Forecasting USD/IDR exchange rate

Kelvin Christian, Kristoffer Chandra, Vincent Widiaman

May 28, 2019

I. Introduction

The data we choose to analyze is the currency exchange of US Dollar with Indonesian Rupiah (Indonesian Rupiah per 1 US dollar). We will also use the data of S&P 500 as the gauge to measure the development of the U.S. economy. The reason we choose S&P 500 is because it can represent the condition of the U.S. economy. Our hypothesis says that the increase in the dollar price in rupiah is moderately correlated with the growth of S&P 500 index.

II. Results

Import the library for forecasting

```
library(quantmod)
library(ggplot2)
library(forecast)
library(vars)
library(lmtest)
library(timeSeries)
library(rugarch)
library(tseries)
library(Metrics)
library(strucchange)
```

a. Time Series plot

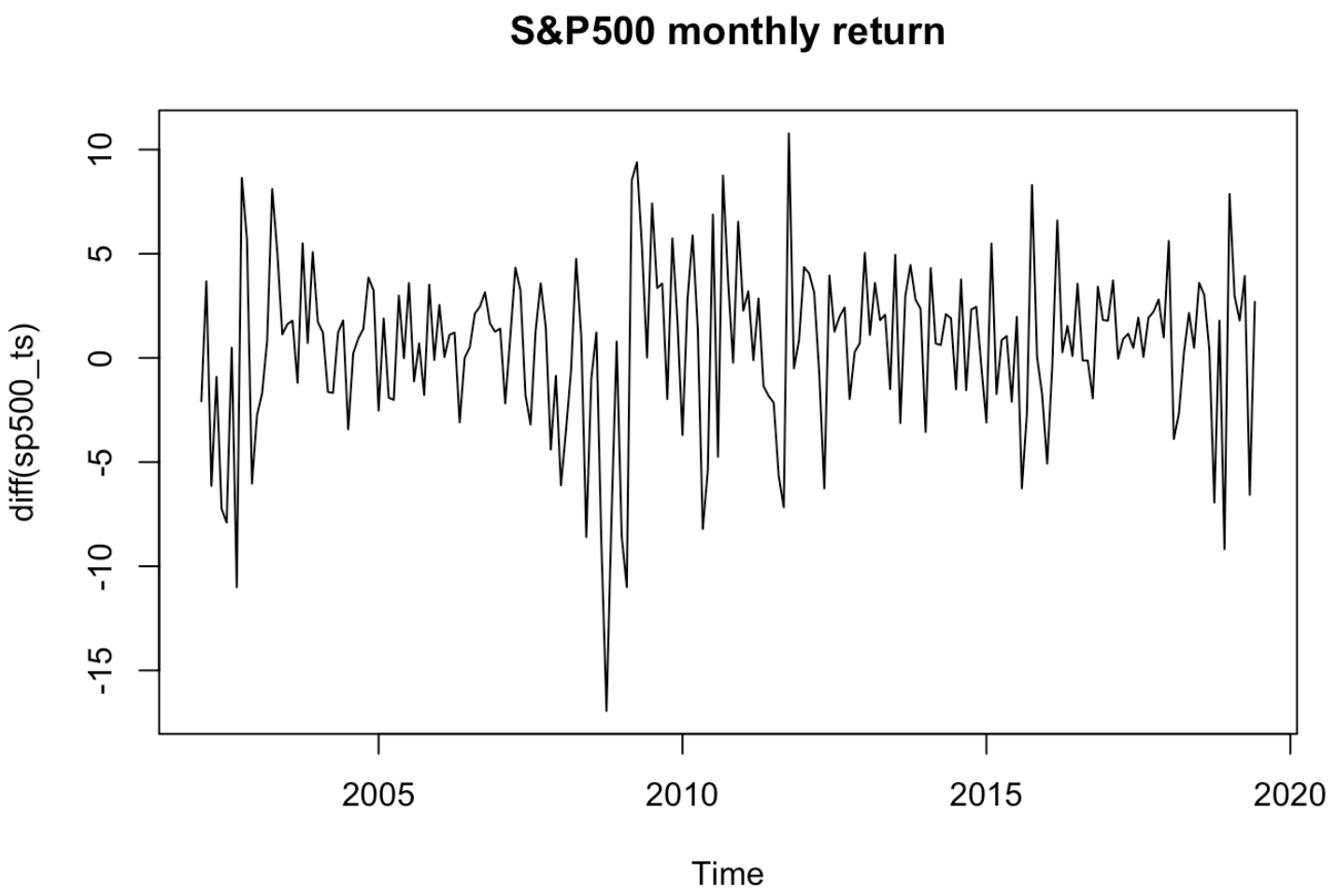
Using getSymbols function to download the daily data from the Yahoo finance. The data that we'll be analyzing is the monthly return data. So, we need to first convert from daily prices to monthly return time series, and calculate the return.

```
#Get necessary data. We will convert the data to monthly data for co
mpatibility
usd_idr <- getSymbols("IDR=X",auto.assign = FALSE, from = as.Date("0
1/01/2002", format = "%m/%d/%Y"))[,6]
usd_idr <- to.monthly(usd_idr)[,4]

sp500 <- getSymbols("^GSPC",auto.assign = FALSE, from = as.Date("01/
01/2002", format = "%m/%d/%Y"))[,6]
sp500 <- to.monthly(sp500)[,4]

#Convert Data to time series and calculate returns
usd_idr_ts <- ts(usd_idr, start = 2002, freq = 12)
sp500_ts <- ts(sp500, start = 2002, freq = 12)

usd_idr_ret=100*(diff(usd_idr_ts)/lag(usd_idr_ts, -1))
sp500_ret=100*(diff(sp500_ts)/lag(sp500_ts, -1))
```

