# Dynamic Correlations of Volatilities in Crude Oil Prices

*© Skye Zhang 2020. All rights reserved.*

*June 03, 2020*

## 1 Introduction

Over the past decades, crude oil has become the biggest traded commodity in the world. In the crude oil market, oil is sold under a variety of contracts and in spot transactions. Crude oil is usually sold close to the point of production, and is transferred as the oil flows from the loading terminal to the ship FOB (free on board). Thus, spot prices are quoted for immediate delivery of crude oil as FOB prices.

Prior to the 21st century, there are three major benchmarks in the global oil market: (1) West Texas Intermediate (WTI), the reference crude for USA; (2) Brent, the reference crude oil for the North Sea; (3) Dubai, the benchmark crude oil for the Middle East and Far East. In recent years, the East Siberian Pacific Ocean (ESPO) price, the reference crude for Russia, has become an emerging benchmark for the Asia-Pacific region. Although crude oil is traded independently in these markets, prices of different markets are tightly correlated, affected by geopolitical and economic policies.

As spot prices change dynamically, their volatility is typically unobservable, yet volatility spillovers appear to be widespread in financial markets. Studies show that the spillover effects hold even when markets do not necessarily trade at the same time. Consequently, a volatility spillover occurs when changes in volatility in one market produce a lagged impact on volatility in other markets, over and above local effects. Volatility spillovers and asymmetries among those four major benchmarks are likely to be important for not only constructing hedge ratios and optimal portfolios, but also supporting policy makers with rigorous econometric suggestions.

## 2 Methodology

### 2.1 Logarithmic Return & Volatility

Logarithmic return is one of the three methods for calculating return $r_i$ of an asset $(i)$ with price $P_t$ at time $t$. It assumes returns are compounded continuously that equal to the natural logarithm of closing price at $t$ less the natural logarithm of the closing price at the beginning of the period, which is defined as

$$r_{i,t} = ln(\frac{P_t}{P_{t-1}}).$$

In finance, volatility $(\sigma)$ is the degree of variation of a trading price series over time, usually measured by the standard deviation of logarithmic returns. Hence, by this definition, volatility can be effectively analyzed by GARCH models.

### 2.2 Time Series Models

The analysis consists of three sections: stationarity check, autocorrelation test, and (E)DCC-MGARCH modeling.

#### 2.2.1 Stationarity

As financial series are sensitive to economic cycle and other factors over time, many price processes are often not stationary by nature. A common way to address the non-stationary series is to take the first order difference of the series. The logarithmic return is essentially the first order difference of the natural logarithm of the price. Thus, one may perform tests directly on the log-return series $(r_{i,t})$ to check stationarity.

In this study, both the ADF test and the KPSS test are performed, because their null hypotheses are opposite to each other. The null hypothesis of an ADF test is that there exists a unit root so that the series is non-stationary, whereas the null hypothesis of a KPSS test is that the series is stationary. If the null hypothesis of the ADF test is rejected and the null hypothesis of the KPSS test cannot be rejected, the series is stationary.

#### 2.2.2 Autocorrelation & ARCH Effect

Essentially, a time series process consists of a mean process and a variance process. Before I dive into the variance, it is necessary to examine whether the mean process is constant. Hence, the Ljung-Box test is conducted to the check the autocorrelation. If the test results are significant, $ARMA(p,q)$ models could be exploited to estimate the mean part.

After autoregressive modeling, I take a closer look at the residuals and the squared residuals of the return series to test for ARCH effect. The Engle's ARCH test is a Lagrange multiplier test to assess the significance of ARCH effects. The null hypothesis is that there is no ARCH effect, meaning a dynamic conditional variance process, in the series. If the hypothesis is rejected, the one may use GARCH to model the variance part.

#### 2.2.3 Multivariate GARCH Models

In order to analysis the volatility spillover effects in major oil markets, that is assessing the effect of the volatility of one oil market on another oil market, the multivariate GARCH models are needed. MGARCH allows the conditional-on-past-history covariance matrix of the dependent variables to follow a flexible dynamic structure. Conditional correlation models use nonlinear combinations of univariate GARCH models to represent the conditional covariances.

**(1). Constant Conditional Correlation MGARCH**

Consider a stochastic process of a $N \times 1$ vector $\{\boldsymbol{r}_t\}$: $\boldsymbol{r}_t = \boldsymbol{\mu}_t + \boldsymbol{\varepsilon}_t$, where $\boldsymbol{\mu}_t$ is a vector of the expected value of the conditional $\boldsymbol{r}_t$ and $\boldsymbol{\varepsilon}_t$ is a multivariate mean-corrected returns of $n$ assets at time $t$.

$$\boldsymbol{\varepsilon}_t = \boldsymbol{H}_t^{1/2}\boldsymbol{z}_t,$$

where $\boldsymbol{H}_t^{1/2}$ is a positive-definite matrix of $N \times N$ dimensions, $\boldsymbol{z}_t$ is an $N \times 1$ multivariate random variable. According to Bollerslev (1990), assuming at time $t$, the correlation of any two components, $r_{i,t}$ and $r_{j,t}$ (two separate series), in $\{\boldsymbol{r}_t\}$ is a constant $\rho_{i,j}$, that is:

$$\rho_{i,j,t} = \frac{h_{i,j,t}}{\sqrt{h_{ii,t}h_{jj,t}}} = \rho_{i,j},$$

where $h_{i,j,t}$ is the element $(i,j)$ of the covariance matrix $\boldsymbol{H}_t$, which denotes the conditional covariance of $r_{i,t}$ and $r_{j,t}$ at time $t$. Variable $h_{ii,t}$ and $h_{jj,t}$ are the conditional variance of $r_{i,t}$ and $r_{j,t}$ at time $t$. Under the assumption that the correlations are constant, $\boldsymbol{H}_t$ can be rewritten as

$$\boldsymbol{H}_t = \boldsymbol{D}_t\boldsymbol{R}\boldsymbol{D}_t = (\rho_{i,j}\sqrt{h_{ii,t}h_{jj,t}})_{i,j},$$

where

$$\boldsymbol{R} = \begin{pmatrix} \rho_{11} & \cdots & \rho_{1N} \\ \vdots & \ddots & \vdots \\ \rho_{N1} & \cdots & \rho_{NN} \end{pmatrix},$$

$$\boldsymbol{D}_t = diag(h_{ii,t}^{1/2} \cdots h_{NN,t}^{1/2}),$$

where $h_{ii,t}$ can be estimated by any univariate $GARCH(p,q)$ model, such as

$$h_{ii,t} = \omega_i + \sum_{p=1}^{P_i} \alpha_{ip}\varepsilon_{i(t-p)} + \sum_{q=1}^{Q_j} \beta_{iq}h_{i(t-q)}.$$

The advantage of using this model comes from its assumption. The number of parameters to be estimated are relatively low. However, in practice, the assumption of constant conditional correlation is often violated, in that the correlation between two markets could vary over time (conditionally). As a result, Engle(2002) proposed an improvement by introducing the Dynamic Conditional Correlation model.

**(2). (Extended) Dynamic Conditional Correlation MGARCH**

The improvement Engle made lies on the correlation matrix $\boldsymbol{R}_t$. Instead, it is defined as

$$\boldsymbol{R}_t = diag(q_{11,t}^{-1/2} \cdots q_{NN,t}^{-1/2})\boldsymbol{Q}_t diag(q_{11,t}^{-1/2} \cdots q_{NN,t}^{-1/2}),$$

where

$$diag(q_{11,t}^{-1/2} \cdots q_{NN,t}^{-1/2}) = \begin{pmatrix} q_{11,t}^{-1/2} & 0 & \cdots & 0 \\ 0 & q_{22,t}^{-1/2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & q_{NN,t}^{-1/2} \end{pmatrix} = \boldsymbol{Q}_t^{*-1}.$$

Matrix $\boldsymbol{Q}_t = (q_{ij,t})$ has to be positive definite to ensure $\boldsymbol{R}_t$ is positive definite and can be generated by the following equation:

$$\boldsymbol{Q}_t = (1 - \alpha - \beta)\overline{\boldsymbol{Q}} + \alpha \boldsymbol{u}_{t-1}\boldsymbol{u}_{t-1}^T + \beta \boldsymbol{Q}_{t-1},$$

where $\overline{\boldsymbol{Q}} = Cov[\boldsymbol{u}_t \boldsymbol{u}_t^T] = E[\boldsymbol{u}_t \boldsymbol{u}_t^T]$ is the unconditional covariance matrix of the standardized errors

$$u_{i,t} = \frac{\varepsilon_{i,t}}{\sqrt{h_{ii,t}}}.$$

The parameters $\alpha$ and $\beta$ are scalars, and must satisfy $\alpha \geqslant 0$, $\beta \geqslant 0$, and $\alpha + \beta < 1$. In this research, only the $DCC(1,1) - MGARCH$ model will be used. The parameters in the above models can be determined by maximum log-likelihood performed by R packages.

