Untitled

Mingwen Yang

4/26/2020

Load data & Unit Root Test

```r
library(vars)
```

```
## Loading required package: MASS

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich

## Loading required package: urca

## Loading required package: lmtest
```

```r
library(quantmod)
```

```
## Loading required package: xts

## Loading required package: TTR

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```r
library(parallel)
library(rugarch)
```

```
##
## Attaching package: 'rugarch'

## The following object is masked from 'package:stats':
##
##     sigma
```

```r
library(rmgarch)
```

```
##
## Attaching package: 'rmgarch'
```

```
## The following objects are masked from 'package:xts':
##
##     first, last
```

```r
library(tseries)
library(zoo)
library(forecast)
```

```
## Registered S3 methods overwritten by 'forecast':
##   method             from
##   fitted.fracdiff    fracdiff
##   residuals.fracdiff fracdiff
```

```r
library(fGarch)
```

```
## Loading required package: timeDate

## Loading required package: timeSeries

##
## Attaching package: 'timeSeries'

## The following object is masked from 'package:zoo':
##
##     time<-

## Loading required package: fBasics

##
## Attaching package: 'fBasics'

## The following object is masked from 'package:TTR':
##
##     volatility
```

```r
library(FinTS)
```

```
##
## Attaching package: 'FinTS'

## The following object is masked from 'package:forecast':
##
##     Acf
```

```r
library(lmtest)
library(urca)
library(xts)
library(Metrics)
```

```
##
## Attaching package: 'Metrics'

## The following object is masked from 'package:forecast':
##
##     accuracy
```

```r
library(DistributionUtils)
```

```
##
## Attaching package: 'DistributionUtils'

## The following object is masked from 'package:fBasics':
```

```
##
##     tsHessian
## The following objects are masked from 'package:timeDate':
##
##     kurtosis, skewness
```

```
library(readxl)
library(fUnitRoots)
```

```
##
## Attaching package: 'fUnitRoots'
## The following objects are masked from 'package:urca':
##
##     punitroot, qunitroot, unitrootTable
```

```
ratedata = read.csv("D:/2019-2020/Time Series/hw/group/rate.csv")
rate = xts(ratedata$currency.rate, as.Date(ratedata$ï..DATE, format='%m/%d/%y', tz = "US"))
rate_train = rate[1:1080]
rate_test = rate[1081:1343]
#head(rate_train)
#tail(rate_train)
#head(rate_test)
#tail(rate_test)

df = read.csv("D:/2019-2020/Time Series/hw/group/data.csv")
data = ts(df[2:10], start = c(2015, 1), end = c(2020, 2), frequency=12)
head(data)
```

```
##           M1_US   M1_EU   GDP_US  GDP_EU   CPI_US CPI_EU XAU_USD WTI_USD   RATE
## Jan 2015 2941.1 6031.11 17984.18 99.6455  99.0423  98.24 1251.57 47.2480 0.8603
## Feb 2015 2979.6 6061.48 17984.18 99.6860  99.2933  98.85 1227.08 50.7400 0.8808
## Mar 2015 3023.9 6119.64 17984.18 99.7202  99.5608 100.00 1179.55 48.0423 0.9237
## Apr 2015 3035.4 6203.81 18219.40 99.7481  99.6646 100.43 1200.30 54.6200 0.9245
## May 2015 2975.9 6302.87 18219.40 99.7713  99.9932 100.71 1199.07 59.4310 0.8964
## Jun 2015 3020.7 6361.57 18219.40 99.7918 100.2700 100.72 1182.39 59.7718 0.8901
```

```
M1_US_rt = dailyReturn(data[, 1])
M1_EU_rt = dailyReturn(data[, 2])
GDP_US_rt = dailyReturn(data[, 3])
GDP_EU_rt = dailyReturn(data[, 4])
CPI_US_rt = dailyReturn(data[, 5])
CPI_EU_rt = dailyReturn(data[, 6])
XAU_USD_rt = dailyReturn(data[, 7])
WTI_USD_rt = dailyReturn(data[, 8])
RATE_rt = dailyReturn(data[, 9])

df_US = read_excel("D:/2019-2020/Time Series/hw/group/US_part.xlsx", sheet=1, na="NA")
data_US = ts(df_US[2:4], start = c(2011, 1), end = c(2020, 2), frequency = 12)
#head(data_US)
M1_US1_rt = dailyReturn(data_US[, 1])
CPI_US1_rt = dailyReturn(data_US[, 2])
RATE1_rt = dailyReturn(data_US[, 3])

df_D = read_excel("D:/2019-2020/Time Series/hw/group/DAILY_part.xlsx", sheet=1, na="NA")
data_D = xts(df_D[, 2:4], as.Date(df_D$DATE, format='%m/%d/%y', tz = "US"))
```

```r
#head(data_D)
XAU_USD1_rt = dailyReturn(data_D$XAU_USD, type = "log")
WTI_USD1_rt = dailyReturn(data_D$WTI_USD, type = "log")
RATE2_rt = dailyReturn(data_D$`currency rate`, type = "log")
```

```r
adfTest(M1_US_rt)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -6.4703
##   P VALUE:
##     0.01
##
## Description:
##  Thu Apr 30 20:18:13 2020 by user: yangm
```

```r
adfTest(M1_EU_rt)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -2.8249
##   P VALUE:
##     0.01
##
## Description:
##  Thu Apr 30 20:18:13 2020 by user: yangm
```

```r
adfTest(GDP_US_rt)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -5.0516
##   P VALUE:
##     0.01
##
## Description:
##  Thu Apr 30 20:18:13 2020 by user: yangm
```

```r
adfTest(GDP_EU_rt)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -1.0457
##   P VALUE:
##     0.2793
##
## Description:
##  Thu Apr 30 20:18:13 2020 by user: yangm
```

```r
adfTest(CPI_US_rt)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -2.7051
##   P VALUE:
##     0.01
##
## Description:
##  Thu Apr 30 20:18:13 2020 by user: yangm
```

```r
adfTest(CPI_EU_rt)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -5.4888
##   P VALUE:
##     0.01
##
## Description:
##  Thu Apr 30 20:18:13 2020 by user: yangm
```

```r
adfTest(XAU_USD_rt)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
```

```
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -4.7869
##   P VALUE:
##     0.01
##
## Description:
##  Thu Apr 30 20:18:13 2020 by user: yangm
```

```r
adfTest(WTI_USD_rt)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -5.2505
##   P VALUE:
##     0.01
##
## Description:
##  Thu Apr 30 20:18:13 2020 by user: yangm
```

```r
adfTest(RATE_rt)
```

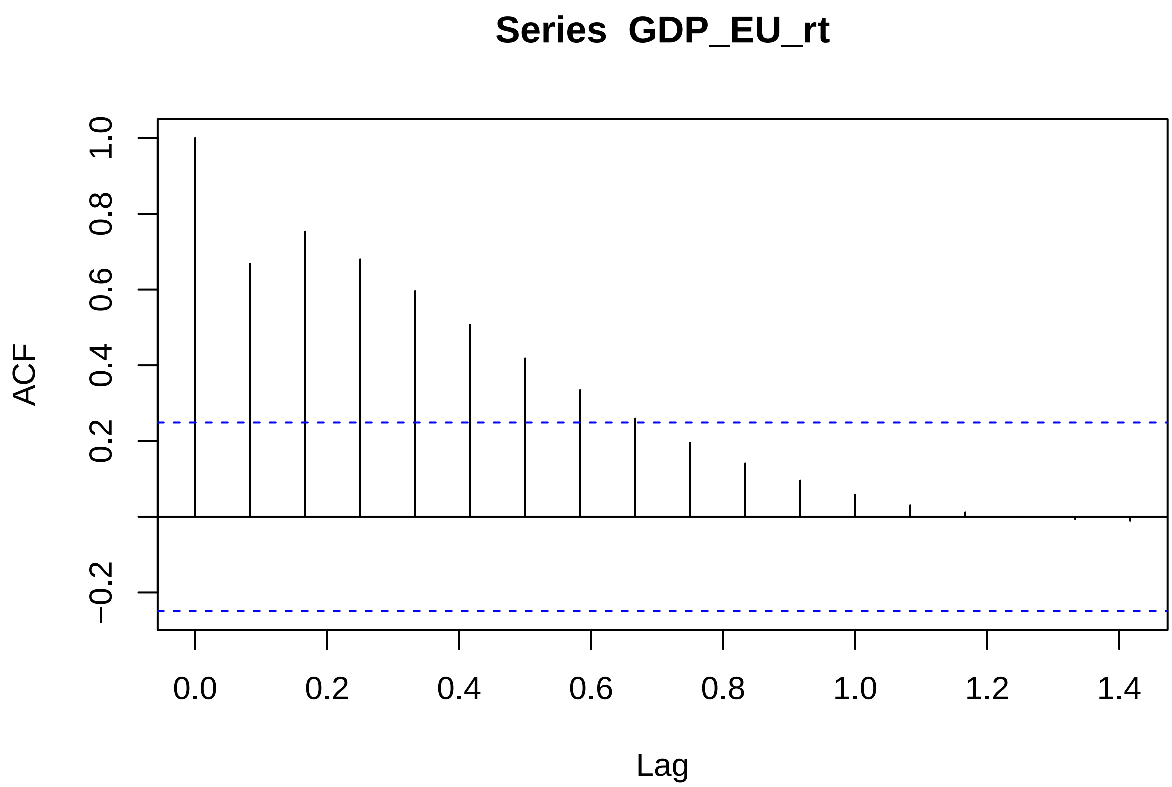
```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -5.3706
##   P VALUE:
##     0.01
##
## Description:
##  Thu Apr 30 20:18:13 2020 by user: yangm
```

```r
GDP_EU_sta = ur.df(GDP_EU_rt, type='none', selectlags='AIC')
summary(GDP_EU_sta)
```

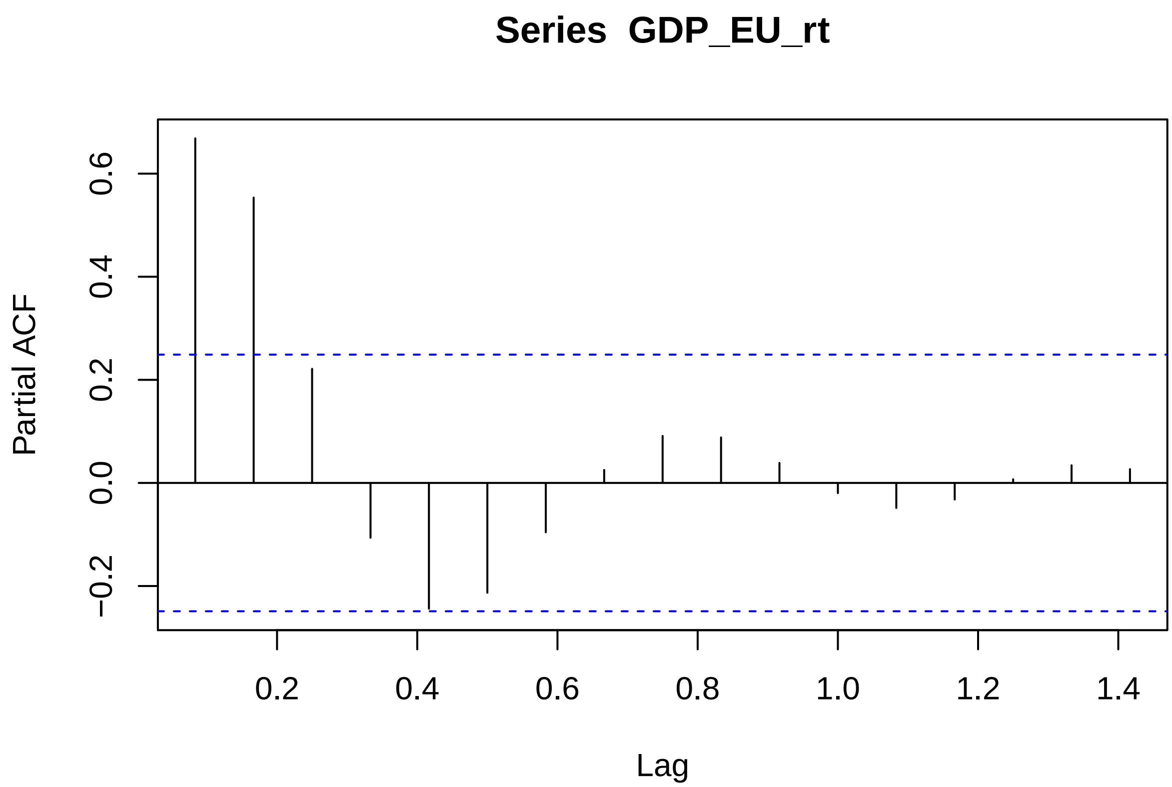
```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
```

```
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -8.207e-04 -1.803e-04 -3.924e-05  6.816e-05  1.858e-03
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.08348    0.07983  -1.046      0.3
## z.diff.lag -1.04775    0.17428  -6.012  1.3e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.00035 on 58 degrees of freedom
## Multiple R-squared:  0.4326, Adjusted R-squared:  0.413
## F-statistic: 22.11 on 2 and 58 DF,  p-value: 7.307e-08
##
##
## Value of test-statistic is: -1.0457
##
## Critical values for test statistics:
##      1pct  5pct 10pct
## tau1 -2.6 -1.95 -1.61
```

```
acf(GDP_EU_rt)
```