Plot the data to get a picture of the return movement

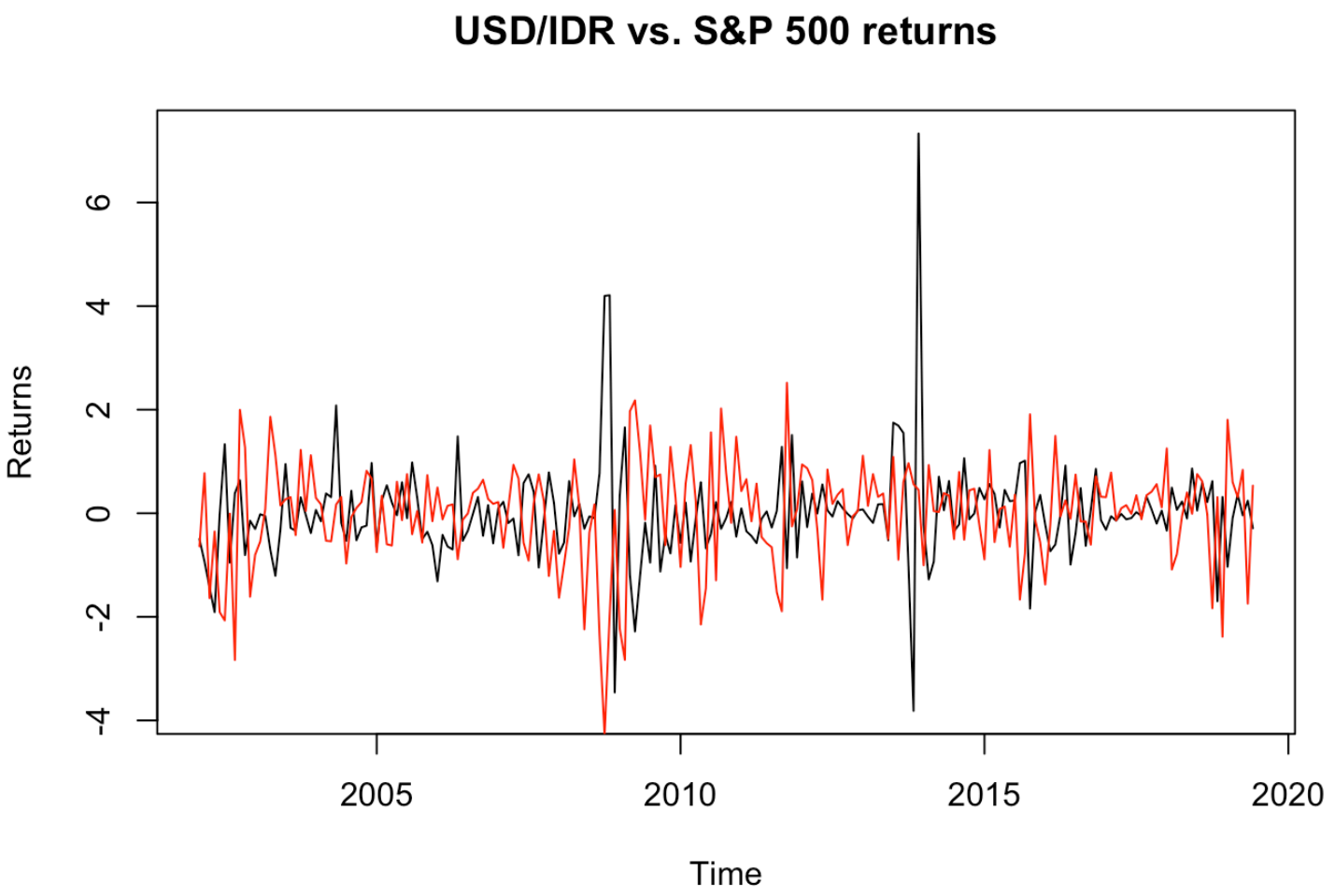
```
# plot for USD/IDR exchange rate (monthly)
plot(usd_idr_ret, main = "Monthly return of dollars in Indonesian ru
piah (IDR)")
```


```
# plot for S&P500 (monthly)
plot(sp500_ret, main = "S&P500 monthly return")
```



Since we are analyzing combined data, it is reasonable to plot both time series in one plot. In order to do that, we need to standardize the return first so they could fit together.