## 3 R Code & Results

### 3.1 Import Libraries.

```
library(fpp)
library(TSA)
library(xts)
library(FinTS)
library(rugarch)
library(rmgarch)
library(quantmod)
library(DataCombine)
library(PerformanceAnalytics)
```

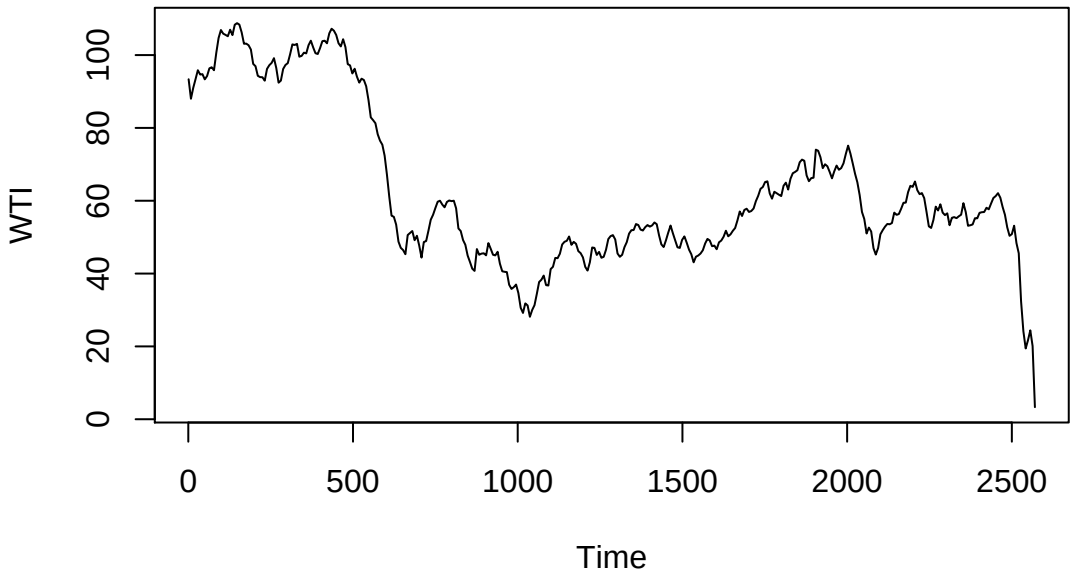
### 3.2 Load weekly price data of the major oil markets, WTI, BRENT, DUBAI, and ESPO.

```
crude_oil = "/Users/tiavas/Dropbox/MSCA/Time\ Series/Project/Oil\ Price\ weekly.csv"
data <- read.csv(crude_oil)
data$Date = as.Date(data$Date, "%Y-%m-%d")
```

### 3.3 Visualizations

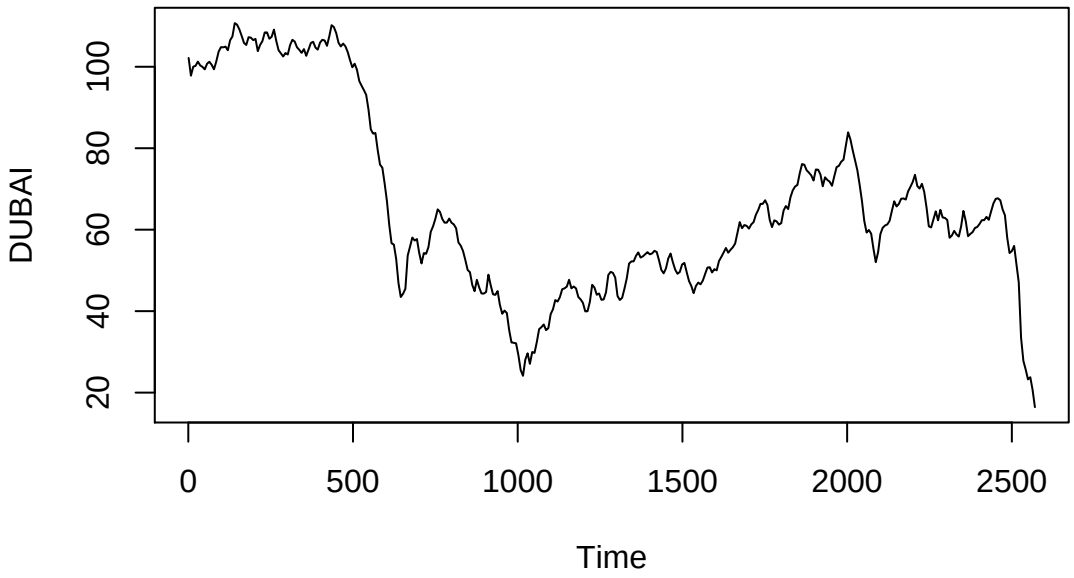
#### 3.3.1 Plot the price series of individual oil markets.

```
WTI <- xts(data$WTI, data$Date)
ts.plot(WTI)
```

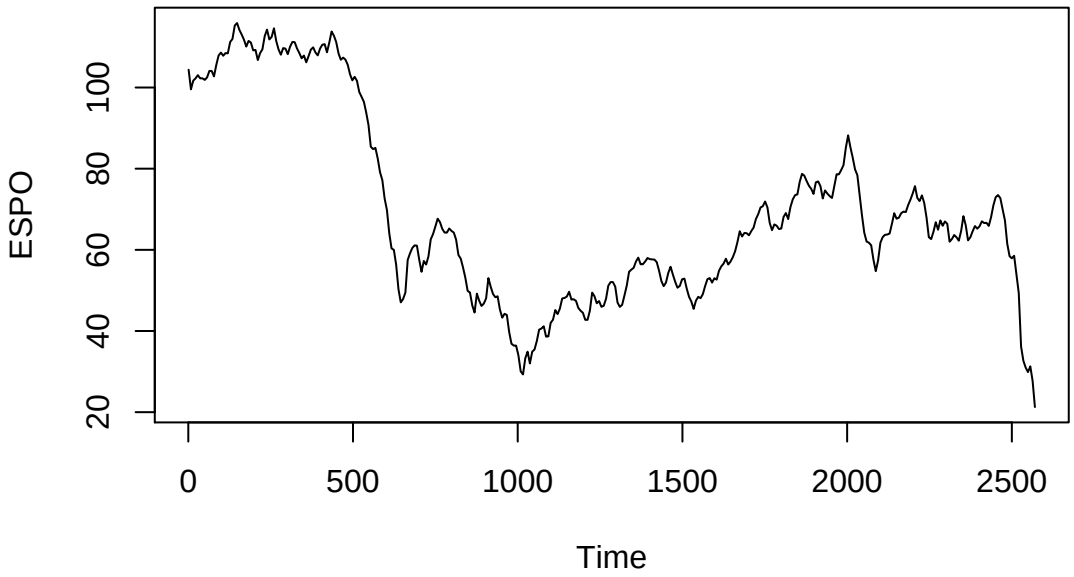



```
BRENT <- xts(data$BRENT, data$Date)
ts.plot(BRENT)
```


```
DUBAI <- xts(data$DUBAI, data$Date)
ts.plot(DUBAI)
```

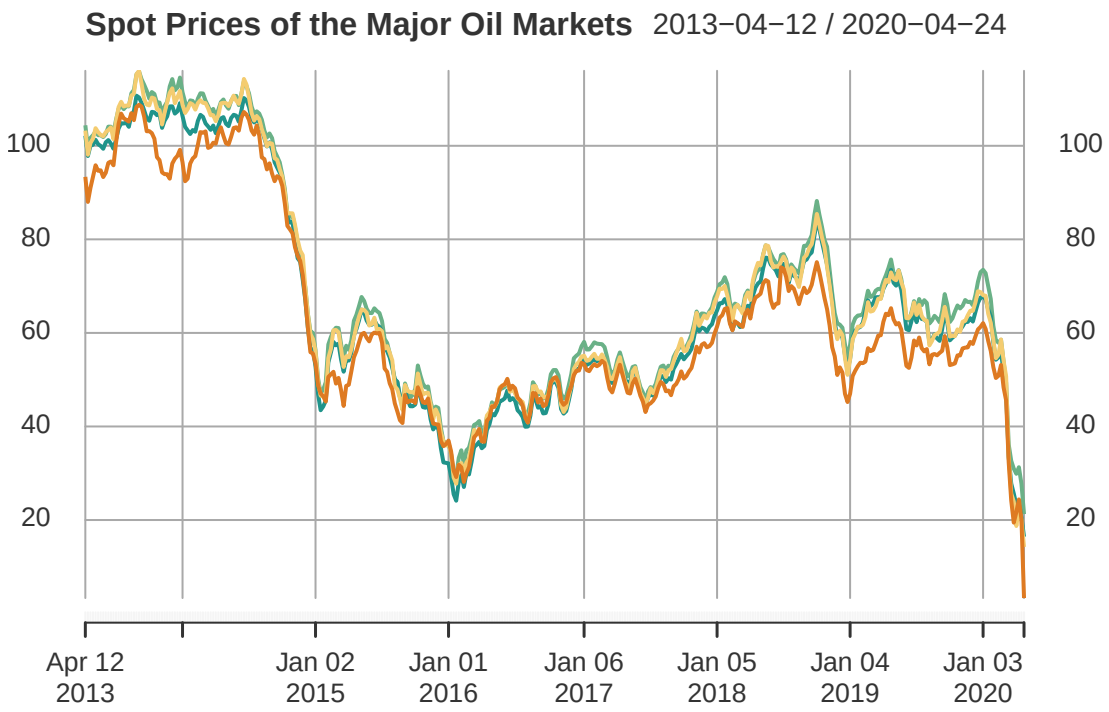



```
ESPO <- xts(data$ESPO, data$Date)
ts.plot(ESPO)
```




**3.3.2 Plot the integrated the price series of all major markets.**

```
MARKET <- xts(cbind(WTI, BRENT, DUBAI, ESPO), index(WTI))
par(mfrow=c(1,1))
plot(
  MARKET,
  main="Spot Prices of the Major Oil Markets",
  major.ticks="years", minor.ticks=NULL,
  grid.ticks.on="years",
  col=c("#DE7A22", "#F4CC70", "#20948B", "#6AB187")
)
```