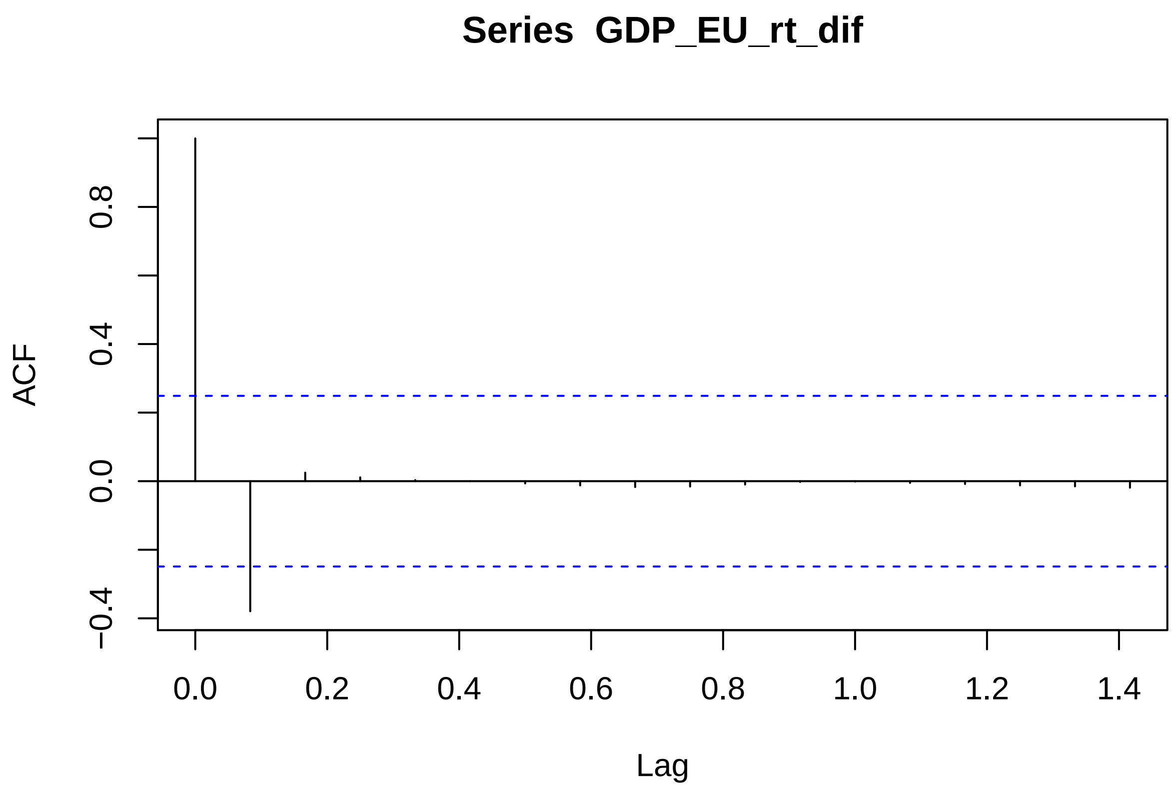

```
pacf(GDP_EU_rt)
```



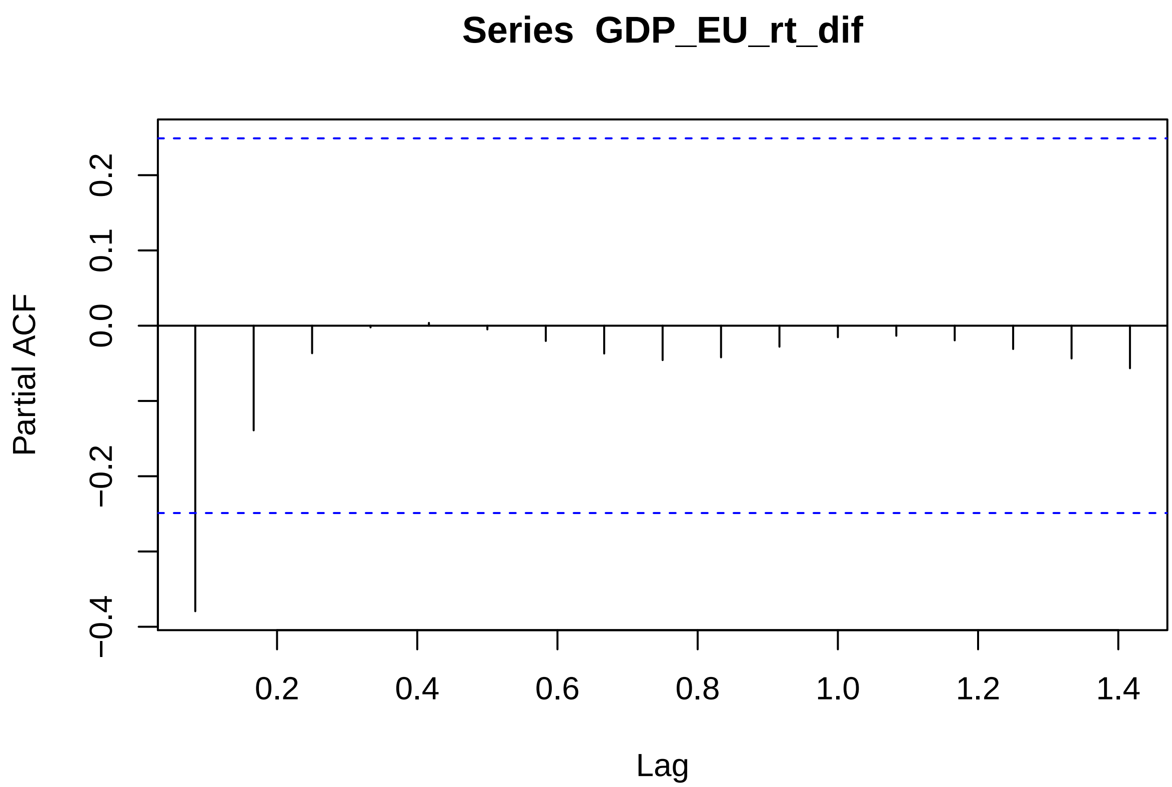

```r
GDP_EU_rt_dif = diff(GDP_EU_rt, 1)
adfTest(GDP_EU_rt_dif)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -2.4217
##   P VALUE:
##     0.01791
##
## Description:
##  Thu Apr 30 20:18:13 2020 by user: yangm
```

```r
acf(GDP_EU_rt_dif, na.action = na.pass)
```




```r
pacf(GDP_EU_rt_dif, na.action = na.pass)
```




```
GDP_EU_rt_dif[1] = 0
data_rt = cbind(M1_US_rt, M1_EU_rt, GDP_US_rt, GDP_EU_rt_dif, CPI_US_rt, CPI_EU_rt, XAU_USD_rt, WTI_USD
names(data_rt) = c("M1_US", "M1_EU", "GDP_US", "GDP_EU", "CPI_US", "CPI_EU", "XAU_USD", "WTI_USD", "RAT
head(data_rt)
```

```
##                 M1_US       M1_EU     GDP_US        GDP_EU      CPI_US
## Jan 2015  0.000000000 0.000000000 0.00000000  0.000000e+00 0.000000000
## Feb 2015  0.013090340 0.005035557 0.00000000  4.064408e-04 0.002534271
## Mar 2015  0.014867767 0.009595016 0.00000000 -6.336357e-05 0.002694039
## Apr 2015  0.003803036 0.013754077 0.01307966 -6.329443e-05 0.001042579
## May 2015 -0.019602029 0.015967607 0.00000000 -4.719695e-05 0.003297058
## Jun 2015  0.015054269 0.009313218 0.00000000 -2.711597e-05 0.002768188
##                CPI_EU      XAU_USD      WTI_USD          RATE
## Jan 2015 0.000000e+00  0.000000000  0.000000000  0.0000000000
## Feb 2015 6.209283e-03 -0.019567423  0.073907890  0.0238288969
## Mar 2015 1.163379e-02 -0.038734231 -0.053167127  0.0487057221
## Apr 2015 4.300000e-03  0.017591454  0.136914761  0.0008660821
## May 2015 2.788012e-03 -0.001024744  0.088081289 -0.0303948080
## Jun 2015 9.929501e-05 -0.013910781  0.005734381 -0.0070281124
```

```
data_rt_train = data_rt[1:50]
data_rt_test = data_rt[51:62]
head(data_rt_train)
```

```
##                 M1_US       M1_EU     GDP_US        GDP_EU      CPI_US
## Jan 2015  0.000000000 0.000000000 0.00000000  0.000000e+00 0.000000000
## Feb 2015  0.013090340 0.005035557 0.00000000  4.064408e-04 0.002534271
```

```
## Mar 2015  0.014867767 0.009595016 0.00000000 -6.336357e-05 0.002694039
## Apr 2015  0.003803036 0.013754077 0.01307966 -6.329443e-05 0.001042579
## May 2015 -0.019602029 0.015967607 0.00000000 -4.719695e-05 0.003297058
## Jun 2015  0.015054269 0.009313218 0.00000000 -2.711597e-05 0.002768188
##                CPI_EU      XAU_USD      WTI_USD          RATE
## Jan 2015 0.000000e+00  0.000000000  0.000000000  0.0000000000
## Feb 2015 6.209283e-03 -0.019567423  0.073907890  0.0238288969
## Mar 2015 1.163379e-02 -0.038734231 -0.053167127  0.0487057221
## Apr 2015 4.300000e-03  0.017591454  0.136914761  0.0008660821
## May 2015 2.788012e-03 -0.001024744  0.088081289 -0.0303948080
## Jun 2015 9.929501e-05 -0.013910781  0.005734381 -0.0070281124
```

```r
tail(data_rt_train)
```

```
##                 M1_US         M1_EU      GDP_US        GDP_EU        CPI_US
## Sep 2018 -0.003960826  0.0087118555 0.000000000  1.171062e-04  1.413229e-03
## Oct 2018  0.012774071  0.0009322294 0.007135121  1.033482e-04  2.045445e-03
## Nov 2018 -0.011241394  0.0118258355 0.000000000  6.661096e-05 -4.547615e-04
## Dec 2018  0.032693249  0.0056064240 0.000000000  1.185759e-05 -3.564701e-05
## Jan 2019 -0.013537716 -0.0046682541 0.009619336 -4.496457e-05 -4.080790e-04
## Feb 2019 -0.011560848  0.0049636199 0.000000000 -1.008665e-04  2.499214e-03
##                 CPI_EU      XAU_USD      WTI_USD          RATE
## Sep 2018  0.0039419287 -0.001924455  0.031029314 -0.0094710095
## Oct 2018  0.0024899445  0.014440373  0.005364252  0.0145755597
## Nov 2018 -0.0055406955  0.004525520 -0.199687884  0.0114929318
## Dec 2018 -0.0004803074  0.025408943 -0.135881640 -0.0001136235
## Jan 2019 -0.0103796252  0.032352119  0.052681773 -0.0048863636
## Feb 2019  0.0032048169  0.020675514  0.066100455  0.0054813292
```

```r
head(data_rt_test)
```

```
##                 M1_US       M1_EU      GDP_US        GDP_EU       CPI_US
## Mar 2019  0.013992059 0.016589706 0.000000000 -1.259414e-04 0.0036098177
## Apr 2019  0.017555076 0.005449530 0.011443290 -1.081730e-04 0.0033375184
## May 2019 -0.008377402 0.010288682 0.000000000 -7.441387e-05 0.0008785533
## Jun 2019  0.010639421 0.010970957 0.000000000 -4.658100e-05 0.0009205101
## Jul 2019  0.008542097 0.003311277 0.009478466 -4.368961e-05 0.0026819653
## Aug 2019 -0.003522586 0.010211425 0.000000000 -4.880721e-05 0.0008089162
##                CPI_EU      XAU_USD     WTI_USD         RATE
## Mar 2019  0.010164569 -0.013835505  0.05454139  0.003747871
## Apr 2019  0.007187350 -0.011208315  0.09938069  0.006675718
## May 2019  0.001332065 -0.001671539 -0.04662916  0.004495897
## Jun 2019  0.001615355  0.060424114 -0.09752982 -0.006489874
## Jul 2019 -0.004743383  0.039686270  0.04817172  0.001576754
## Aug 2019  0.001429797  0.061982595 -0.04623632  0.010007871
```

```r
tail(data_rt_test)
```

```
##                  M1_US         M1_EU       GDP_US        GDP_EU       CPI_US
## Sep 2019  0.0071220628  0.0001729649  0.000000000 -4.793411e-05 0.0011661543
## Oct 2019  0.0122077118  0.0065624278  0.008661189 -1.696112e-05 0.0024792327
## Nov 2019  0.0001274892  0.0142193326  0.000000000  2.115868e-05 0.0023127736
## Dec 2019  0.0303130736  0.0003387976  0.000000000  4.132768e-05 0.0024048826
## Jan 2020 -0.0146734962 -0.0052919190 -0.024350913  1.889987e-03 0.0014545081
## Feb 2020 -0.0089904571  0.0089903623  0.000000000 -2.883148e-03 0.0008892031
##                CPI_EU      XAU_USD     WTI_USD          RATE
```

```
## Sep 2019  0.002189225  0.003166009  0.04060189  0.0099087063
## Oct 2019  0.001424637 -0.008771872 -0.05294307 -0.0003307243
## Nov 2019 -0.003224583 -0.016561873  0.05672116 -0.0034186149
## Dec 2019  0.003139867  0.008522418  0.04619973 -0.0019918114
## Jan 2020 -0.010054064  0.051815184 -0.03106880 -0.0032154341
## Feb 2020  0.001820446  0.023057194 -0.12308128  0.0190211346
```

```
adfTest(M1_US1_rt)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -5.8253
##   P VALUE:
##     0.01
##
## Description:
##  Thu Apr 30 20:18:13 2020 by user: yangm
```

```
adfTest(CPI_US1_rt)
```