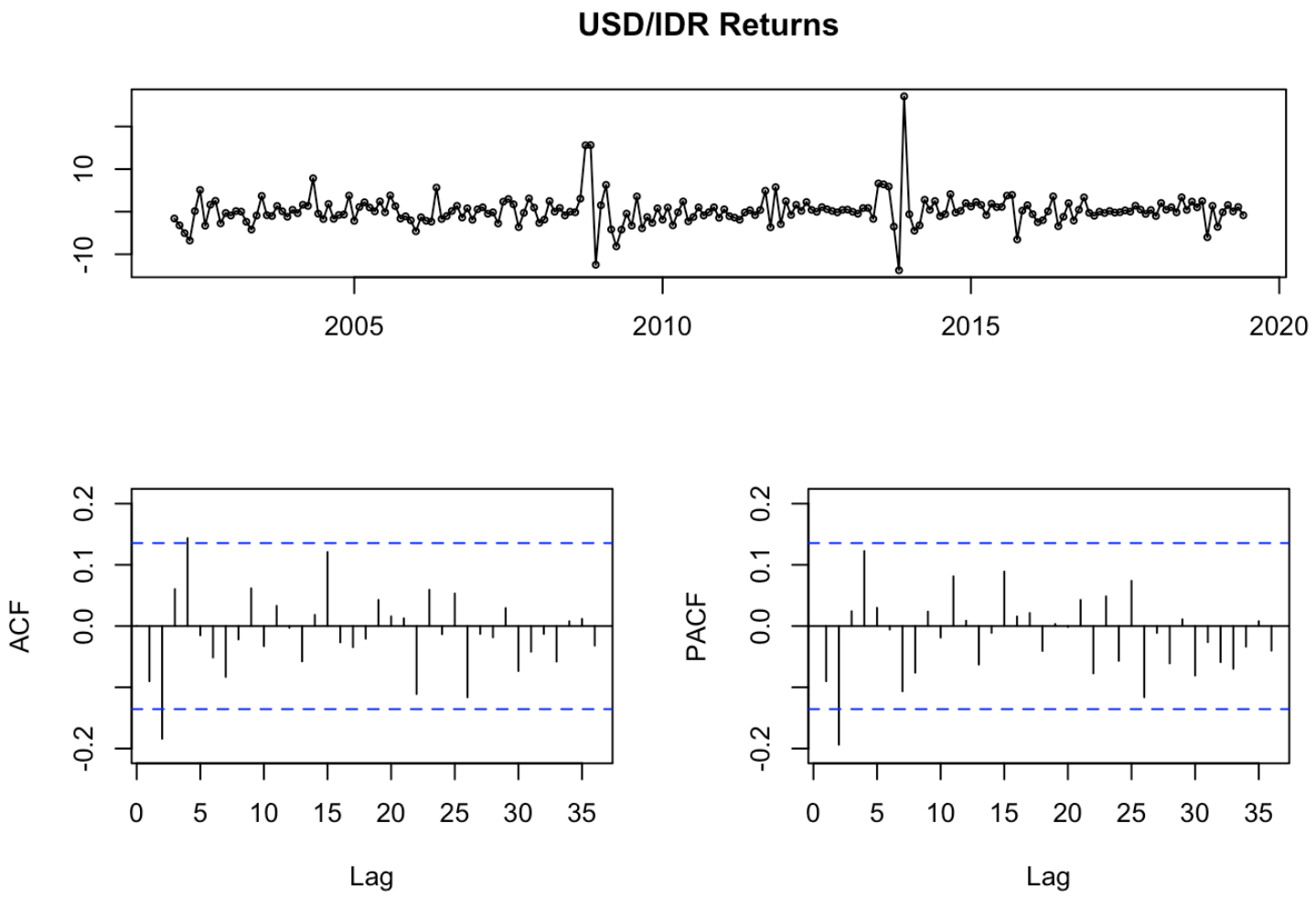
```
plot((usd_idr_ret-mean(usd_idr_ret))/sd(usd_idr_ret), main = "USD/ID
R vs. S&P 500 returns", ylab = "Returns")
lines((sp500_ret-mean(sp500_ret))/sd(sp500_ret), col = "red")
```



Looking at this combined plot, we can see that at year 2009, the returns for S&P 500 spikes down while the returns for USD/IDR spikes up. We initially predicted that the return of S&P 500 may explain the return of USD/IDR in a direct relationship. But this plot contradicts our initial assumption because the returns of dollar price spikes up when the returns of S&P 500 spikes down. They seem to have an inverse relationship.

