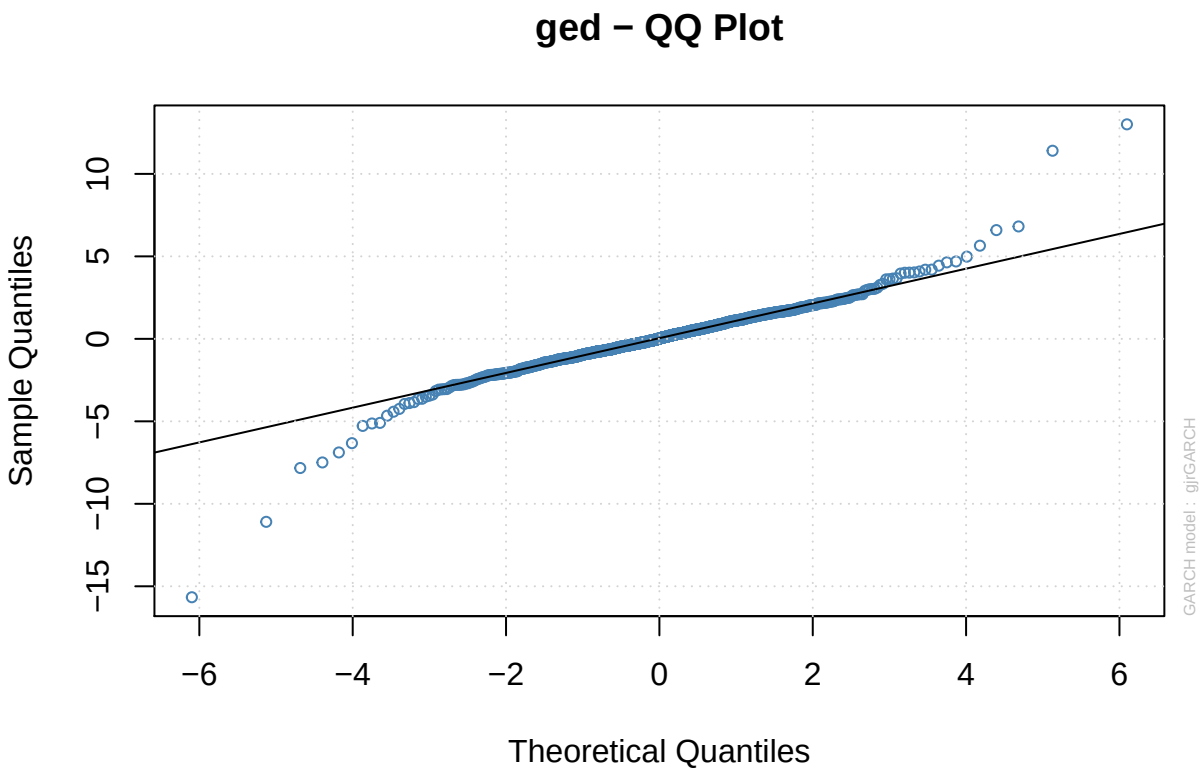
2.QQ PLOTS






ged − QQ Plot
Sample Quantiles
10
5
0
−5
−10
−15
−6
−4
−2
0
2
4
6
Theoretical Quantiles
GARCH model : sGARCH


std − QQ Plot
Sample Quantiles
10
5
0
−5
−10
−15
−5
0
5
Theoretical Quantiles
GARCH model : gjrGARCH



3. Mean Square Error

```
#MSE
e1<-mean((fits[[1]]@fit$residuals)^2)
e2<-mean((fits[[2]]@fit$residuals)^2)
e3<-mean((fit_d[[1]]@fit$residuals)^2)
e4<-mean((fit_d[[2]]@fit$residuals)^2)
e5<-mean((fit_d_gjr[[1]]@fit$residuals)^2)
e6<-mean((fit_d_gjr[[2]]@fit$residuals)^2)

d1<-((fits[[1]]@fit$residuals)^2)-(fits[[1]]@fit$sigma)^2
d1<-mean(d1^2)
d2<-((fits[[2]]@fit$residuals)^2)-(fits[[2]]@fit$sigma)^2
d2<-mean(d2^2)
d3<-((fit_d[[1]]@fit$residuals)^2)-(fit_d[[1]]@fit$sigma)^2
d3<-mean(d3^2)
d4<-((fit_d[[2]]@fit$residuals)^2)-(fit_d[[2]]@fit$sigma)^2
d4<-mean(d4^2)
d5<-((fit_d_gjr[[1]]@fit$residuals)^2)-(fit_d_gjr[[1]]@fit$sigma)^2
d5<-mean(d5^2)
d6<-((fit_d_gjr[[2]]@fit$residuals)^2)-(fit_d_gjr[[2]]@fit$sigma)^2
d6<-mean(d6^2)

table_errors<-matrix(c(e1,e2,e3,e4,e5,e6,
```

Table 11: Mean Squared Errors mu and sigma

	ARMA(1,0)-GARCH(1,1) norm	ARMA(0,0)-GARCH(1,1) norm	ARMA(0,0)-GARCH(1,1) std	ARM
mean_MSE	0.0016562	0.0016615	0.0016619	
var_MSE	0.0000787	0.0000784	0.0000769	

```
                      d1,d2,d3,d4,d5,d6),nrow=2,ncol=6,byrow = TRUE,
                    dimnames=list(
                      c("mean_MSE","var_MSE"),
                      c("ARMA(1,0)-GARCH(1,1) norm",
                                    "ARMA(0,0)-GARCH(1,1) norm",
                                    "ARMA(0,0)-GARCH(1,1) std",
                                    "ARMA(0,0)-GARCH(1,1) ged",
                                    "ARMA(0,0)-GJR-GARCH(1,1) std",
                                    "ARMA(0,0)-GJR-GARCH(1,1) ged"
                    )))

kable(table_errors,caption='Mean Squared Errors mu and sigma')%>%
  kable_styling()
```