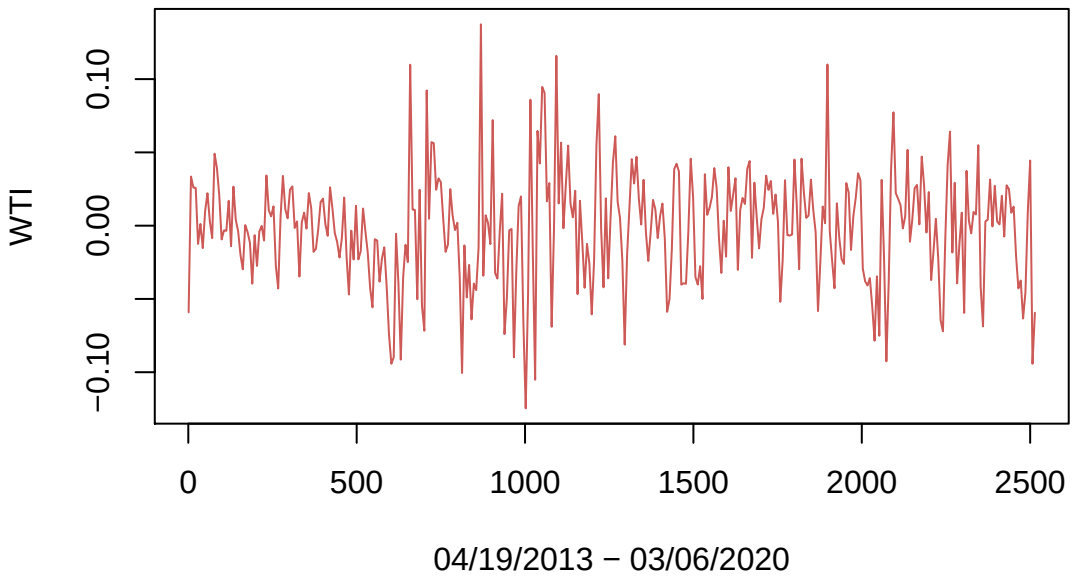



**3.4 Calcuate the log-return of all major markets and test for stationarity.**

```
market <- log(MARKET)

logreturn <- diff(market)[-1, ]
return.train = logreturn[c(1:360),]
```

**3.4.1 WTI**

```
ts.plot(return.train$WTI,
        xlab="04/19/2013 - 03/06/2020",
        ylab="WTI", col="#CE5A57")
```



```
adf.test(return.train$WTI)
```

```
## Warning in adf.test(return.train$WTI): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  return.train$WTI
## Dickey-Fuller = -5.5083, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```
kpss.test(return.train$WTI)
```

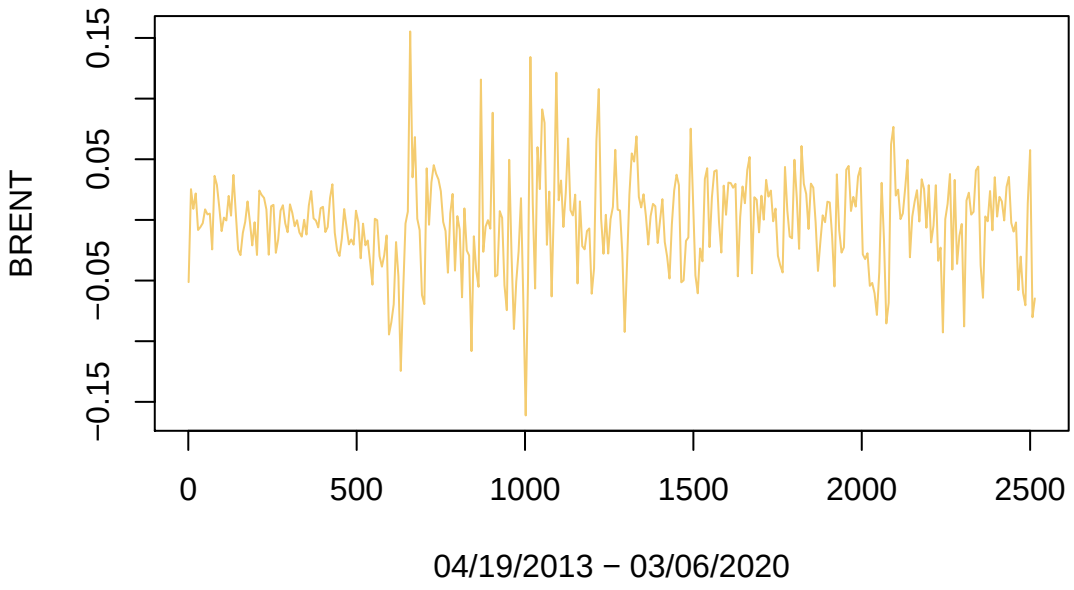
```
## Warning in kpss.test(return.train$WTI): p-value greater than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  return.train$WTI
## KPSS Level = 0.10701, Truncation lag parameter = 5, p-value = 0.1
```

Through both the qualitative and the quantitative methods, the WTI log-return series is stationary.

#### 3.4.2 BRENT

```
ts.plot(return.train$BRENT,
        xlab="04/19/2013 - 03/06/2020",
        ylab="BRENT",col="#F4CC70")
```




```
adf.test(return.train$BRENT)
```

```
## Warning in adf.test(return.train$BRENT): p-value smaller than printed p-
## value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  return.train$BRENT
```

```
## Dickey-Fuller = -6.0029, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```
kpss.test(return.train$BRENT)
```

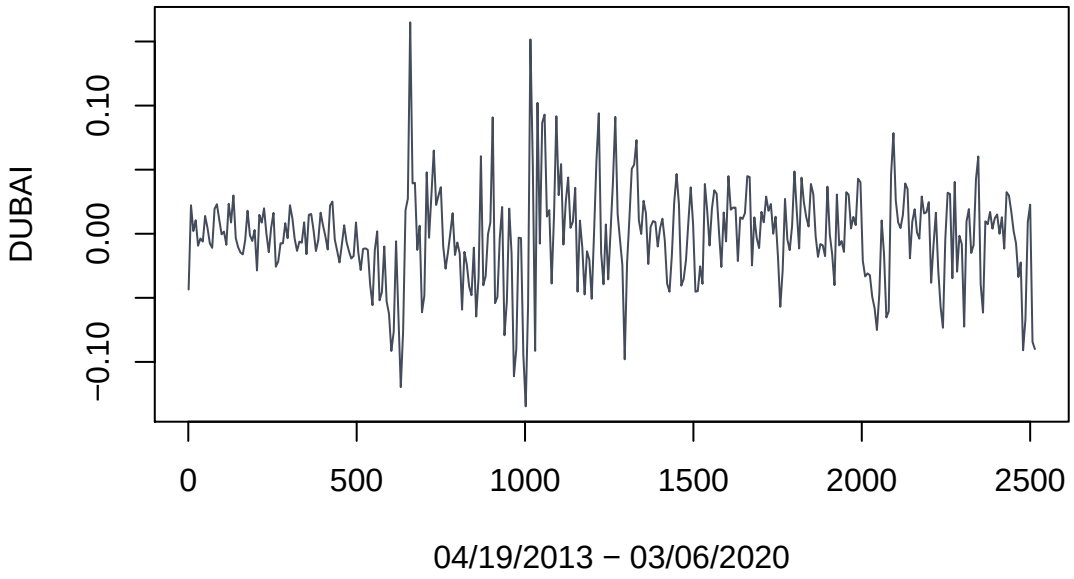
```
## Warning in kpss.test(return.train$BRENT): p-value greater than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  return.train$BRENT
## KPSS Level = 0.12681, Truncation lag parameter = 5, p-value = 0.1
```

Through both the qualitative and the quantitative methods, the BRENT log-return series is stationary.

#### 3.4.3 DUBAI

```
ts.plot(return.train$DUBAI,
        xlab="04/19/2013 - 03/06/2020",
        ylab="DUBAI",col="#444C5C")
```




```
adf.test(return.train$DUBAI)
```

```
## Warning in adf.test(return.train$DUBAI): p-value smaller than printed p-
## value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  return.train$DUBAI
## Dickey-Fuller = -5.2874, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```
kpss.test(return.train$DUBAI)
```

```
## Warning in kpss.test(return.train$DUBAI): p-value greater than printed p-
## value
```

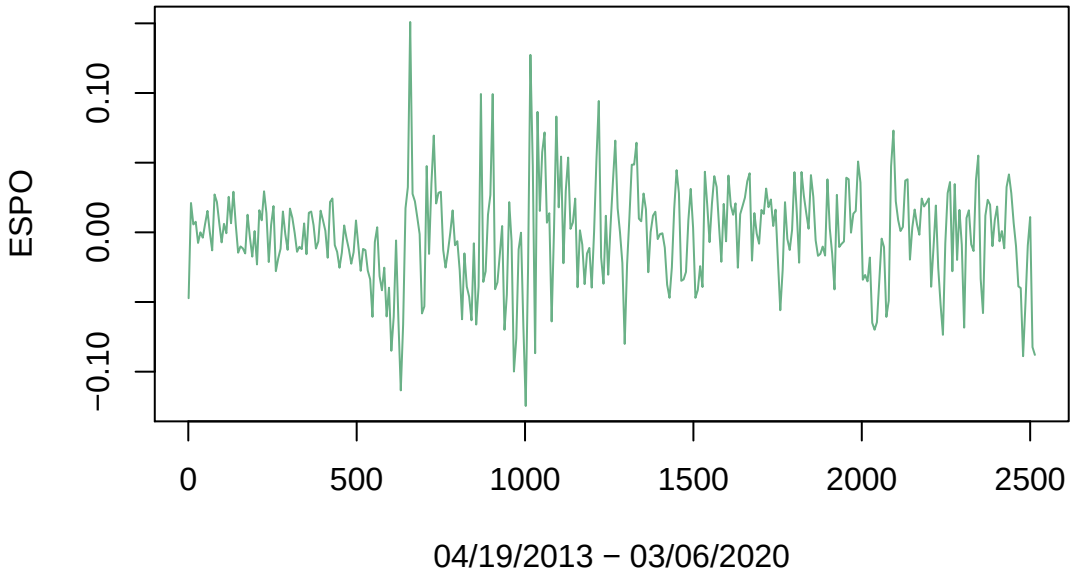
```
##
```

```
##  KPSS Test for Level Stationarity
##
## data:  return.train$DUBAI
## KPSS Level = 0.12726, Truncation lag parameter = 5, p-value = 0.1
```

Through both the qualitative and the quantitative methods, the DUBAI log-return series is stationary.