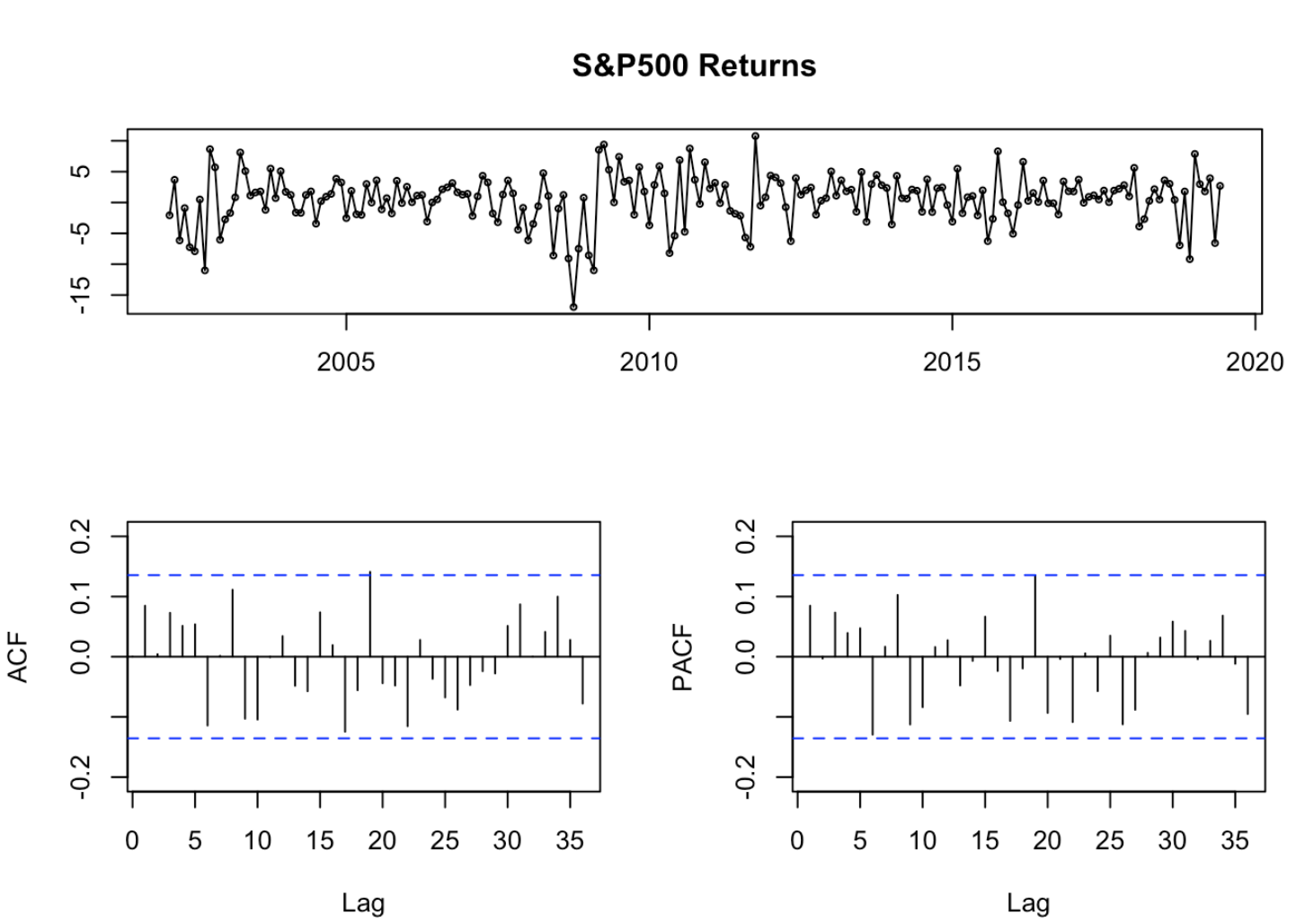
Then, we now consider the plot of ACF and PACF for both USD/IDR and S&P 500 returns to do further analysis and forecast.

```
tsdisplay(usd_idr_ret, main = "USD/IDR Returns")
```



Even though it looks like white noise, we can see that the ACF and PACF for the USD/IDR spikes at lag 2 while spikes at the other lags are not different from 0. Thus, it is reasonable to consider ARMA(2,2) as a good model for now.

```
tsdisplay(sp500_ret, main="S&P500 Returns")
```



The ACF and PACF for S&P500 monthly returns don't have any significant spikes at all lags. Our conclusion is that the returns follows white noise model.

b. Model fitting

Since the data that we're dealing with is the monthly returns data, it is covariance stationary. They have the same mean, or we could say that they have mean reversion. So both of our data doesn't have any significant linear trend. The only thing we consider is the seasonality and cycles component. Now, we will explore the model to find the best fit.