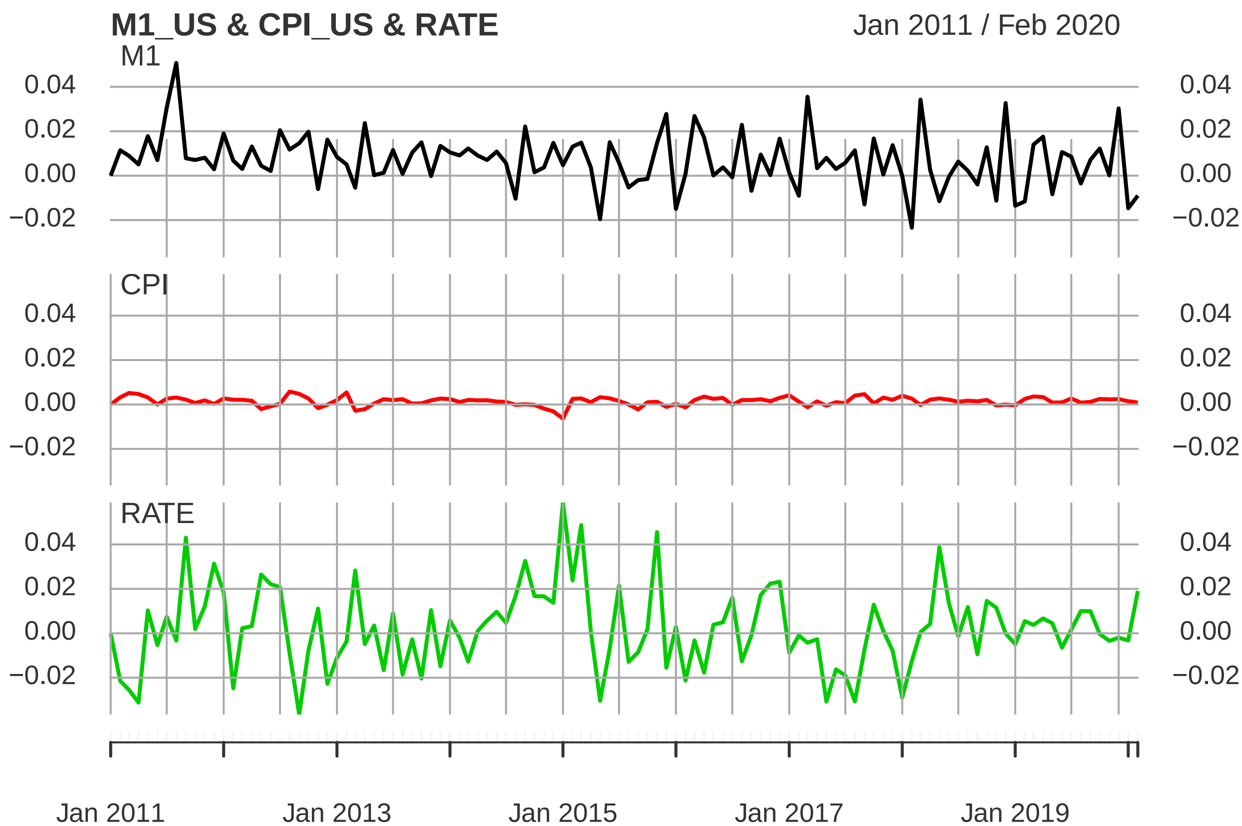
```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -4.7782
##   P VALUE:
##     0.01
##
## Description:
##  Thu Apr 30 20:18:13 2020 by user: yangm
```

```
adfTest(RATE1_rt)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -5.5966
##   P VALUE:
##     0.01
##
## Description:
```

```
##  Thu Apr 30 20:18:13 2020 by user: yangm
```

```r
data_US_rt = cbind(M1_US1_rt, CPI_US1_rt, RATE1_rt)
names(data_US_rt) = c("M1", "CPI", "RATE")
#head(data_US_rt)
plot(as.xts(data_US_rt), type="l", multi.panel=TRUE, theme="white", main="M1_US & CPI_US & RATE", major
```




```r
adfTest(XAU_USD1_rt)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -25.9239
##   P VALUE:
##     0.01
##
## Description:
##  Thu Apr 30 20:18:13 2020 by user: yangm
```

```r
adfTest(WTI_USD1_rt)
```

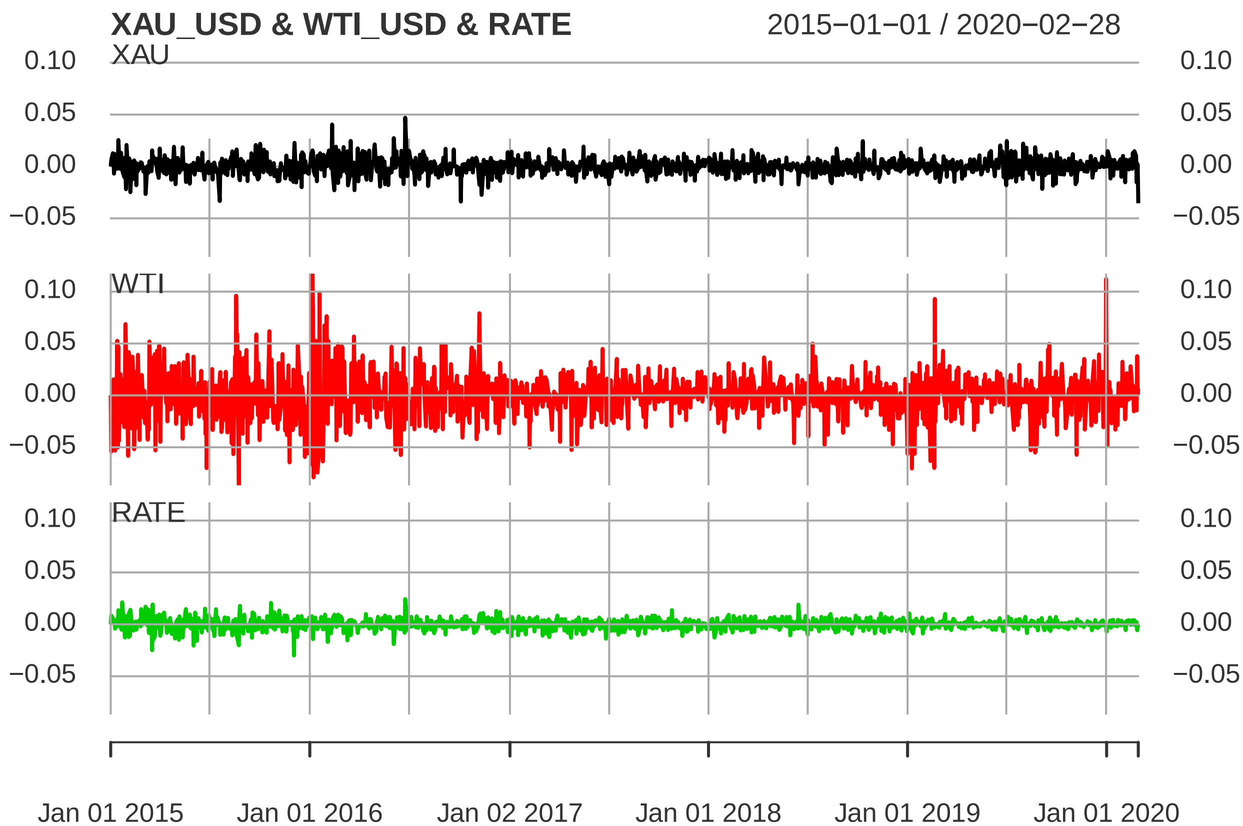
```
##
## Title:
```

```
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -27.2385
##   P VALUE:
##     0.01
##
## Description:
##  Thu Apr 30 20:18:13 2020 by user: yangm
```

```
adfTest(RATE2_rt)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -26.3354
##   P VALUE:
##     0.01
##
## Description:
##  Thu Apr 30 20:18:13 2020 by user: yangm
```

```
data_D_rt = data.frame(XAU_USD1_rt, WTI_USD1_rt, RATE2_rt)
names(data_D_rt) = c("XAU", "WTI", "RATE")
#head(data_D_rt)
plot(as.xts(data_D_rt), type="l", multi.panel=TRUE, theme="white", main="XAU_USD & WTI_USD & RATE", maj
```




```
data_D_rt_train = data_D_rt[1:1330,]
data_D_rt_test = data_D_rt[1331:1344,]
```