#### 3.4.3 ESPO

```
ts.plot(return.train$ESPO,
        xlab="04/19/2013 - 03/06/2020",
        ylab="ESPO", col="#6AB187")
```




```
adf.test(return.train$ESPO)
```

```
## Warning in adf.test(return.train$ESPO): p-value smaller than printed p-
## value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  return.train$ESPO
## Dickey-Fuller = -5.6229, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```
kpss.test(return.train$ESPO)
```

```
## Warning in kpss.test(return.train$ESPO): p-value greater than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  return.train$ESPO
## KPSS Level = 0.1253, Truncation lag parameter = 5, p-value = 0.1
```

Through both the qualitative and the quantitative methods, the ESPO log-return series is stationary.

**3.5 Ljung-Box test on the mean process for autocorrelation.**

I combined the four tests together.

```
for(i in 1:4) print(Box.test(return.train$WTI, lag = 5*i, type =  "Ljung-Box"))
```

```
##
##  Box-Ljung test
##
## data:  return.train$WTI
## X-squared = 32.751, df = 5, p-value = 4.217e-06
##
##
##  Box-Ljung test
##
## data:  return.train$WTI
## X-squared = 34.41, df = 10, p-value = 0.0001573
##
##
##  Box-Ljung test
##
## data:  return.train$WTI
## X-squared = 45.022, df = 15, p-value = 7.596e-05
##
##
##  Box-Ljung test
##
## data:  return.train$WTI
## X-squared = 54.762, df = 20, p-value = 4.456e-05
```

```
for(i in 1:4) print(Box.test(return.train$BRENT, lag = 5*i, type =  "Ljung-Box"))
```

```
##
##  Box-Ljung test
##
## data:  return.train$BRENT
## X-squared = 32.43, df = 5, p-value = 4.882e-06
##
##
##  Box-Ljung test
##
## data:  return.train$BRENT
## X-squared = 32.865, df = 10, p-value = 0.000287
##
##
##  Box-Ljung test
##
## data:  return.train$BRENT
## X-squared = 38.053, df = 15, p-value = 0.0008861
##
##
##  Box-Ljung test
##
## data:  return.train$BRENT
## X-squared = 42.119, df = 20, p-value = 0.002668
```

```r
for(i in 1:4)print(Box.test(return.train$DUBAI, lag = 5*i, type =  "Ljung-Box"))
```

```
##
##  Box-Ljung test
##
## data:  return.train$DUBAI
## X-squared = 53.885, df = 5, p-value = 2.213e-10
##
##
##  Box-Ljung test
##
## data:  return.train$DUBAI
## X-squared = 56.322, df = 10, p-value = 1.786e-08
##
##
##  Box-Ljung test
##
## data:  return.train$DUBAI
## X-squared = 68.182, df = 15, p-value = 9.398e-09
##
##
##  Box-Ljung test
##
## data:  return.train$DUBAI
## X-squared = 74.787, df = 20, p-value = 2.956e-08
```

```r
for(i in 1:4) print(Box.test(return.train$ESPO, lag = 5*i, type =  "Ljung-Box"))
```

```
##
##  Box-Ljung test
##
## data:  return.train$ESPO
## X-squared = 52.786, df = 5, p-value = 3.721e-10
##
##
##  Box-Ljung test
##
## data:  return.train$ESPO
## X-squared = 55.427, df = 10, p-value = 2.628e-08
##
##
##  Box-Ljung test
##
## data:  return.train$ESPO
## X-squared = 66.033, df = 15, p-value = 2.25e-08
##
##
##  Box-Ljung test
##
## data:  return.train$ESPO
## X-squared = 69.961, df = 20, p-value = 1.848e-07
```

The test results indicate that all four series have autocorrelations. I use $ARMA(p, q)$ models the filter the mean process and examine the residuals.

### 3.6 Mean Process Filtering by $ARMA(p, q)$

- **WTI**

```
wti_arma <- auto.arima(return.train$WTI, stationary = T, seasonal = F, ic = "aic", allowdrift = FALSE, t
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,0,2) with non-zero mean : -1366.586
##  ARIMA(0,0,0) with non-zero mean : -1333.404
##  ARIMA(1,0,0) with non-zero mean : -1361.595
##  ARIMA(0,0,1) with non-zero mean : -1362.08
##  ARIMA(0,0,0) with zero mean     : -1334.403
##  ARIMA(1,0,2) with non-zero mean : -1366.417
##  ARIMA(2,0,1) with non-zero mean : -1368.464
##  ARIMA(1,0,1) with non-zero mean : -1363.331
##  ARIMA(2,0,0) with non-zero mean : -1361.998
##  ARIMA(3,0,1) with non-zero mean : -1361.894
##  ARIMA(3,0,0) with non-zero mean : -1360.025
##  ARIMA(3,0,2) with non-zero mean : -1367.245
##  ARIMA(2,0,1) with zero mean     : -1370.141
##  ARIMA(1,0,1) with zero mean     : -1364.679
##  ARIMA(2,0,0) with zero mean     : -1363.355
##  ARIMA(3,0,1) with zero mean     : -1363.602
##  ARIMA(2,0,2) with zero mean     : -1368.226
##  ARIMA(1,0,0) with zero mean     : -1363.115
##  ARIMA(1,0,2) with zero mean     : -1367.384
##  ARIMA(3,0,0) with zero mean     : -1361.409
##  ARIMA(3,0,2) with zero mean     : -1368.526
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(2,0,1) with zero mean     : -1367.945
##
##  Best model: ARIMA(2,0,1) with zero mean
```

The WTI log-return series need an $ARMA(2, 1)$ model to filter the mean process.

- **BRENT**

```
brent_arma <- auto.arima(return.train$BRENT, stationary = T, seasonal = F, ic = "aic", allowdrift = FALS
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,0,2) with non-zero mean : -1342.712
##  ARIMA(0,0,0) with non-zero mean : -1308.977
##  ARIMA(1,0,0) with non-zero mean : -1335.72
##  ARIMA(0,0,1) with non-zero mean : -1340.03
##  ARIMA(0,0,0) with zero mean     : -1310.057
##  ARIMA(1,0,2) with non-zero mean : -1344.401
##  ARIMA(0,0,2) with non-zero mean : -1339.083
##  ARIMA(1,0,1) with non-zero mean : -1343.23
##  ARIMA(1,0,3) with non-zero mean : -1342.909
##  ARIMA(0,0,3) with non-zero mean : -1338.586
##  ARIMA(2,0,1) with non-zero mean : -1344.49
```

```
##  ARIMA(2,0,0) with non-zero mean : -1337.772
##  ARIMA(3,0,1) with non-zero mean : -1340.263
##  ARIMA(3,0,0) with non-zero mean : -1336.605
##  ARIMA(3,0,2) with non-zero mean : -1339.877
##  ARIMA(2,0,1) with zero mean     : -1346.039
##  ARIMA(1,0,1) with zero mean     : -1344.622
##  ARIMA(2,0,0) with zero mean     : -1339.137
##  ARIMA(3,0,1) with zero mean     : -1341.886
##  ARIMA(2,0,2) with zero mean     : -1342.599
##  ARIMA(1,0,0) with zero mean     : -1337.26
##  ARIMA(1,0,2) with zero mean     : -1345.615
##  ARIMA(3,0,0) with zero mean     : -1338.058
##  ARIMA(3,0,2) with zero mean     : -1341.215
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(2,0,1) with zero mean     : -1343.187
##
##  Best model: ARIMA(2,0,1) with zero mean
```

The BRENT log-return series need an $ARMA(2,1)$ model to filter the mean process.

- **DUBAI**

```
dubai_arma <- auto.arima(return.train$DUBAI, stationary = T, seasonal = F, ic = "aic", allowdrift = FALS
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,0,2) with non-zero mean : -1381.634
##  ARIMA(0,0,0) with non-zero mean : -1330.647
##  ARIMA(1,0,0) with non-zero mean : -1376.046
##  ARIMA(0,0,1) with non-zero mean : -1384.144
##  ARIMA(0,0,0) with zero mean     : -1331.487
##  ARIMA(1,0,1) with non-zero mean : -1383.941
##  ARIMA(0,0,2) with non-zero mean : -1382.219
##  ARIMA(1,0,2) with non-zero mean : -1383.054
##  ARIMA(0,0,1) with zero mean     : -1385.363
##  ARIMA(1,0,1) with zero mean     : -1385.304
##  ARIMA(0,0,2) with zero mean     : -1383.422
##  ARIMA(1,0,0) with zero mean     : -1377.516
##  ARIMA(1,0,2) with zero mean     : -1384.336
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(0,0,1) with zero mean     : -1385.662
##
##  Best model: ARIMA(0,0,1) with zero mean
```

The DUBAI log-return series need an $ARMA(0,1)$ model to filter the mean process.