Model for USD/IDR returns

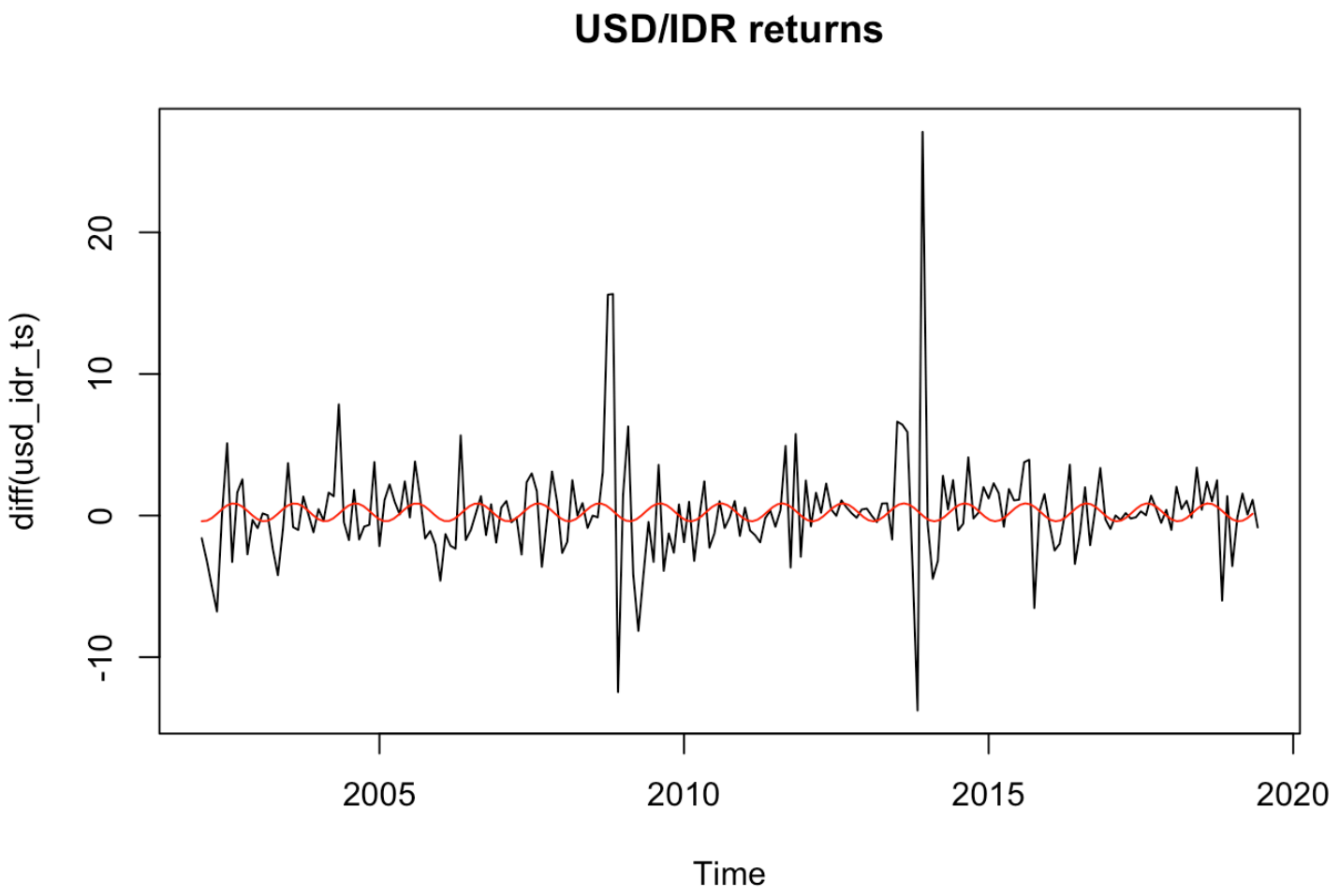
Periodic trend

The first model that we propose is the periodic trend for detecting seasonality. We created the variable time as a predictor and we use linear regression of returns with respect to sin and cos of time.

```r
# specify the index
t = seq(2002 + (1/12), 2019 + (4/12), length = length(usd_idr_ret))

# periodic
mod_period_usdidr <- lm(usd_idr_ret ~ I(sin(2*pi*t)) + I(cos(2*pi*t)
))
summary(mod_period_usdidr)
```

```
##
## Call:
## lm(formula = usd_idr_ret ~ I(sin(2 * pi * t)) + I(cos(2 * pi *
##     t)))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.3063  -1.6018  -0.0698   1.2802  26.8723
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)          0.2280     0.2529   0.902    0.368
## I(sin(2 * pi * t))  -0.4002     0.3564  -1.123    0.263
## I(cos(2 * pi * t))  -0.4926     0.3589  -1.373    0.171
##
## Residual standard error: 3.655 on 206 degrees of freedom
## Multiple R-squared:  0.0151, Adjusted R-squared:  0.005534
## F-statistic: 1.579 on 2 and 206 DF,  p-value: 0.2087
```

```r
plot(usd_idr_ret, main = "USD/IDR returns")
lines(mod_period_usdidr$fit ~ t, col = "red")
```



The result is not statistically significant, independently and jointly.