ARIMA & GARCH

Use "auto.arima" to fit a model for daily return of currency rate.
Then check out ARCH effect to build GARCH model.

```
md_arima_1 = auto.arima(rate_train)
md_arima_1
```

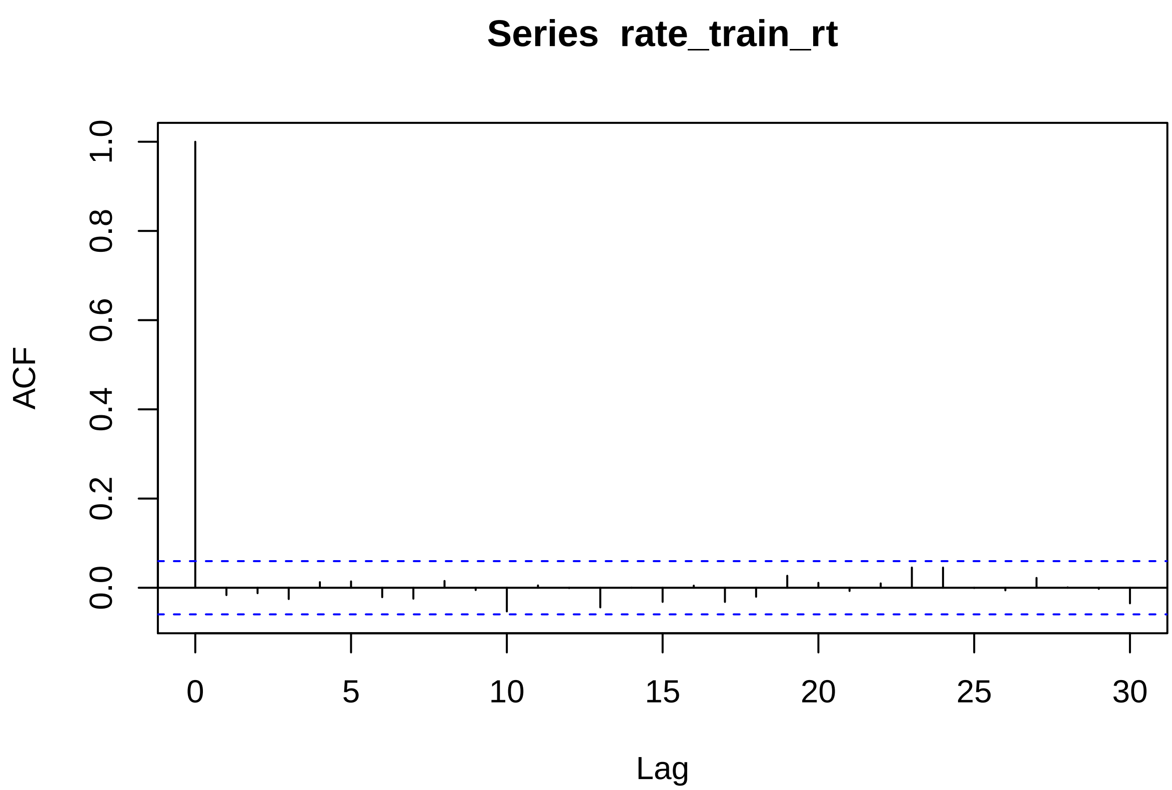
```
## Series: rate_train
## ARIMA(1,1,1)
##
## Coefficients:
##          ar1      ma1
##       0.8903  -0.9086
## s.e.  0.0926   0.0844
##
## sigma^2 estimated as 2.45e-05:  log likelihood=4197.75
## AIC=-8389.5   AICc=-8389.48   BIC=-8374.55
```

```
rate_train_rt = dailyReturn(rate_train, type = "log")
rate_test_rt = dailyReturn(rate_test, type = "log")
adfTest(rate_train_rt)
```

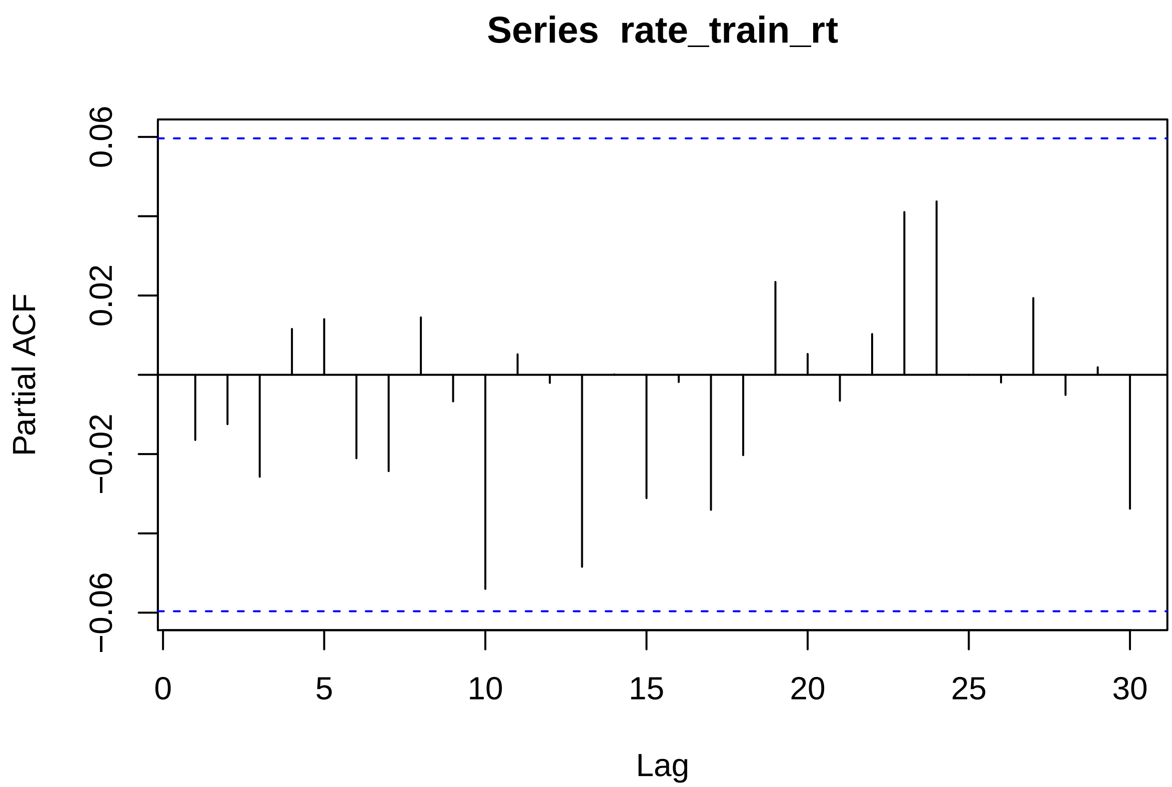
```
##
```

```
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -23.699
##   P VALUE:
##     0.01
##
## Description:
##  Thu Apr 30 20:18:13 2020 by user: yangm
```

```r
acf(rate_train_rt)
```




```r
pacf(rate_train_rt)
```




```
md_arima_2 = auto.arima(rate_train_rt)
md_arima_2
```

```
## Series: rate_train_rt
## ARIMA(2,0,1) with zero mean
##
## Coefficients:
##          ar1      ar2      ma1
##       0.8742  -0.0013  -0.8919
## s.e.  0.1063   0.0314   0.1021
##
## sigma^2 estimated as 3.077e-05:  log likelihood=4079.17
## AIC=-8150.34   AICc=-8150.3   BIC=-8130.4
```