- **ESPO**

```
espo_arma <- auto.arima(return.train$ESPO, stationary = T, seasonal = F, ic = "aic", allowdrift = FALSE
```

```
##
##  Fitting models using approximations to speed things up...
```

```
##
##  ARIMA(2,0,2) with non-zero mean : -1422.952
##  ARIMA(0,0,0) with non-zero mean : -1370.71
##  ARIMA(1,0,0) with non-zero mean : -1417.302
##  ARIMA(0,0,1) with non-zero mean : -1425.049
##  ARIMA(0,0,0) with zero mean     : -1371.501
##  ARIMA(1,0,1) with non-zero mean : -1425.522
##  ARIMA(2,0,1) with non-zero mean : -1424.343
##  ARIMA(1,0,2) with non-zero mean : -1424.647
##  ARIMA(0,0,2) with non-zero mean : -1423.127
##  ARIMA(2,0,0) with non-zero mean : -1421.486
##  ARIMA(1,0,1) with zero mean     : -1426.861
##  ARIMA(0,0,1) with zero mean     : -1426.227
##  ARIMA(1,0,0) with zero mean     : -1418.762
##  ARIMA(2,0,1) with zero mean     : -1425.744
##  ARIMA(1,0,2) with zero mean     : -1425.858
##  ARIMA(0,0,2) with zero mean     : -1424.29
##  ARIMA(2,0,0) with zero mean     : -1422.722
##  ARIMA(2,0,2) with zero mean     : -1424.347
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(1,0,1) with zero mean     : -1424.754
##
##  Best model: ARIMA(1,0,1) with zero mean
```

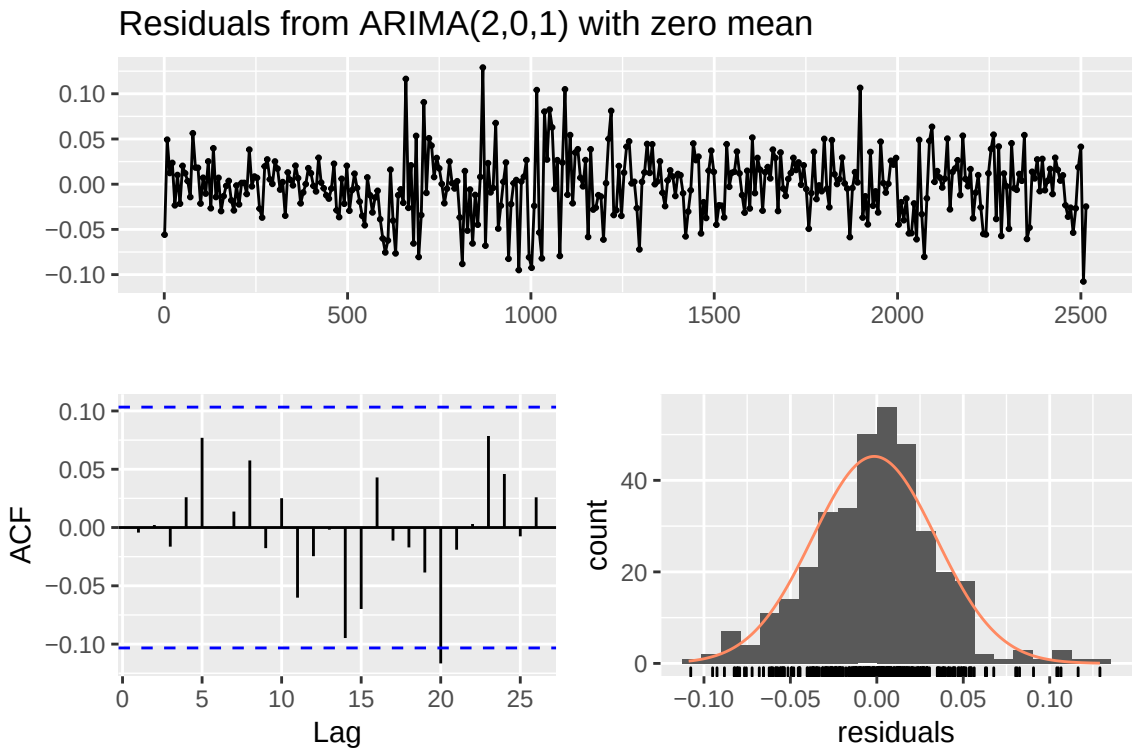
The ESPO log-return series need an $ARMA(1,1)$ model to filter the mean process.

#### 3.7. Check the residuals and determine the distributions of the residuals.

Usually in academic studies, researchers will conduct the Ljung-Box tests on the residuals of each $ARMA$ model and compare the p-value of the previous ones to examine whether autocorrelations in the mean processes are eliminated. In this study, I use the `checkresiduals()` function, which applies a light Ljung-Box test with $lags = 10$.

**WTI**

```
checkresiduals(wti_arma)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,1) with zero mean
## Q* = 4.1741, df = 7, p-value = 0.7595
##
## Model df: 3.   Total lags used: 10
```

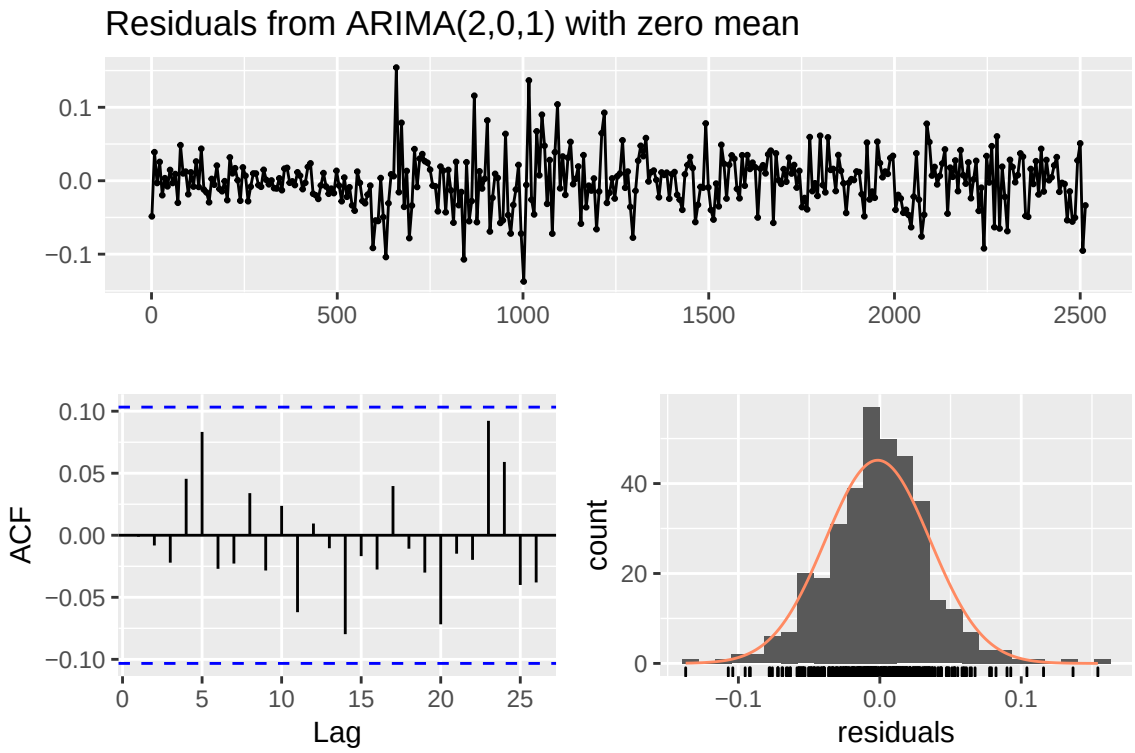
```
shapiro.test(wti_arma$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  wti_arma$residuals
## W = 0.98528, p-value = 0.0009999
```

The residuals do not follow a normal distribution.

**BRENT**

```
checkresiduals(brent_arma)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,1) with zero mean
## Q* = 4.9049, df = 7, p-value = 0.6716
##
## Model df: 3.   Total lags used: 10
```

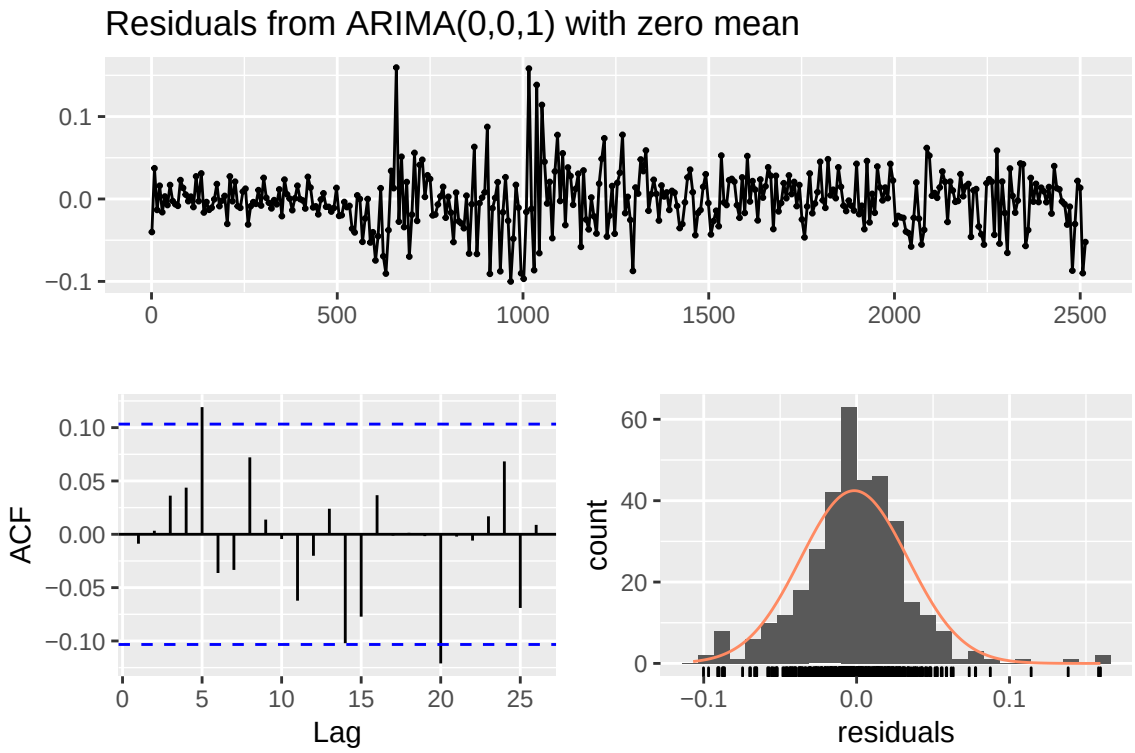
```
shapiro.test(brent_arma$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  brent_arma$residuals
## W = 0.98156, p-value = 0.0001451
```

The residuals do not follow a normal distribution.