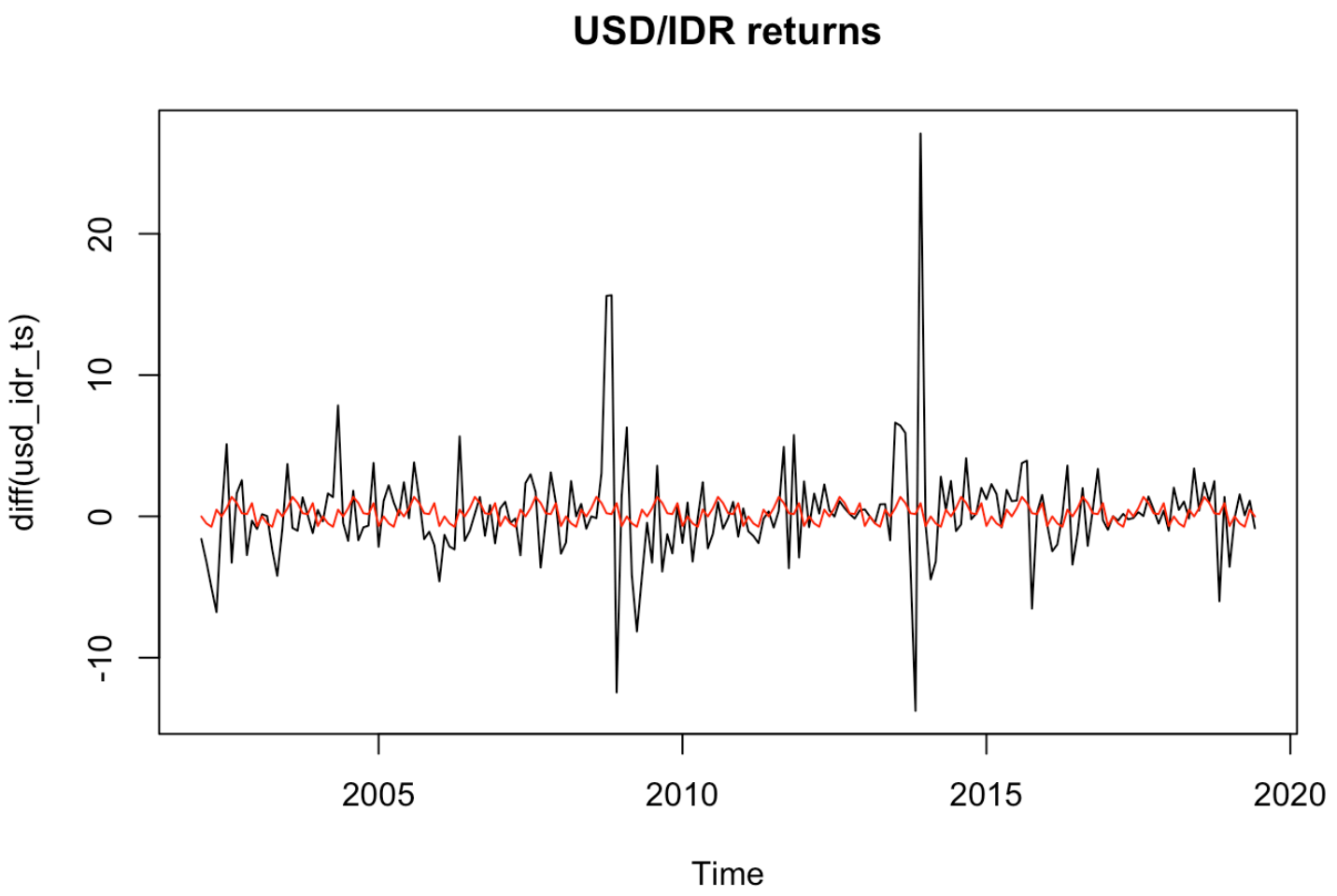
Seasonal Dummy trend

Since we're using the monthly data, we could use dummy variable for each month to know whether some month is significantly different from the others.

```
mod_seasonal_usdidr <- tslm(usd_idr_ret ~ season)
summary(mod_seasonal_usdidr)
```

```
##
## Call:
## tslm(formula = usd_idr_ret ~ season)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.927  -1.597  -0.060   1.138  26.164
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.68424    0.89945  -0.761    0.448
## season2      0.67657    1.25422   0.539    0.590
## season3      0.18518    1.25422   0.148    0.883
## season4     -0.05422    1.25422  -0.043    0.966
## season5      1.15672    1.25422   0.922    0.358
## season6      0.67862    1.25422   0.541    0.589
## season7      1.26175    1.27201   0.992    0.322
## season8      2.05868    1.27201   1.618    0.107
## season9      1.62904    1.27201   1.281    0.202
## season10     0.90645    1.27201   0.713    0.477
## season11     0.84425    1.27201   0.664    0.508
## season12     1.61551    1.27201   1.270    0.206
##
## Residual standard error: 3.709 on 197 degrees of freedom
## Multiple R-squared:  0.03047,    Adjusted R-squared:  -0.02366
## F-statistic: 0.5629 on 11 and 197 DF,  p-value: 0.8572
```

```
plot(usd_idr_ret, main = "USD/IDR returns")
lines(mod_seasonal_usdidr$fit, col = "red")
```



All seasonal dummy variables are not statistically significant and the joint hypothesis test is also not statistically significant, therefore we conclude that there is no seasonal component in USD/IDR returns.

ARMA (2,2)

Using our first hypothesis, we can try to fit ARMA(2,2) because the ACF and PACF spikes at lag 2. The function that we're using is arima function. However, the model object from arima function doesn't tell us the significant level of the coefficient. So we can create a function to analyze the significant level before summarizing the model.

```
# function to give summary for arima model
arima_summary <- function(model){
  coef_arma <- model$coef # coefficient
  se_arma <- sqrt(diag(model$var.coef)) # se
  t_arma <- coef_arma / se_arma # t stats
  p_arma <- pnorm(abs(t_arma), lower.tail = FALSE) # p values
  cbind("coef" = coef_arma, "se" = se_arma, "t" = t_arma, "p" = roun
d(p_arma, 5))
}

mod_arma_22_usdidr <- arima(usd_idr_ret, order = c(2,0,2))
arima_summary(mod_arma_22_usdidr)
```

```