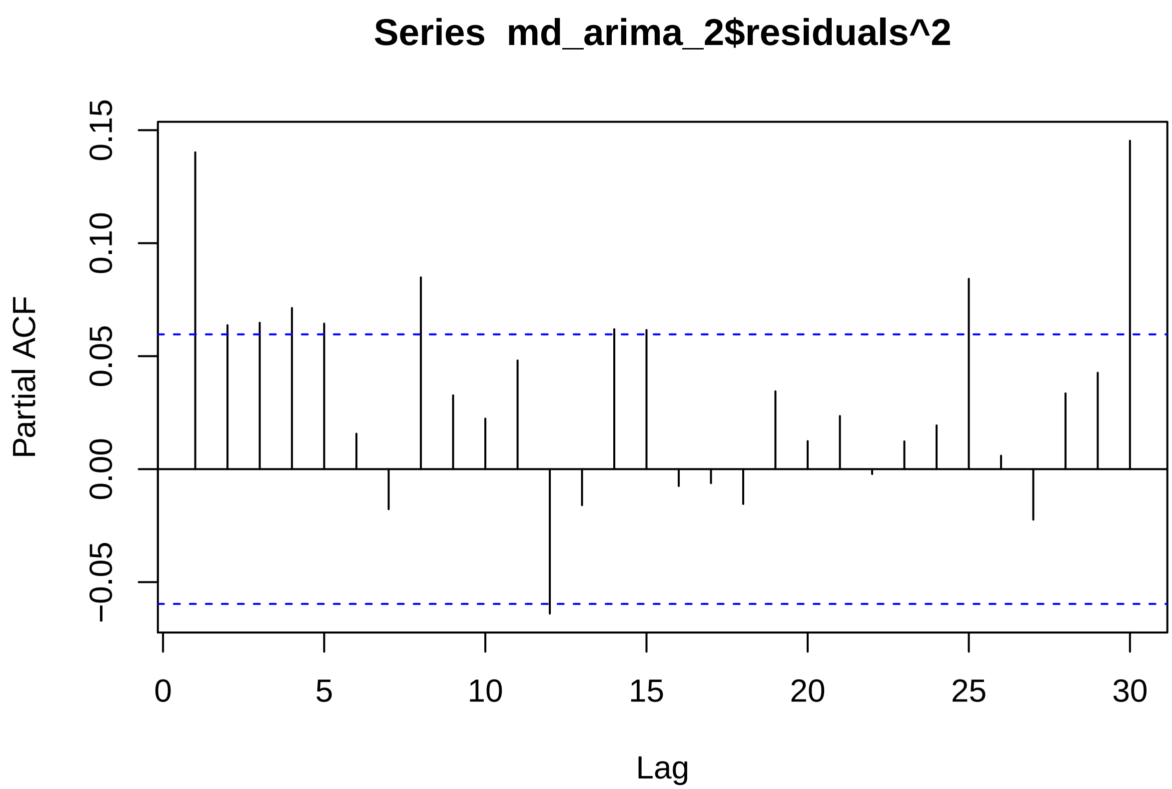
```
pred_arima = predict(md_arima_2, 263)
rmse(predicted = as.numeric(unlist(pred_arima[1])), actual = as.numeric(unlist(rate_test_rt)))
```

```
## [1] 0.002864903
```

```
Box.test(md_arima_2$residuals^2, 10, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  md_arima_2$residuals^2
## X-squared = 76.58, df = 10, p-value = 2.341e-12
```

```r
pacf(md_arima_2$residuals^2)
```




```r
md_arch_G = garchFit(~garch(1, 0), data = rate_train_rt, trace = F)
summary(md_arch_G)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(1, 0), data = rate_train_rt, trace = F)
##
## Mean and Variance Equation:
##  data ~ garch(1, 0)
## <environment: 0x000000001e313410>
##  [data = rate_train_rt]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##         mu       omega      alpha1
## 7.4600e-05  2.7171e-05  1.1065e-01
##
## Std. Errors:
##  based on Hessian
```

```
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     7.460e-05   1.643e-04    0.454 0.649719
## omega  2.717e-05   1.400e-06   19.410  < 2e-16 ***
## alpha1 1.106e-01   3.316e-02    3.337 0.000848 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  4089.615    normalized:  3.786681
##
## Description:
##  Thu Apr 30 20:18:14 2020 by user: yangm
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  226.2105  0
##  Shapiro-Wilk Test  R    W      0.9811974 1.327954e-10
##  Ljung-Box Test     R    Q(10)  5.749473  0.8358539
##  Ljung-Box Test     R    Q(15)  9.499013  0.8500155
##  Ljung-Box Test     R    Q(20)  12.16536  0.9102667
##  Ljung-Box Test     R^2  Q(10)  27.13427  0.002479927
##  Ljung-Box Test     R^2  Q(15)  41.87955  0.0002345467
##  Ljung-Box Test     R^2  Q(20)  47.57534  0.0004877931
##  LM Arch Test       R    TR^2   28.51072  0.004654617
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -7.567806 -7.553960 -7.567822 -7.562563
```

```
pred_arch_G = predict(md_arch_G, 263)
pred_arch_G
```

```
##     meanForecast   meanError standardDeviation
## 1   7.460029e-05 0.005213301       0.005213301
## 2   7.460029e-05 0.005493445       0.005493445
## 3   7.460029e-05 0.005523569       0.005523569
## 4   7.460029e-05 0.005526892       0.005526892
## 5   7.460029e-05 0.005527260       0.005527260
## 6   7.460029e-05 0.005527300       0.005527300
## 7   7.460029e-05 0.005527305       0.005527305
## 8   7.460029e-05 0.005527305       0.005527305
## 9   7.460029e-05 0.005527305       0.005527305
## 10  7.460029e-05 0.005527305       0.005527305
## 11  7.460029e-05 0.005527305       0.005527305
## 12  7.460029e-05 0.005527305       0.005527305
## 13  7.460029e-05 0.005527305       0.005527305
## 14  7.460029e-05 0.005527305       0.005527305
## 15  7.460029e-05 0.005527305       0.005527305
## 16  7.460029e-05 0.005527305       0.005527305
## 17  7.460029e-05 0.005527305       0.005527305
## 18  7.460029e-05 0.005527305       0.005527305
## 19  7.460029e-05 0.005527305       0.005527305
```

```
## 20  7.460029e-05 0.005527305       0.005527305
## 21  7.460029e-05 0.005527305       0.005527305
## 22  7.460029e-05 0.005527305       0.005527305
## 23  7.460029e-05 0.005527305       0.005527305
## 24  7.460029e-05 0.005527305       0.005527305
## 25  7.460029e-05 0.005527305       0.005527305
## 26  7.460029e-05 0.005527305       0.005527305
## 27  7.460029e-05 0.005527305       0.005527305
## 28  7.460029e-05 0.005527305       0.005527305
## 29  7.460029e-05 0.005527305       0.005527305
## 30  7.460029e-05 0.005527305       0.005527305
## 31  7.460029e-05 0.005527305       0.005527305
## 32  7.460029e-05 0.005527305       0.005527305
## 33  7.460029e-05 0.005527305       0.005527305
## 34  7.460029e-05 0.005527305       0.005527305
## 35  7.460029e-05 0.005527305       0.005527305
## 36  7.460029e-05 0.005527305       0.005527305
## 37  7.460029e-05 0.005527305       0.005527305
## 38  7.460029e-05 0.005527305       0.005527305
## 39  7.460029e-05 0.005527305       0.005527305
## 40  7.460029e-05 0.005527305       0.005527305
## 41  7.460029e-05 0.005527305       0.005527305
## 42  7.460029e-05 0.005527305       0.005527305
## 43  7.460029e-05 0.005527305       0.005527305
## 44  7.460029e-05 0.005527305       0.005527305
## 45  7.460029e-05 0.005527305       0.005527305
## 46  7.460029e-05 0.005527305       0.005527305
## 47  7.460029e-05 0.005527305       0.005527305
## 48  7.460029e-05 0.005527305       0.005527305
## 49  7.460029e-05 0.005527305       0.005527305
## 50  7.460029e-05 0.005527305       0.005527305
## 51  7.460029e-05 0.005527305       0.005527305
## 52  7.460029e-05 0.005527305       0.005527305
## 53  7.460029e-05 0.005527305       0.005527305
## 54  7.460029e-05 0.005527305       0.005527305
## 55  7.460029e-05 0.005527305       0.005527305
## 56  7.460029e-05 0.005527305       0.005527305
## 57  7.460029e-05 0.005527305       0.005527305
## 58  7.460029e-05 0.005527305       0.005527305
## 59  7.460029e-05 0.005527305       0.005527305
## 60  7.460029e-05 0.005527305       0.005527305
## 61  7.460029e-05 0.005527305       0.005527305
## 62  7.460029e-05 0.005527305       0.005527305
## 63  7.460029e-05 0.005527305       0.005527305
## 64  7.460029e-05 0.005527305       0.005527305
## 65  7.460029e-05 0.005527305       0.005527305
## 66  7.460029e-05 0.005527305       0.005527305
## 67  7.460029e-05 0.005527305       0.005527305
## 68  7.460029e-05 0.005527305       0.005527305
## 69  7.460029e-05 0.005527305       0.005527305
## 70  7.460029e-05 0.005527305       0.005527305
## 71  7.460029e-05 0.005527305       0.005527305
## 72  7.460029e-05 0.005527305       0.005527305
## 73  7.460029e-05 0.005527305       0.005527305
```

```
## 74  7.460029e-05 0.005527305       0.005527305
## 75  7.460029e-05 0.005527305       0.005527305
## 76  7.460029e-05 0.005527305       0.005527305
## 77  7.460029e-05 0.005527305       0.005527305
## 78  7.460029e-05 0.005527305       0.005527305
## 79  7.460029e-05 0.005527305       0.005527305
## 80  7.460029e-05 0.005527305       0.005527305
## 81  7.460029e-05 0.005527305       0.005527305
## 82  7.460029e-05 0.005527305       0.005527305
## 83  7.460029e-05 0.005527305       0.005527305
## 84  7.460029e-05 0.005527305       0.005527305
## 85  7.460029e-05 0.005527305       0.005527305
## 86  7.460029e-05 0.005527305       0.005527305
## 87  7.460029e-05 0.005527305       0.005527305
## 88  7.460029e-05 0.005527305       0.005527305
## 89  7.460029e-05 0.005527305       0.005527305
## 90  7.460029e-05 0.005527305       0.005527305
## 91  7.460029e-05 0.005527305       0.005527305
## 92  7.460029e-05 0.005527305       0.005527305
## 93  7.460029e-05 0.005527305       0.005527305
## 94  7.460029e-05 0.005527305       0.005527305
## 95  7.460029e-05 0.005527305       0.005527305
## 96  7.460029e-05 0.005527305       0.005527305
## 97  7.460029e-05 0.005527305       0.005527305
## 98  7.460029e-05 0.005527305       0.005527305
## 99  7.460029e-05 0.005527305       0.005527305
## 100 7.460029e-05 0.005527305       0.005527305
## 101 7.460029e-05 0.005527305       0.005527305
## 102 7.460029e-05 0.005527305       0.005527305
## 103 7.460029e-05 0.005527305       0.005527305
## 104 7.460029e-05 0.005527305       0.005527305
## 105 7.460029e-05 0.005527305       0.005527305
## 106 7.460029e-05 0.005527305       0.005527305
## 107 7.460029e-05 0.005527305       0.005527305
## 108 7.460029e-05 0.005527305       0.005527305
## 109 7.460029e-05 0.005527305       0.005527305
## 110 7.460029e-05 0.005527305       0.005527305
## 111 7.460029e-05 0.005527305       0.005527305
## 112 7.460029e-05 0.005527305       0.005527305
## 113 7.460029e-05 0.005527305       0.005527305
## 114 7.460029e-05 0.005527305       0.005527305
## 115 7.460029e-05 0.005527305       0.005527305
## 116 7.460029e-05 0.005527305       0.005527305
## 117 7.460029e-05 0.005527305       0.005527305
## 118 7.460029e-05 0.005527305       0.005527305
## 119 7.460029e-05 0.005527305       0.005527305
## 120 7.460029e-05 0.005527305       0.005527305
## 121 7.460029e-05 0.005527305       0.005527305
## 122 7.460029e-05 0.005527305       0.005527305
## 123 7.460029e-05 0.005527305       0.005527305
## 124 7.460029e-05 0.005527305       0.005527305
## 125 7.460029e-05 0.005527305       0.005527305
## 126 7.460029e-05 0.005527305       0.005527305
## 127 7.460029e-05 0.005527305       0.005527305
```

```
## 128 7.460029e-05 0.005527305       0.005527305
## 129 7.460029e-05 0.005527305       0.005527305
## 130 7.460029e-05 0.005527305       0.005527305
## 131 7.460029e-05 0.005527305       0.005527305
## 132 7.460029e-05 0.005527305       0.005527305
## 133 7.460029e-05 0.005527305       0.005527305
## 134 7.460029e-05 0.005527305       0.005527305
## 135 7.460029e-05 0.005527305       0.005527305
## 136 7.460029e-05 0.005527305       0.005527305
## 137 7.460029e-05 0.005527305       0.005527305
## 138 7.460029e-05 0.005527305       0.005527305
## 139 7.460029e-05 0.005527305       0.005527305
## 140 7.460029e-05 0.005527305       0.005527305
## 141 7.460029e-05 0.005527305       0.005527305
## 142 7.460029e-05 0.005527305       0.005527305
## 143 7.460029e-05 0.005527305       0.005527305
## 144 7.460029e-05 0.005527305       0.005527305
## 145 7.460029e-05 0.005527305       0.005527305
## 146 7.460029e-05 0.005527305       0.005527305
## 147 7.460029e-05 0.005527305       0.005527305
## 148 7.460029e-05 0.005527305       0.005527305
## 149 7.460029e-05 0.005527305       0.005527305
## 150 7.460029e-05 0.005527305       0.005527305
## 151 7.460029e-05 0.005527305       0.005527305
## 152 7.460029e-05 0.005527305       0.005527305
## 153 7.460029e-05 0.005527305       0.005527305
## 154 7.460029e-05 0.005527305       0.005527305
## 155 7.460029e-05 0.005527305       0.005527305
## 156 7.460029e-05 0.005527305       0.005527305
## 157 7.460029e-05 0.005527305       0.005527305
## 158 7.460029e-05 0.005527305       0.005527305
## 159 7.460029e-05 0.005527305       0.005527305
## 160 7.460029e-05 0.005527305       0.005527305
## 161 7.460029e-05 0.005527305       0.005527305
## 162 7.460029e-05 0.005527305       0.005527305
## 163 7.460029e-05 0.005527305       0.005527305
## 164 7.460029e-05 0.005527305       0.005527305
## 165 7.460029e-05 0.005527305       0.005527305
## 166 7.460029e-05 0.005527305       0.005527305
## 167 7.460029e-05 0.005527305       0.005527305
## 168 7.460029e-05 0.005527305       0.005527305
## 169 7.460029e-05 0.005527305       0.005527305
## 170 7.460029e-05 0.005527305       0.005527305
## 171 7.460029e-05 0.005527305       0.005527305
## 172 7.460029e-05 0.005527305       0.005527305
## 173 7.460029e-05 0.005527305       0.005527305
## 174 7.460029e-05 0.005527305       0.005527305
## 175 7.460029e-05 0.005527305       0.005527305
## 176 7.460029e-05 0.005527305       0.005527305
## 177 7.460029e-05 0.005527305       0.005527305
## 178 7.460029e-05 0.005527305       0.005527305
## 179 7.460029e-05 0.005527305       0.005527305
## 180 7.460029e-05 0.005527305       0.005527305
## 181 7.460029e-05 0.005527305       0.005527305
```

```
## 182 7.460029e-05 0.005527305        0.005527305
## 183 7.460029e-05 0.005527305        0.005527305
## 184 7.460029e-05 0.005527305        0.005527305
## 185 7.460029e-05 0.005527305        0.005527305
## 186 7.460029e-05 0.005527305        0.005527305
## 187 7.460029e-05 0.005527305        0.005527305
## 188 7.460029e-05 0.005527305        0.005527305
## 189 7.460029e-05 0.005527305        0.005527305
## 190 7.460029e-05 0.005527305        0.005527305
## 191 7.460029e-05 0.005527305        0.005527305
## 192 7.460029e-05 0.005527305        0.005527305
## 193 7.460029e-05 0.005527305        0.005527305
## 194 7.460029e-05 0.005527305        0.005527305
## 195 7.460029e-05 0.005527305        0.005527305
## 196 7.460029e-05 0.005527305        0.005527305
## 197 7.460029e-05 0.005527305        0.005527305
## 198 7.460029e-05 0.005527305        0.005527305
## 199 7.460029e-05 0.005527305        0.005527305
## 200 7.460029e-05 0.005527305        0.005527305
## 201 7.460029e-05 0.005527305        0.005527305
## 202 7.460029e-05 0.005527305        0.005527305
## 203 7.460029e-05 0.005527305        0.005527305
## 204 7.460029e-05 0.005527305        0.005527305
## 205 7.460029e-05 0.005527305        0.005527305
## 206 7.460029e-05 0.005527305        0.005527305
## 207 7.460029e-05 0.005527305        0.005527305
## 208 7.460029e-05 0.005527305        0.005527305
## 209 7.460029e-05 0.005527305        0.005527305
## 210 7.460029e-05 0.005527305        0.005527305
## 211 7.460029e-05 0.005527305        0.005527305
## 212 7.460029e-05 0.005527305        0.005527305
## 213 7.460029e-05 0.005527305        0.005527305
## 214 7.460029e-05 0.005527305        0.005527305
## 215 7.460029e-05 0.005527305        0.005527305
## 216 7.460029e-05 0.005527305        0.005527305
## 217 7.460029e-05 0.005527305        0.005527305
## 218 7.460029e-05 0.005527305        0.005527305
## 219 7.460029e-05 0.005527305        0.005527305
## 220 7.460029e-05 0.005527305        0.005527305
## 221 7.460029e-05 0.005527305        0.005527305
## 222 7.460029e-05 0.005527305        0.005527305
## 223 7.460029e-05 0.005527305        0.005527305
## 224 7.460029e-05 0.005527305        0.005527305
## 225 7.460029e-05 0.005527305        0.005527305
## 226 7.460029e-05 0.005527305        0.005527305
## 227 7.460029e-05 0.005527305        0.005527305
## 228 7.460029e-05 0.005527305        0.005527305
## 229 7.460029e-05 0.005527305        0.005527305
## 230 7.460029e-05 0.005527305        0.005527305
## 231 7.460029e-05 0.005527305        0.005527305
## 232 7.460029e-05 0.005527305        0.005527305
## 233 7.460029e-05 0.005527305        0.005527305
## 234 7.460029e-05 0.005527305        0.005527305
## 235 7.460029e-05 0.005527305        0.005527305
```

```
## 236 7.460029e-05 0.005527305       0.005527305
## 237 7.460029e-05 0.005527305       0.005527305
## 238 7.460029e-05 0.005527305       0.005527305
## 239 7.460029e-05 0.005527305       0.005527305
## 240 7.460029e-05 0.005527305       0.005527305
## 241 7.460029e-05 0.005527305       0.005527305
## 242 7.460029e-05 0.005527305       0.005527305
## 243 7.460029e-05 0.005527305       0.005527305
## 244 7.460029e-05 0.005527305       0.005527305
## 245 7.460029e-05 0.005527305       0.005527305
## 246 7.460029e-05 0.005527305       0.005527305
## 247 7.460029e-05 0.005527305       0.005527305
## 248 7.460029e-05 0.005527305       0.005527305
## 249 7.460029e-05 0.005527305       0.005527305
## 250 7.460029e-05 0.005527305       0.005527305
## 251 7.460029e-05 0.005527305       0.005527305
## 252 7.460029e-05 0.005527305       0.005527305
## 253 7.460029e-05 0.005527305       0.005527305
## 254 7.460029e-05 0.005527305       0.005527305
## 255 7.460029e-05 0.005527305       0.005527305
## 256 7.460029e-05 0.005527305       0.005527305
## 257 7.460029e-05 0.005527305       0.005527305
## 258 7.460029e-05 0.005527305       0.005527305
## 259 7.460029e-05 0.005527305       0.005527305
## 260 7.460029e-05 0.005527305       0.005527305
## 261 7.460029e-05 0.005527305       0.005527305
## 262 7.460029e-05 0.005527305       0.005527305
## 263 7.460029e-05 0.005527305       0.005527305
```

```r
rmse(predicted = as.numeric(unlist(pred_arch_G[1])), actual = as.numeric(unlist(rate_test_rt)))
```

```
## [1] 0.002861563
```

```r
md_arch_t = garchFit(~garch(1, 0), data = rate_train_rt, cond.dist = c("std"), trace = F)
summary(md_arch_t)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(1, 0), data = rate_train_rt, cond.dist = c("std"),
##     trace = F)
##
## Mean and Variance Equation:
##  data ~ garch(1, 0)
## <environment: 0x00000000212b82e0>
##  [data = rate_train_rt]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##         mu       omega      alpha1       shape
## 8.7910e-05  2.6882e-05  1.2063e-01  6.3475e+00
```

```
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     8.791e-05   1.532e-04    0.574  0.56598
## omega  2.688e-05   1.913e-06   14.054  < 2e-16 ***
## alpha1 1.206e-01   4.043e-02    2.983  0.00285 **
## shape  6.347e+00   1.142e+00    5.558 2.72e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  4118.921    normalized:  3.813816
##
## Description:
##  Thu Apr 30 20:18:14 2020 by user: yangm
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  229.3519  0
##  Shapiro-Wilk Test  R    W      0.9810736 1.194785e-10
##  Ljung-Box Test     R    Q(10)  5.724152  0.8378819
##  Ljung-Box Test     R    Q(15)  9.467828  0.8518147
##  Ljung-Box Test     R    Q(20)  12.14253  0.9110828
##  Ljung-Box Test     R^2  Q(10)  26.04315  0.003682578
##  Ljung-Box Test     R^2  Q(15)  40.35298  0.0004010477
##  Ljung-Box Test     R^2  Q(20)  46.04301  0.0007951392
##  LM Arch Test       R    TR^2   27.84604  0.005826142
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -7.620225 -7.601763 -7.620252 -7.613234
```

```r
pred_arch_t = predict(md_arch_t, 263)
pred_arch_t[1]
```

```
##     meanForecast
## 1   8.791044e-05
## 2   8.791044e-05
## 3   8.791044e-05
## 4   8.791044e-05
## 5   8.791044e-05
## 6   8.791044e-05
## 7   8.791044e-05
## 8   8.791044e-05
## 9   8.791044e-05
## 10  8.791044e-05
## 11  8.791044e-05
## 12  8.791044e-05
## 13  8.791044e-05
## 14  8.791044e-05
## 15  8.791044e-05
```

```
## 16  8.791044e-05
## 17  8.791044e-05
## 18  8.791044e-05
## 19  8.791044e-05
## 20  8.791044e-05
## 21  8.791044e-05
## 22  8.791044e-05
## 23  8.791044e-05
## 24  8.791044e-05
## 25  8.791044e-05
## 26  8.791044e-05
## 27  8.791044e-05
## 28  8.791044e-05
## 29  8.791044e-05
## 30  8.791044e-05
## 31  8.791044e-05
## 32  8.791044e-05
## 33  8.791044e-05
## 34  8.791044e-05
## 35  8.791044e-05
## 36  8.791044e-05
## 37  8.791044e-05
## 38  8.791044e-05
## 39  8.791044e-05
## 40  8.791044e-05
## 41  8.791044e-05
## 42  8.791044e-05
## 43  8.791044e-05
## 44  8.791044e-05
## 45  8.791044e-05
## 46  8.791044e-05
## 47  8.791044e-05
## 48  8.791044e-05
## 49  8.791044e-05
## 50  8.791044e-05
## 51  8.791044e-05
## 52  8.791044e-05
## 53  8.791044e-05
## 54  8.791044e-05
## 55  8.791044e-05
## 56  8.791044e-05
## 57  8.791044e-05
## 58  8.791044e-05
## 59  8.791044e-05
## 60  8.791044e-05
## 61  8.791044e-05
## 62  8.791044e-05
## 63  8.791044e-05
## 64  8.791044e-05
## 65  8.791044e-05
## 66  8.791044e-05
## 67  8.791044e-05
## 68  8.791044e-05
## 69  8.791044e-05
```

```
## 70  8.791044e-05
## 71  8.791044e-05
## 72  8.791044e-05
## 73  8.791044e-05
## 74  8.791044e-05
## 75  8.791044e-05
## 76  8.791044e-05
## 77  8.791044e-05
## 78  8.791044e-05
## 79  8.791044e-05
## 80  8.791044e-05
## 81  8.791044e-05
## 82  8.791044e-05
## 83  8.791044e-05
## 84  8.791044e-05
## 85  8.791044e-05
## 86  8.791044e-05
## 87  8.791044e-05
## 88  8.791044e-05
## 89  8.791044e-05
## 90  8.791044e-05
## 91  8.791044e-05
## 92  8.791044e-05
## 93  8.791044e-05
## 94  8.791044e-05
## 95  8.791044e-05
## 96  8.791044e-05
## 97  8.791044e-05
## 98  8.791044e-05
## 99  8.791044e-05
## 100 8.791044e-05
## 101 8.791044e-05
## 102 8.791044e-05
## 103 8.791044e-05
## 104 8.791044e-05
## 105 8.791044e-05
## 106 8.791044e-05
## 107 8.791044e-05
## 108 8.791044e-05
## 109 8.791044e-05
## 110 8.791044e-05
## 111 8.791044e-05
## 112 8.791044e-05
## 113 8.791044e-05
## 114 8.791044e-05
## 115 8.791044e-05
## 116 8.791044e-05
## 117 8.791044e-05
## 118 8.791044e-05
## 119 8.791044e-05
## 120 8.791044e-05
## 121 8.791044e-05
## 122 8.791044e-05
## 123 8.791044e-05
```

```
## 124 8.791044e-05
## 125 8.791044e-05
## 126 8.791044e-05
## 127 8.791044e-05
## 128 8.791044e-05
## 129 8.791044e-05
## 130 8.791044e-05
## 131 8.791044e-05
## 132 8.791044e-05
## 133 8.791044e-05
## 134 8.791044e-05
## 135 8.791044e-05
## 136 8.791044e-05
## 137 8.791044e-05
## 138 8.791044e-05
## 139 8.791044e-05
## 140 8.791044e-05
## 141 8.791044e-05
## 142 8.791044e-05
## 143 8.791044e-05
## 144 8.791044e-05
## 145 8.791044e-05
## 146 8.791044e-05
## 147 8.791044e-05
## 148 8.791044e-05
## 149 8.791044e-05
## 150 8.791044e-05
## 151 8.791044e-05
## 152 8.791044e-05
## 153 8.791044e-05
## 154 8.791044e-05
## 155 8.791044e-05
## 156 8.791044e-05
## 157 8.791044e-05
## 158 8.791044e-05
## 159 8.791044e-05
## 160 8.791044e-05
## 161 8.791044e-05
## 162 8.791044e-05
## 163 8.791044e-05
## 164 8.791044e-05
## 165 8.791044e-05
## 166 8.791044e-05
## 167 8.791044e-05
## 168 8.791044e-05
## 169 8.791044e-05
## 170 8.791044e-05
## 171 8.791044e-05
## 172 8.791044e-05
## 173 8.791044e-05
## 174 8.791044e-05
## 175 8.791044e-05
## 176 8.791044e-05
## 177 8.791044e-05
```

```
## 178 8.791044e-05
## 179 8.791044e-05
## 180 8.791044e-05
## 181 8.791044e-05
## 182 8.791044e-05
## 183 8.791044e-05
## 184 8.791044e-05
## 185 8.791044e-05
## 186 8.791044e-05
## 187 8.791044e-05
## 188 8.791044e-05
## 189 8.791044e-05
## 190 8.791044e-05
## 191 8.791044e-05
## 192 8.791044e-05
## 193 8.791044e-05
## 194 8.791044e-05
## 195 8.791044e-05
## 196 8.791044e-05
## 197 8.791044e-05
## 198 8.791044e-05
## 199 8.791044e-05
## 200 8.791044e-05
## 201 8.791044e-05
## 202 8.791044e-05
## 203 8.791044e-05
## 204 8.791044e-05
## 205 8.791044e-05
## 206 8.791044e-05
## 207 8.791044e-05
## 208 8.791044e-05
## 209 8.791044e-05
## 210 8.791044e-05
## 211 8.791044e-05
## 212 8.791044e-05
## 213 8.791044e-05
## 214 8.791044e-05
## 215 8.791044e-05
## 216 8.791044e-05
## 217 8.791044e-05
## 218 8.791044e-05
## 219 8.791044e-05
## 220 8.791044e-05
## 221 8.791044e-05
## 222 8.791044e-05
## 223 8.791044e-05
## 224 8.791044e-05
## 225 8.791044e-05
## 226 8.791044e-05
## 227 8.791044e-05
## 228 8.791044e-05
## 229 8.791044e-05
## 230 8.791044e-05
## 231 8.791044e-05
```

```
## 232 8.791044e-05
## 233 8.791044e-05
## 234 8.791044e-05
## 235 8.791044e-05
## 236 8.791044e-05
## 237 8.791044e-05
## 238 8.791044e-05
## 239 8.791044e-05
## 240 8.791044e-05
## 241 8.791044e-05
## 242 8.791044e-05
## 243 8.791044e-05
## 244 8.791044e-05
## 245 8.791044e-05
## 246 8.791044e-05
## 247 8.791044e-05
## 248 8.791044e-05
## 249 8.791044e-05
## 250 8.791044e-05
## 251 8.791044e-05
## 252 8.791044e-05
## 253 8.791044e-05
## 254 8.791044e-05
## 255 8.791044e-05
## 256 8.791044e-05
## 257 8.791044e-05
## 258 8.791044e-05
## 259 8.791044e-05
## 260 8.791044e-05
## 261 8.791044e-05
## 262 8.791044e-05
## 263 8.791044e-05
```

```r
rmse(predicted = as.numeric(unlist(pred_arch_t[1])), actual = as.numeric(unlist(rate_test_rt)))
```

```
## [1] 0.002861171
```

```r
md_garch_G = garchFit(~garch(1, 1), data = rate_train_rt, trace = F)
summary(md_garch_G)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(1, 1), data = rate_train_rt, trace = F)
##
## Mean and Variance Equation:
##  data ~ garch(1, 1)
## <environment: 0x0000000020d83688>
##  [data = rate_train_rt]
##
## Conditional Distribution:
##  norm
##
```

```
## Coefficient(s):
##         mu       omega      alpha1       beta1
## 1.6178e-05  5.1841e-08  1.3089e-02  9.8430e-01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     1.618e-05   1.558e-04    0.104    0.917
## omega  5.184e-08   4.734e-08    1.095    0.273
## alpha1 1.309e-02   3.037e-03    4.310 1.64e-05 ***
## beta1  9.843e-01   3.089e-03  318.694  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  4121.965    normalized:  3.816634
##
## Description:
##  Thu Apr 30 20:18:14 2020 by user: yangm
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  77.34754  0
##  Shapiro-Wilk Test  R    W      0.9903362 1.549384e-06
##  Ljung-Box Test     R    Q(10)  3.616231  0.9630037
##  Ljung-Box Test     R    Q(15)  7.300761  0.9487601
##  Ljung-Box Test     R    Q(20)  8.588938  0.9871979
##  Ljung-Box Test     R^2  Q(10)  8.738132  0.5571231
##  Ljung-Box Test     R^2  Q(15)  20.51427  0.1530793
##  Ljung-Box Test     R^2  Q(20)  23.81096  0.2507476
##  LM Arch Test       R    TR^2   16.09945  0.1867234
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -7.625860 -7.607398 -7.625888 -7.618870
```

```r
pred_garch_G = predict(md_garch_G, 263)
rmse(predicted = as.numeric(unlist(pred_garch_G[1])), actual = as.numeric(unlist(rate_test_rt)))
```

```
## [1] 0.00286401
```

```r
md_garch_t = garchFit(~garch(1, 1), data = rate_train_rt, cond.dist = c("std"), trace = F)
summary(md_garch_t)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(1, 1), data = rate_train_rt, cond.dist = c("std"),
##     trace = F)
##
```

```
## Mean and Variance Equation:
##  data ~ garch(1, 1)
## <environment: 0x000000002521dbf0>
##  [data = rate_train_rt]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##         mu       omega      alpha1       beta1       shape
## 4.6927e-05  1.5507e-07  1.7696e-02  9.7603e-01  8.4884e+00
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     4.693e-05   1.502e-04    0.312  0.75472
## omega  1.551e-07   1.010e-07    1.536  0.12453
## alpha1 1.770e-02   5.692e-03    3.109  0.00188 **
## beta1  9.760e-01   7.186e-03  135.816  < 2e-16 ***
## shape  8.488e+00   1.897e+00    4.475 7.66e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  4137.931    normalized:  3.831418
##
## Description:
##  Thu Apr 30 20:18:14 2020 by user: yangm
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  95.93217  0
##  Shapiro-Wilk Test  R    W      0.9890606 3.310632e-07
##  Ljung-Box Test     R    Q(10)  3.529829  0.9660758
##  Ljung-Box Test     R    Q(15)  7.177869  0.9525247
##  Ljung-Box Test     R    Q(20)  8.399585  0.9888765
##  Ljung-Box Test     R^2  Q(10)  7.80879   0.6475078
##  Ljung-Box Test     R^2  Q(15)  18.94319  0.2163198
##  Ljung-Box Test     R^2  Q(20)  22.23991  0.3276358
##  LM Arch Test       R    TR^2   13.59259  0.3274774
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -7.653577 -7.630499 -7.653619 -7.644838
```

```r
pred_garch_t = predict(md_garch_t, 263)
rmse(predicted = as.numeric(unlist(pred_garch_t[1])), actual = as.numeric(unlist(rate_test_rt)))
```

```
## [1] 0.002862574
```

```r
md_tgarch = garchFit(~aparch(1,1), rate_train_rt, delta = 2, include.delta = F, trace = F)
summary(md_tgarch)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~aparch(1, 1), data = rate_train_rt, delta = 2,
##     include.delta = F, trace = F)
##
## Mean and Variance Equation:
##  data ~ aparch(1, 1)
## <environment: 0x0000000020ca1720>
##  [data = rate_train_rt]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##          mu        omega       alpha1       gamma1        beta1
##  4.1302e-05   5.7739e-08   9.1342e-03  -3.6299e-01   9.8701e-01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error  t value Pr(>|t|)
## mu      4.130e-05   1.564e-04    0.264 0.791692
## omega   5.774e-08   3.882e-08    1.487 0.136937
## alpha1  9.134e-03   2.397e-03    3.811 0.000138 ***
## gamma1 -3.630e-01   1.984e-01   -1.829 0.067343 .
## beta1   9.870e-01   2.136e-03  462.073  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  4124.41    normalized:  3.818898
##
## Description:
##  Thu Apr 30 20:18:14 2020 by user: yangm
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  67.8008   1.887379e-15
##  Shapiro-Wilk Test  R    W      0.9911581 4.428322e-06
##  Ljung-Box Test     R    Q(10)  3.45741   0.9685247
##  Ljung-Box Test     R    Q(15)  7.230621  0.9509312
##  Ljung-Box Test     R    Q(20)  8.564348  0.987426
##  Ljung-Box Test     R^2  Q(10)  7.695832  0.6585199
##  Ljung-Box Test     R^2  Q(15)  20.1506   0.1662314
##  Ljung-Box Test     R^2  Q(20)  23.00657  0.2884719
##  LM Arch Test       R    TR^2   15.07848  0.2371728
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
```

```
## -7.628536 -7.605459 -7.628579 -7.619798
```

```r
pred_tgarch = predict(md_tgarch, 263)
rmse(predicted = as.numeric(unlist(pred_tgarch[1])), actual = as.numeric(unlist(rate_test_rt)))
```

```
## [1] 0.002862812
```