**DUBAI**

```
checkresiduals(dubai_arma)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,1) with zero mean
## Q* = 9.3308, df = 9, p-value = 0.4073
##
## Model df: 1.   Total lags used: 10
```

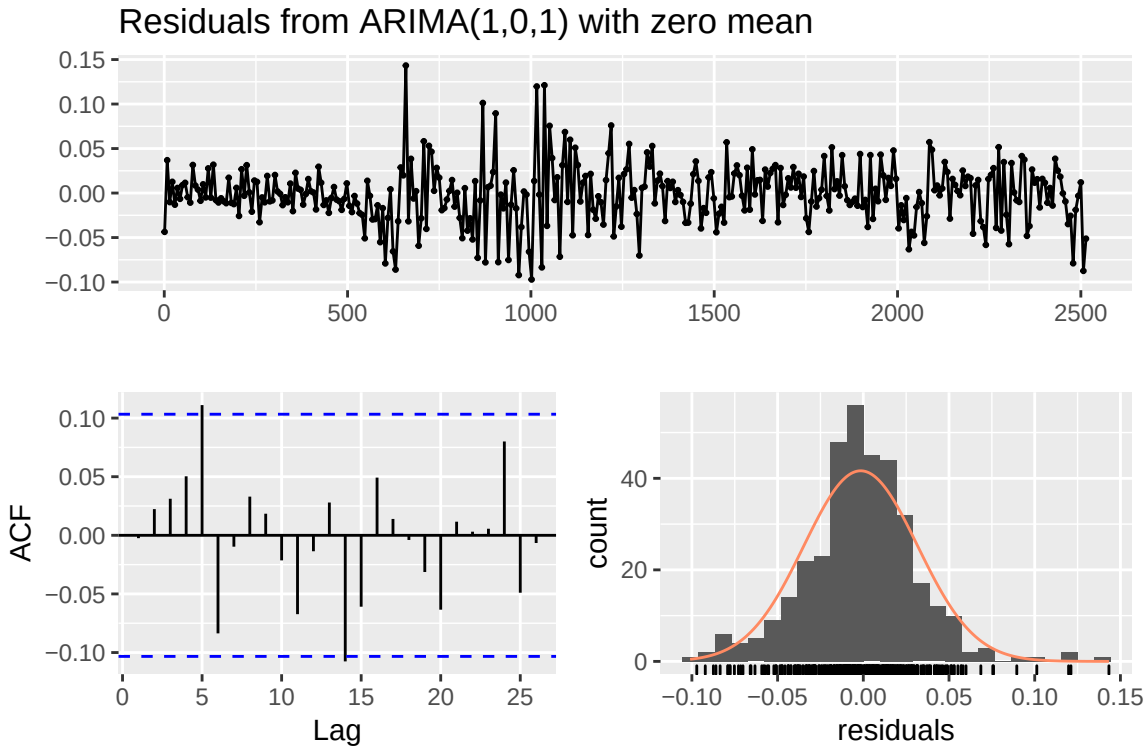
```
shapiro.test(dubai_arma$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  dubai_arma$residuals
## W = 0.95699, p-value = 9.063e-09
```

The residuals do not follow a normal distribution.

**ESPO**

```
checkresiduals(espo_arma)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,1) with zero mean
## Q* = 9.3078, df = 8, p-value = 0.317
##
## Model df: 2.   Total lags used: 10
```

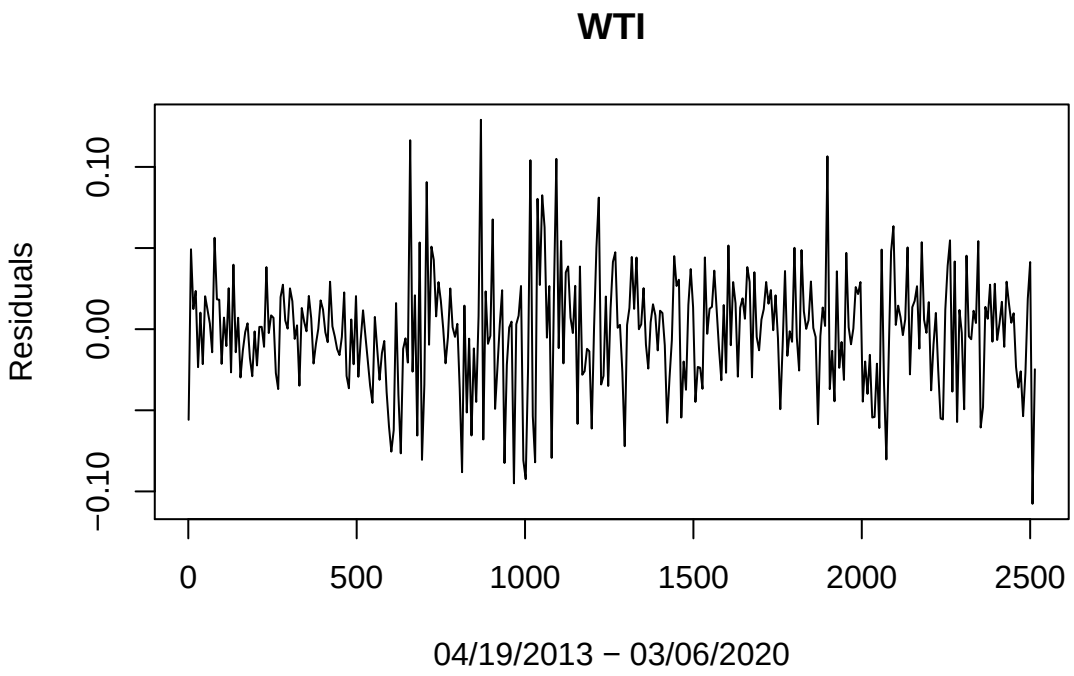
```
shapiro.test(espo_arma$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  espo_arma$residuals
## W = 0.97333, p-value = 3.438e-06
```

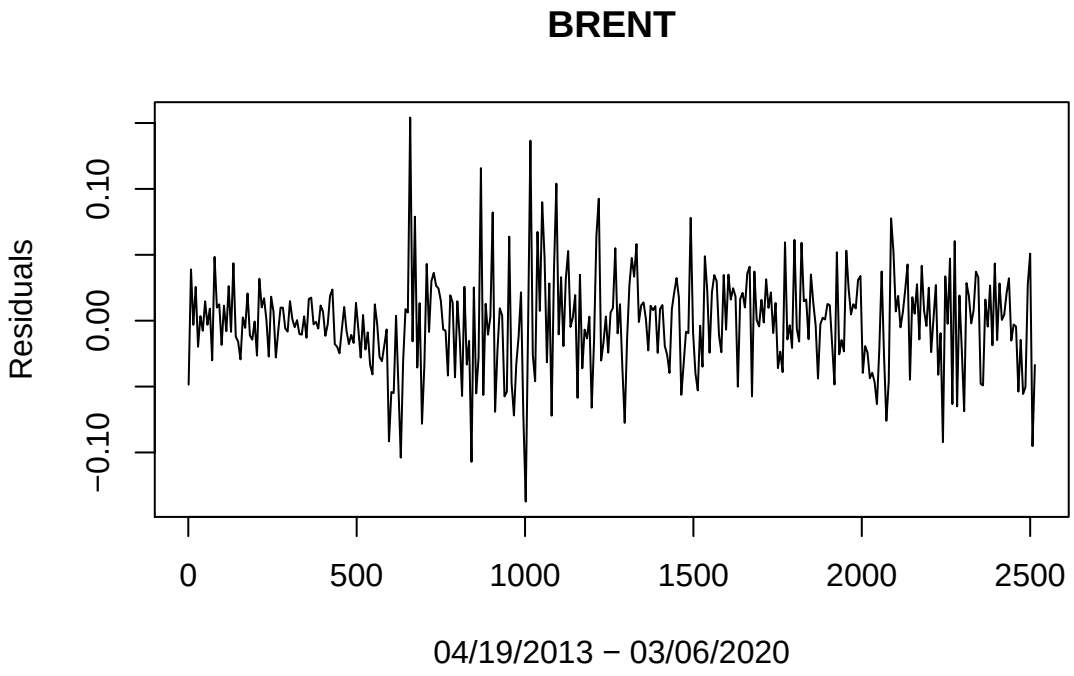
The residuals do not follow a normal distribution.