##                coef         se         t       p
## ar1        0.5892018 0.14761778  3.991401 0.00003
## ar2       -0.7748280 0.09915721 -7.814137 0.00000
## ma1       -0.7538443 0.16379389 -4.602396 0.00000
## ma2        0.7728762 0.10153365  7.612020 0.00000
## intercept  0.2163540 0.20939717  1.033223 0.15075
```

It turns out that ARMA (2,2) does a great job. We can see that all the coefficients are statistically significant from the p-values.

ARMA (p,q)

Another thing to consider is fitting another combination (p,q) of ARMA model. To do that, we use for loop to fit all combinations of the order of ARMA, then we compare the model using AIC as our measurement. We first creates an empty matrix to store the AIC for each combination.

```
#For loop to find best ARMA(p,q)
m = 4 # combination for p = 1,2,3,4 with q = 1,2,3,4

AIC_mat <- matrix(NA, ncol = m, nrow = m)
rownames(AIC_mat) <- 1:m
colnames(AIC_mat) <- 1:m

for(i in 1:m){
  for(j in 1:m){
    AIC_mat[i,j] <- arima(usd_idr_ret, order = c(i,0,j))$aic
  }
}


AIC_mat # All AIC's
```

```
##          1        2        3        4
## 1 1138.667 1137.122 1136.635 1135.392
## 2 1135.341 1131.615 1132.254 1134.156
## 3 1136.329 1132.405 1134.204 1135.834
## 4 1136.016 1134.064 1135.827 1137.813
```

We can see that ARMA(2,2) has the smallest AIC compared to the others, we could also try ARMA(2,3) and ARMA(3,2) because their AIC are closed compared to ARMA(2,2)

ARMA(2,3)

```
mod_arma_23_usdidr <- arima(usd_idr_ret, order = c(2,0,3))
arima_summary(mod_arma_23_usdidr)
```

```
##                coef         se          t       p
## ar1        0.5690043 0.15833922  3.5935776 0.00016
## ar2       -0.6623429 0.15542441 -4.2615118 0.00001
## ma1       -0.6868572 0.16838746 -4.0790281 0.00002
## ma2        0.6021453 0.19272763  3.1243332 0.00089
## ma3        0.1108851 0.08986796  1.2338671 0.10863
## intercept  0.2122659 0.22774231  0.9320442 0.17566
```

Here the coefficient for ma3 is not significant.

ARMA(3,2)

```
mod_arma_32_usdidr <- arima(usd_idr_ret, order = c(3,0,2))
arima_summary(mod_arma_32_usdidr)
```

```
##                  coef         se          t       p
## ar1         0.69826975 0.16231402  4.301968 0.00001
## ar2        -0.75270025 0.11977804 -6.284125 0.00000
## ar3         0.09891972 0.08638328  1.145126 0.12608
## ma1        -0.82073777 0.15081989 -5.441840 0.00000
## ma2         0.71961358 0.13675258  5.262157 0.00000
## intercept   0.21215031 0.22834941  0.929060 0.17643
```

And here, the coefficient for ar3 is not statistically significant.