VAR & VEC

```r
#Apply granger test for determining whether one time series is useful in forecasting currency rate.
grangertest(M1_US_rt, RATE_rt, order = 1, na.action = na.omit)
```

```
## Granger causality test
##
## Model 1: RATE_rt ~ Lags(RATE_rt, 1:1) + Lags(M1_US_rt, 1:1)
## Model 2: RATE_rt ~ Lags(RATE_rt, 1:1)
##   Res.Df Df      F Pr(>F)
## 1     58
## 2     59 -1 0.5625 0.4563
```

```r
grangertest(M1_EU_rt, RATE_rt, order = 1, na.action = na.omit)
```

```
## Granger causality test
##
## Model 1: RATE_rt ~ Lags(RATE_rt, 1:1) + Lags(M1_EU_rt, 1:1)
## Model 2: RATE_rt ~ Lags(RATE_rt, 1:1)
##   Res.Df Df      F Pr(>F)
## 1     58
## 2     59 -1 0.6713  0.416
```

```r
grangertest(GDP_US_rt, RATE_rt, order = 1, na.action = na.omit)
```

```
## Granger causality test
##
## Model 1: RATE_rt ~ Lags(RATE_rt, 1:1) + Lags(GDP_US_rt, 1:1)
## Model 2: RATE_rt ~ Lags(RATE_rt, 1:1)
##   Res.Df Df      F Pr(>F)
## 1     58
## 2     59 -1 1.0581 0.3079
```

```r
grangertest(GDP_EU_rt_dif, RATE_rt, order = 1, na.action = na.omit)
```

```
## Granger causality test
##
## Model 1: RATE_rt ~ Lags(RATE_rt, 1:1) + Lags(GDP_EU_rt_dif, 1:1)
## Model 2: RATE_rt ~ Lags(RATE_rt, 1:1)
##   Res.Df Df      F  Pr(>F)
## 1     58
## 2     59 -1 4.9294 0.03033 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
grangertest(CPI_US_rt, RATE_rt, order = 1, na.action = na.omit)
```

```
## Granger causality test
##
## Model 1: RATE_rt ~ Lags(RATE_rt, 1:1) + Lags(CPI_US_rt, 1:1)
```

```
## Model 2: RATE_rt ~ Lags(RATE_rt, 1:1)
##   Res.Df Df      F Pr(>F)
## 1     58
## 2     59 -1 1.0585 0.3078
```

```r
grangertest(CPI_EU_rt, RATE_rt, order = 1, na.action = na.omit)
```

```
## Granger causality test
##
## Model 1: RATE_rt ~ Lags(RATE_rt, 1:1) + Lags(CPI_EU_rt, 1:1)
## Model 2: RATE_rt ~ Lags(RATE_rt, 1:1)
##   Res.Df Df      F Pr(>F)
## 1     58
## 2     59 -1 0.4625 0.4992
```

```r
grangertest(XAU_USD_rt, RATE_rt, order = 1, na.action = na.omit)
```

```
## Granger causality test
##
## Model 1: RATE_rt ~ Lags(RATE_rt, 1:1) + Lags(XAU_USD_rt, 1:1)
## Model 2: RATE_rt ~ Lags(RATE_rt, 1:1)
##   Res.Df Df      F Pr(>F)
## 1     58
## 2     59 -1 0.4613 0.4997
```

```r
grangertest(WTI_USD_rt, RATE_rt, order = 1, na.action = na.omit)
```

```
## Granger causality test
##
## Model 1: RATE_rt ~ Lags(RATE_rt, 1:1) + Lags(WTI_USD_rt, 1:1)
## Model 2: RATE_rt ~ Lags(RATE_rt, 1:1)
##   Res.Df Df     F Pr(>F)
## 1     58
## 2     59 -1 0.957  0.332
```