**3.8 Visualize the variance process.**

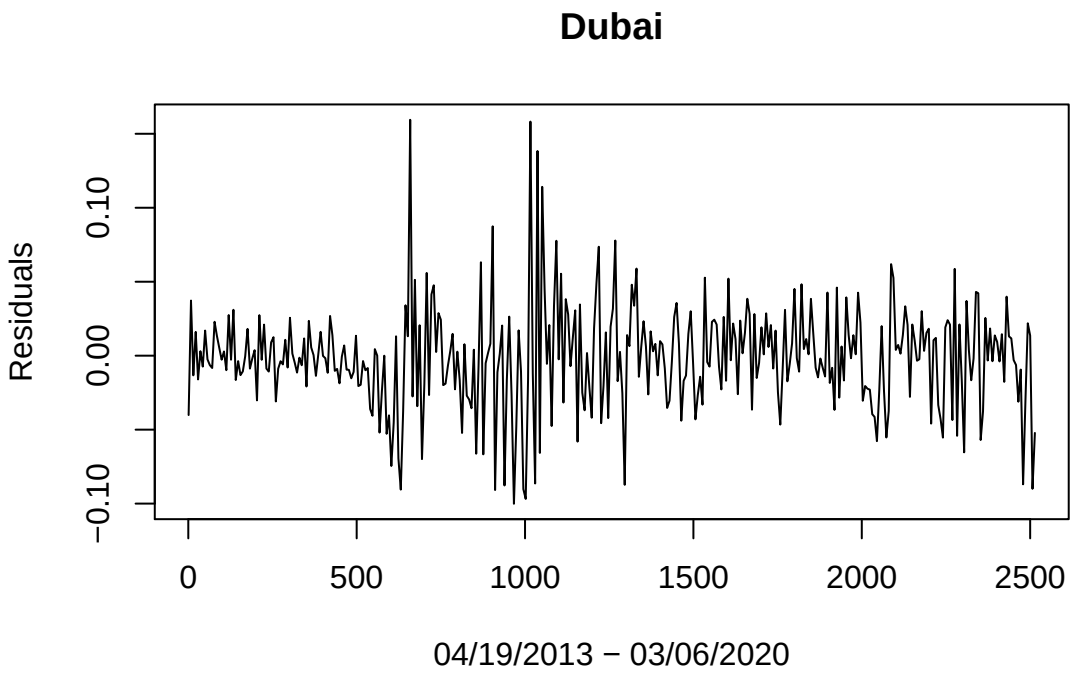
```
r_wti <- wti_arma$residuals
r_brent <- brent_arma$residuals
r_dubai <- dubai_arma$residuals
r_espo <- espo_arma$residuals

plot(r_wti, main="WTI",
     xlab="04/19/2013 - 03/06/2020",
     ylab="Residuals")
```



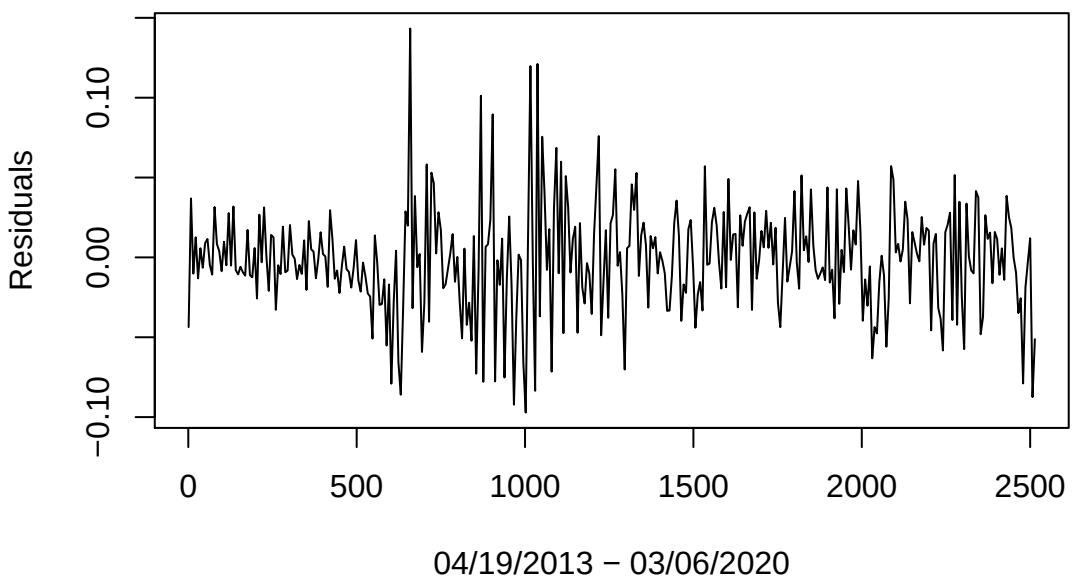

```
plot(r_brent, main="BRENT",
     xlab="04/19/2013 - 03/06/2020",
     ylab="Residuals")
```



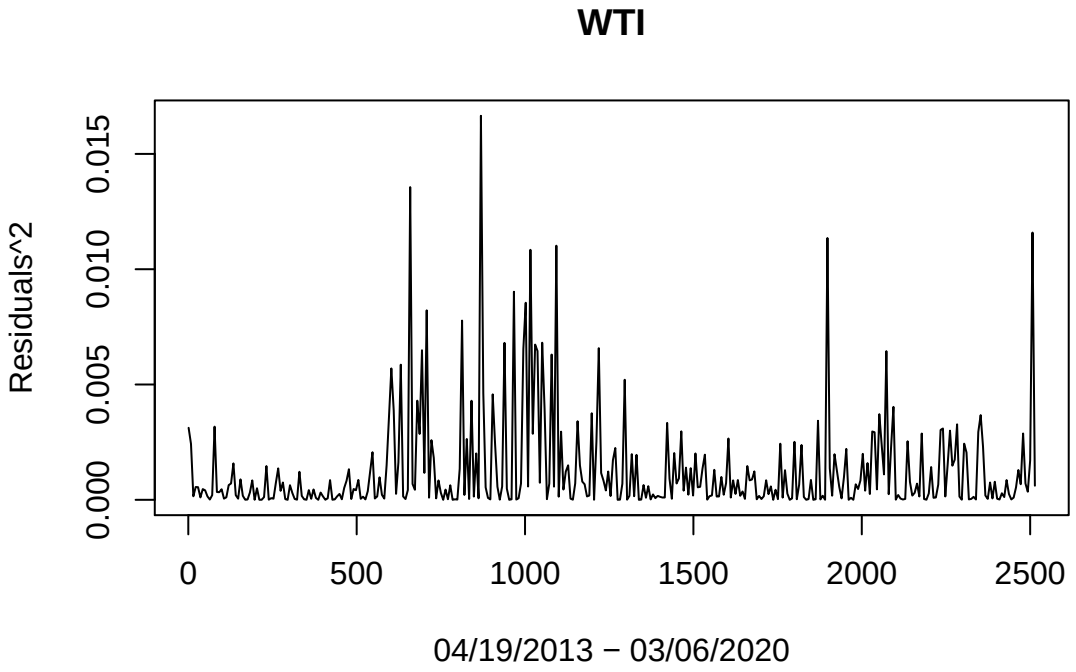

```
plot(r_dubai, main="Dubai",
     xlab="04/19/2013 - 03/06/2020",
     ylab="Residuals")
```



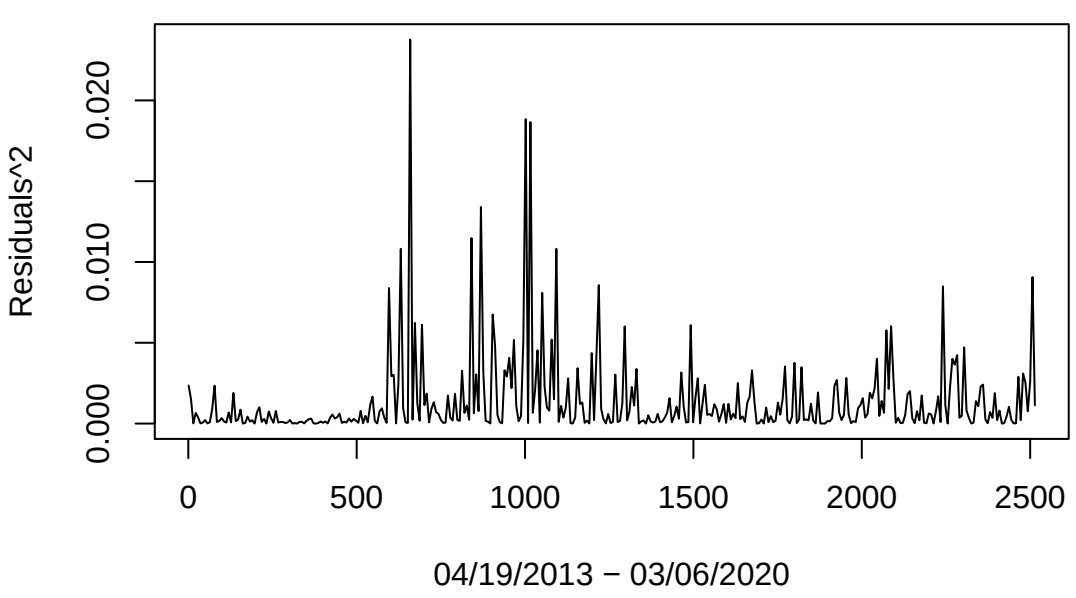

```
plot(r_espo, main="ESPO",
     xlab="04/19/2013 - 03/06/2020",
     ylab="Residuals")
```



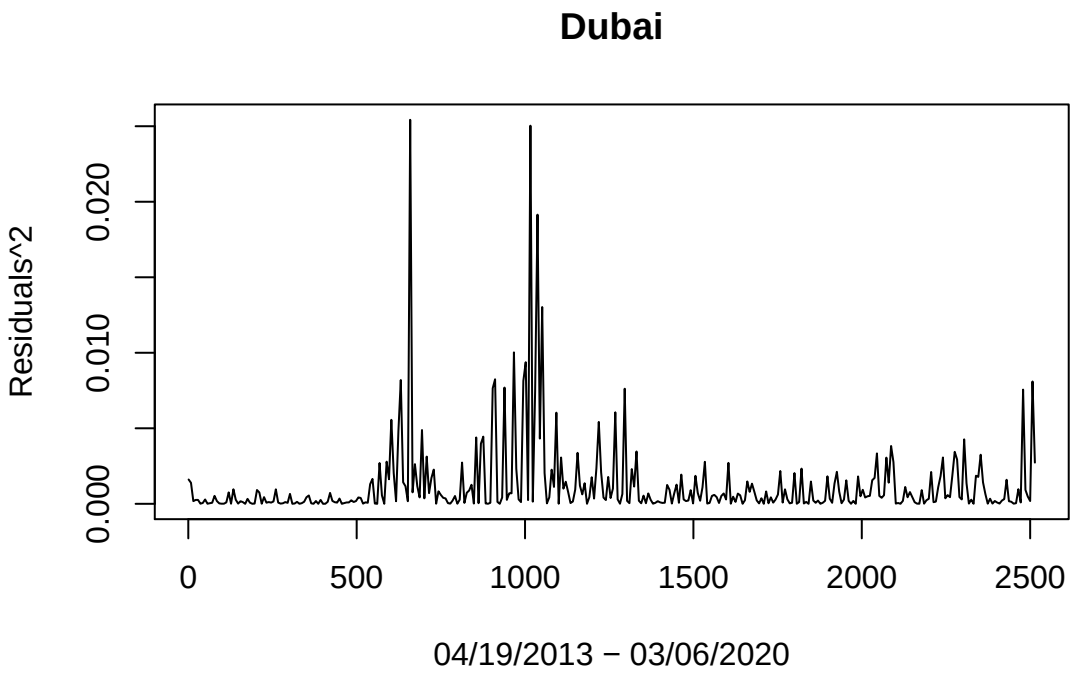

```
plot(r_wti^2, main="WTI",
     xlab="04/19/2013 - 03/06/2020",
     ylab="Residuals^2")
```