From the result, only GDP_EU_rt is significant, while it is a first order differential. So we tried cointegration test.

```r
currency.rate = as.numeric(RATE_rt)
#M1_US = as.numeric(M1_US_rt)
#M1_EU = as.numeric(M1_EU_rt)
#GDP_US = as.numeric(GDP_US_rt)
GDP_EU = as.numeric(GDP_EU_rt_dif)
#CPI_US = as.numeric(CPI_US_rt)
#CPI_EU = as.numeric(CPI_EU_rt)
#XAU_USD = as.numeric(XAU_USD_rt)
#WTI_USD = as.numeric(WTI_USD_rt)
```

```r
#md_lm = lm(currency.rate ~ M1_US + M1_EU + GDP_US + GDP_EU + CPI_US + CPI_EU + XAU_USD + WTI_USD)
md_lm = lm(currency.rate ~ GDP_EU_rt, data_rt_train)
summary(md_lm)
```

```
##
## Call:
## lm(formula = currency.rate ~ GDP_EU_rt, data = data_rt_train)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
```

```
## -0.031104 -0.008666 -0.000061  0.004642  0.049526
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0008595  0.0020683   0.416    0.679
## GDP_EU_rt   -4.8953780  3.2925461  -1.487    0.142
##
## Residual standard error: 0.01622 on 60 degrees of freedom
## Multiple R-squared:  0.03553,    Adjusted R-squared:  0.01946
## F-statistic: 2.211 on 1 and 60 DF,  p-value: 0.1423
```

```r
dwtest(md_lm)
```

```
##
##  Durbin-Watson test
##
## data:  md_lm
## DW = 1.5607, p-value = 0.03119
## alternative hypothesis: true autocorrelation is greater than 0
```

```r
res = residuals(md_lm)
ur.df(res, type = "none", selectlags = "AIC")
```

```
##
## ###############################################################
## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #
## ###############################################################
##
## The value of the test statistic is: -5.6952
```

```r
x = cbind(GDP_EU_rt, RATE_rt)[1:50]
names(x) = c("GDP_EU", "RATE")
head(x)
```

```
##                GDP_EU          RATE
## Jan 2015 0.0000000000  0.0000000000
## Feb 2015 0.0004064408  0.0238288969
## Mar 2015 0.0003430773  0.0487057221
## Apr 2015 0.0002797828  0.0008660821
## May 2015 0.0002325859 -0.0303948080
## Jun 2015 0.0002054699 -0.0070281124
```

```r
vecm = ca.jo(x, K = 2, ecdet = "const")
summary(vecm)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant in cointegra
##
## Eigenvalues (lambda):
## [1] 5.430146e-01 1.116981e-01 5.204170e-17
##
## Values of teststatistic and critical values of test:
##
```

```
##           test 10pct  5pct  1pct
## r <= 1 |  5.69  7.52  9.24 12.97
## r = 0  | 37.59 13.75 15.67 20.20
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##               GDP_EU.l2       RATE.l2      constant
## GDP_EU.l2  1.0000000000  1.0000000000  1.0000000000
## RATE.l2    0.0720521911 -0.0135857305 -0.0044551604
## constant  -0.0001026826  0.0002173345 -0.0009482182
##
## Weights W:
## (This is the loading matrix)
##
##             GDP_EU.l2     RATE.l2      constant
## GDP_EU.d  -0.01118603 -0.03512347  3.263710e-17
## RATE.d   -14.86184372  3.24985614 -1.146792e-14
```

To construct the VAR model for the selected endogenous variables, the co-integration test is needed. Apply durbin-watson test to linear model. As the value of p-value is small enough, indicating that the residual sequence is not independent at the significance level of 5% and has autocorrelation. According to the stationariness test results of the residual series, the null hypothesis that the residual series has unit root is rejected at the significance level of 5%, that is, the residual series is stable, indicating that there is a co-integration relationship between currency rate and GDP of EU.
From the result of Johansen procedure, we know the rank of the co-integration vector is 1.
Then used cajorls() to estimate the coefficient matrix of the VEC model. Then VEC model is converted to horizontal VAR model to predict currency rate.

```r
md_vec = cajorls(vecm, r=1)
md_vec
```

```
## $rlm
##
## Call:
## lm(formula = substitute(form1), data = data.mat)
##
## Coefficients:
##             GDP_EU.d    RATE.d
## ect1        -1.119e-02  -1.486e+01
## GDP_EU.dl1   7.195e-01   5.522e+01
## RATE.dl1    -6.986e-04  -8.953e-01
##
##
## $beta
##                    ect1
## GDP_EU.l2  1.0000000000
## RATE.l2    0.0720521911
## constant  -0.0001026826
```

```r
md_var = vec2var(vecm, r = 1)
md_var
```

```
##
## Coefficient matrix of lagged endogenous variables:
##
```

```
## A1:
##        GDP_EU.l1       RATE.l1
## GDP_EU  1.719549 -0.0006985824
## RATE   55.223957  0.1046873174
##
##
## A2:
##          GDP_EU.l2       RATE.l2
## GDP_EU  -0.7307353 -0.0001073955
## RATE   -70.0858006 -0.1755157216
##
##
## Coefficient matrix of deterministic regressor(s).
##
##            constant
## GDP_EU 1.148610e-06
## RATE   1.526052e-03
```

```
pred_var = predict(md_var, 12)
rmse(predicted = pred_var$fcst$RATE[1], actual = as.numeric(unlist(data_rt_test$RATE)))
```

```
## [1] 0.006963041
```

```
#irf(md_var)
```

DCC-GARCH

```
#Calculate the maximum, minimum, median, skewness, kurtosis and extremum
data_outline = function(x){
  m = mean(x)
  d=max(x)
  xd=min(x)
  me = median(x)
  s = sd(x)
  kur=kurtosis(x)
  ske=skewness(x)
  R = max(x)-min(x)
  data.frame(Mean=m, Median=me, max=d,min=xd,std_dev=s, Skewness=ske, Kurtosis=kur, R=R)
}
for (i in  1:3){print(data_outline(data_D_rt[,i]))}
```

```
##          Mean       Median        max         min     std_dev   Skewness
## 1 0.000218366 0.0002179542 0.04692827 -0.03546172 0.007752035 0.09691199
##   Kurtosis          R
## 1 2.613654 0.08238999
##           Mean       Median       max         min    std_dev  Skewness Kurtosis
## 1 5.168222e-05 0.0003836707 0.1171551 -0.08648298 0.02193947 0.1926181 2.554549
##           R
## 1 0.2036381
##           Mean Median        max         min     std_dev   Skewness Kurtosis
## 1 7.937433e-05      0 0.02430363 -0.02995268 0.005129614 -0.1159179 2.713915
##            R
## 1 0.05425631
```

```r
ArchTest(M1_US1_rt)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  M1_US1_rt
## Chi-squared = 18.056, df = 12, p-value = 0.114
```

```r
ArchTest(CPI_US1_rt)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  CPI_US1_rt
## Chi-squared = 12.432, df = 12, p-value = 0.4117
```

```r
ArchTest(RATE1_rt)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  RATE1_rt
## Chi-squared = 13.725, df = 12, p-value = 0.3186
```

```r
ArchTest(XAU_USD1_rt)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  XAU_USD1_rt
## Chi-squared = 32.052, df = 12, p-value = 0.001358
```

```r
ArchTest(WTI_USD1_rt)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  WTI_USD1_rt
## Chi-squared = 207.04, df = 12, p-value < 2.2e-16
```

```r
ArchTest(RATE2_rt)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  RATE2_rt
## Chi-squared = 91.377, df = 12, p-value = 2.671e-14
```

There is no ARCH effect of the monthly time series data, we decided to use the daily data, because ARCH effect shows in all three time series.

```r
#Calculate the correlation coefficient matrix
corre = cor(data_D_rt)
corre
```

```
##              XAU         WTI       RATE
## XAU   1.00000000  0.01908568 -0.2776450
## WTI   0.01908568  1.00000000 -0.0317455
## RATE -0.27764499 -0.03174550  1.0000000
```

```r
#1:   Conditional Mean (vs Realized Returns)
#2:   Conditional Sigma (vs Realized Absolute Returns)
#3:   Conditional Covariance
#4:   Conditional Correlation
#5:   EW Portfolio Plot with conditional density VaR limits
myuspec = multispec(replicate(3, ugarchspec(mean.model = list(armaOrder = c(1,1)))))
mydcc = dccspec(myuspec, VAR = TRUE, lag = 1, lag.max = 12, dccOrder = c(1, 1), distribution = 'mvnorm')
md_dcc = dccfit(mydcc, data = data_D_rt_train, fit.control = list(eval.se=TRUE))

myuspec1 = multispec(replicate(2, ugarchspec(mean.model = list(armaOrder = c(1,1)))))
mydcc1 = dccspec(myuspec1, VAR = TRUE, lag = 1, lag.max = 12, dccOrder = c(1, 1), distribution = 'mvnorm
md_dcc_xau = dccfit(mydcc1, data = data_D_rt_train[, c(1, 3)], fit.control = list(eval.se=TRUE))
#plot(md_dcc_xau)

md_dcc_wti = dccfit(mydcc1, data = data_D_rt_train[, c(2, 3)], fit.control = list(eval.se=TRUE))
#plot(md_dcc_wti)
```

```r
#Predict with DCC-GARCH
pred_dcc = dccforecast(md_dcc, n.ahead = 14)
#pred_dcc
#fitted(pred_dcc)
rmse(predicted = as.numeric(unlist(fitted(pred_dcc)[3])), actual = as.numeric(unlist(data_D_rt_test[3]))
```

```
## [1] 0.002723984
```