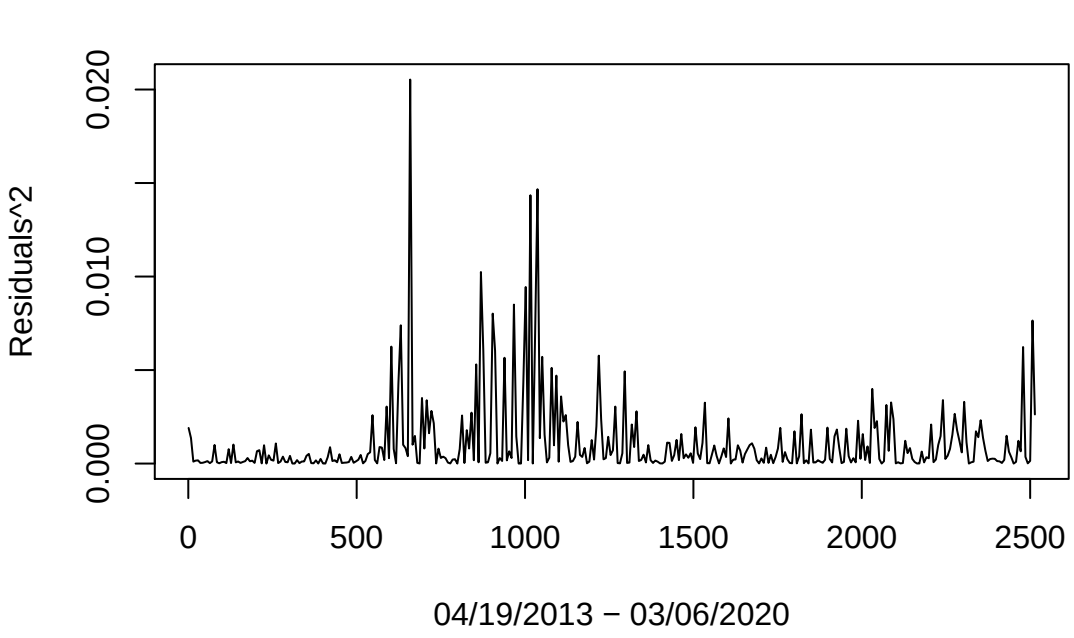

```
plot(r_brent^2, main="BRENT",
     xlab="04/19/2013 - 03/06/2020",
     ylab="Residuals^2")
```




```
plot(r_dubai^2, main="Dubai",
     xlab="04/19/2013 - 03/06/2020",
     ylab="Residuals^2")
```




```
plot(r_espo^2, main="ESPO",
     xlab="04/19/2013 - 03/06/2020",
     ylab="Residuals^2")
```




Intuitively, there exists conditional heteroscedasticity in all series. WTI and BRENT are more volatile than the others. Dubai is the least volatile market. It is clear that there were profund disruptions between 2015 and 2016. Now I will test the ARCH effect statistically.

**ARCH-LM Tests**

```
ArchTest(r_wti)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  r_wti
## Chi-squared = 45.262, df = 12, p-value = 9.292e-06
```

```
ArchTest(r_brent)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  r_brent
## Chi-squared = 44.732, df = 12, p-value = 1.145e-05
```

```
ArchTest(r_dubai)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  r_dubai
## Chi-squared = 68.027, df = 12, p-value = 7.478e-10
```

```
ArchTest(r_espo)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  r_espo
## Chi-squared = 56.984, df = 12, p-value = 7.958e-08
```

All series have ARCH effects in their variance processes.

### 3.9 The (E)DCC-MGARCH Models

The univarite GARCH models should have been fitted respectively to calculate the matrices that are essential to estimate the parameters in DCC models. However, the `dccfit()` function performs the univarite GARCH on the backend and will return the coefficients of univarite models in the result table as well. Hence, I will go directly to the DCC-MGARCH models.

First, I make multiple specifications for different series.

```
ugarch.spec1 <- ugarchspec(variance.model=list(garchOrder=c(1,1)),
                           mean.model=list(armaOrder=c(2,1)),
                           distribution.model = "sstd")
ugarch.spec2 <- ugarchspec(variance.model=list(garchOrder=c(1,1)),
                            mean.model=list(armaOrder=c(0,1)),
                            distribution.model = "sstd")
ugarch.spec3 <- ugarchspec(variance.model=list(garchOrder=c(1,1)),
                            mean.model=list(armaOrder=c(1,1)),
                            distribution.model = "sstd")
```

Then, I fit the DCC model with all 4 series together.

```
dcc.garch.spec = dccspec(uspec = multispec(c(ugarch.spec1,
                                             ugarch.spec1,
                                             ugarch.spec2,
                                             ugarch.spec3)),
                            dccOrder = c(1,1),
                            distribution = "mvt")
```

```r
dcc.fit = dccfit(dcc.garch.spec, data = return.train)

dcc.fit
```

```
##
## *---------------------------------*
## *          DCC GARCH Fit          *
## *---------------------------------*
##
## Distribution         :  mvt
## Model                :  DCC(1,1)
## No. Parameters       :  42
## [VAR GARCH DCC UncQ] : [0+33+3+6]
## No. Series           :  4
## No. Obs.             :  360
## Log-Likelihood       :  4011.804
## Av.Log-Likelihood    :  11.14
##
## Optimal Parameters
## ------------------------------------
##                  Estimate  Std. Error    t value Pr(>|t|)
## [WTI].mu       -0.000798    0.002392  -0.333381 0.738847
## [WTI].ar1      -0.609917    0.025893 -23.555219 0.000000
## [WTI].ar2       0.235153    0.025382   9.264387 0.000000
## [WTI].ma1       0.964612    0.008940 107.902260 0.000000
## [WTI].omega     0.000026    0.000022   1.155343 0.247950
## [WTI].alpha1    0.100664    0.034638   2.906212 0.003658
## [WTI].beta1     0.886059    0.042098  21.047550 0.000000
## [WTI].skew      0.934833    0.062877  14.867644 0.000000
## [WTI].shape    10.520950    5.514086   1.908013 0.056389
## [BRENT].mu     -0.000545    0.002322  -0.234733 0.814416
## [BRENT].ar1    -0.630384    0.072939  -8.642568 0.000000
## [BRENT].ar2     0.234228    0.051886   4.514270 0.000006
## [BRENT].ma1     0.957562    0.020198  47.409161 0.000000
## [BRENT].omega   0.000024    0.000023   1.051933 0.292830
## [BRENT].alpha1  0.166988    0.069361   2.407507 0.016062
## [BRENT].beta1   0.832012    0.070129  11.864008 0.000000
## [BRENT].skew    0.911949    0.075341  12.104239 0.000000
## [BRENT].shape  10.656765    5.123989   2.079779 0.037546
## [DUBAI].mu     -0.000151    0.002066  -0.073169 0.941672
## [DUBAI].ma1     0.380306    0.052363   7.262931 0.000000
## [DUBAI].omega   0.000029    0.000020   1.444063 0.148721
## [DUBAI].alpha1  0.190241    0.054571   3.486135 0.000490
## [DUBAI].beta1   0.802627    0.053808  14.916443 0.000000
## [DUBAI].skew    0.969715    0.069251  14.002839 0.000000
## [DUBAI].shape  15.084562   12.401141   1.216385 0.223838
## [ESPO].mu      -0.000002    0.002235  -0.000872 0.999304
## [ESPO].ar1      0.045083    0.145076   0.310754 0.755988
## [ESPO].ma1      0.348381    0.122471   2.844603 0.004447
## [ESPO].omega    0.000024    0.000019   1.270867 0.203776
## [ESPO].alpha1   0.171236    0.064077   2.672354 0.007532
## [ESPO].beta1    0.821540    0.059883  13.719179 0.000000
## [ESPO].skew     0.980208    0.070732  13.857984 0.000000
## [ESPO].shape   38.926643   83.388840   0.466809 0.640637
```

```
## [Joint]dcca1    0.116725    0.027285   4.278028 0.000019
## [Joint]dccb1    0.640543    0.147438   4.344478 0.000014
## [Joint]mshape   7.460614    1.046338   7.130214 0.000000
##
## Information Criteria
## ---------------------
##
## Akaike       -22.054
## Bayes        -21.601
## Shibata      -22.078
## Hannan-Quinn -21.874
##
##
## Elapsed time : 3.664056
```

Both coefficients, $DCC\alpha$ and $DCC\beta$, are statistically significant at a 1% confidence level. More importantly, they are jointly significant. Therefore, the major markets have dynamic (conditional) correlations over time. As a result, one has to measure the volatility spillovers dynamically when making forecasts.