For now, we conclude that ARMA(2,2) is the best model because all variables are statistically significant.

Auto Arima

To check what R think the best model is, we can use auto arima function.

```
mod_autoarima_usdidr = auto.arima(usd_idr_ret)
mod_autoarima_usdidr
```

```
## Series: usd_idr_ret
## ARIMA(2,0,2)(1,0,0)[12] with zero mean
##
## Coefficients:
##          ar1      ar2      ma1     ma2    sar1
##       0.6002  -0.7995  -0.7603  0.7975  0.0545
## s.e.  0.1202   0.0976   0.1372  0.0914  0.0752
##
## sigma^2 estimated as 12.74:  log likelihood=-560.08
## AIC=1132.15   AICc=1132.57   BIC=1152.21
```

The auto arima agrees with ARMA(2,2) with additional seasonal AR(1). Next, we check the significant level for the seasonal ar coefficient.

```
arima_summary(mod_autoarima_usdidr)
```

```
##              coef         se          t       p
## ar1   0.60020361 0.12018183  4.9941294 0.00000
## ar2  -0.79954306 0.09759463 -8.1924907 0.00000
## ma1  -0.76029329 0.13717170 -5.5426394 0.00000
## ma2   0.79747234 0.09141463  8.7236841 0.00000
## sar1  0.05451889 0.07520649  0.7249227 0.23425
```

The s-ar(1) coefficient is not statistically significant. We could also compare the AIC with ARMA(2,2) to check which one is better.

```
AIC(mod_arma_22_usdidr, mod_autoarima_usdidr)
```

```
##                        df      AIC
## mod_arma_22_usdidr      6 1131.615
## mod_autoarima_usdidr    6 1132.152
```

The AIC shows that ARMA(2,2) is still a better model than ARMA(2,2) + S-AR(1). Therefore we will use ARMA(2,2) model for USD/IDR returns.

Model for S&P500 returns

Since the ACF and PACF of this data tells us that it is white noise, we can't make a guess. However, since the two dataset are somewhat have similar characteristic, we would choose ARMA(2,2) for the S&P500 as well.

ARMA(2,2)

We analyze the significance of ARMA(2,2) coefficients in the similar way as the USD/IDR,

```
mod_arma_22_sp500 <- arima(sp500_ret, order = c(2,0,2))
summary(mod_arma_22_sp500)
```

```
##
## Call:
## arima(x = sp500_ret, order = c(2, 0, 2))
##
## Coefficients:
##           ar1      ar2     ma1     ma2  intercept
##       -1.7792  -0.9390  1.8533  0.9944     0.5227
## s.e.   0.0448   0.0468  0.0998  0.1119     0.2836
##
## sigma^2 estimated as 15.7:  log likelihood = -585.75,  aic = 1183
.5
##
## Training set error measures:
##                       ME     RMSE      MAE      MPE     MAPE
MASE
## Training set 0.0005804051 3.961941 2.932991 77.31808 194.1101 0.7
092201
##                     ACF1
## Training set 0.05543543
```

The standard errors are NaN in this case so we cannot analyze the significance of each coefficient parameters.

Auto Arima

Then, we cross-check with auto arima to see which one is better,

```
mod_autoarima_sp500 = auto.arima(sp500_ret)
mod_autoarima_sp500
```

```
## Series: sp500_ret
## ARIMA(2,0,2)(1,0,0)[12] with non-zero mean
##
## Coefficients:
##           ar1      ar2     ma1     ma2     sar1    mean
##       -0.2627  -0.9347  0.3425  0.9397  -0.0272  0.5277
## s.e.   0.0511   0.0468  0.0413  0.0565   0.0802  0.2776
##
## sigma^2 estimated as 16.14:  log likelihood=-584.47
## AIC=1182.94   AICc=1183.49   BIC=1206.33
```

```
arima_summary(mod_autoarima_sp500)
```

```
##                  coef         se           t       p
## ar1       -0.26265755 0.05108300  -5.1417802 0.00000
## ar2       -0.93466094 0.04684745 -19.9511583 0.00000
## ma1        0.34250247 0.04125267   8.3025519 0.00000
## ma2        0.93966849 0.05646708  16.6409965 0.00000
## sar1      -0.02719424 0.08024887  -0.3388738 0.36735
## intercept  0.52766309 0.27756200   1.9010639 0.02865
```

We can see that the coefficients for AR and MA order 1,2 are statistically significant. However, S-AR(1) is not statistically significant. ARMA(2,2) may be a good model.

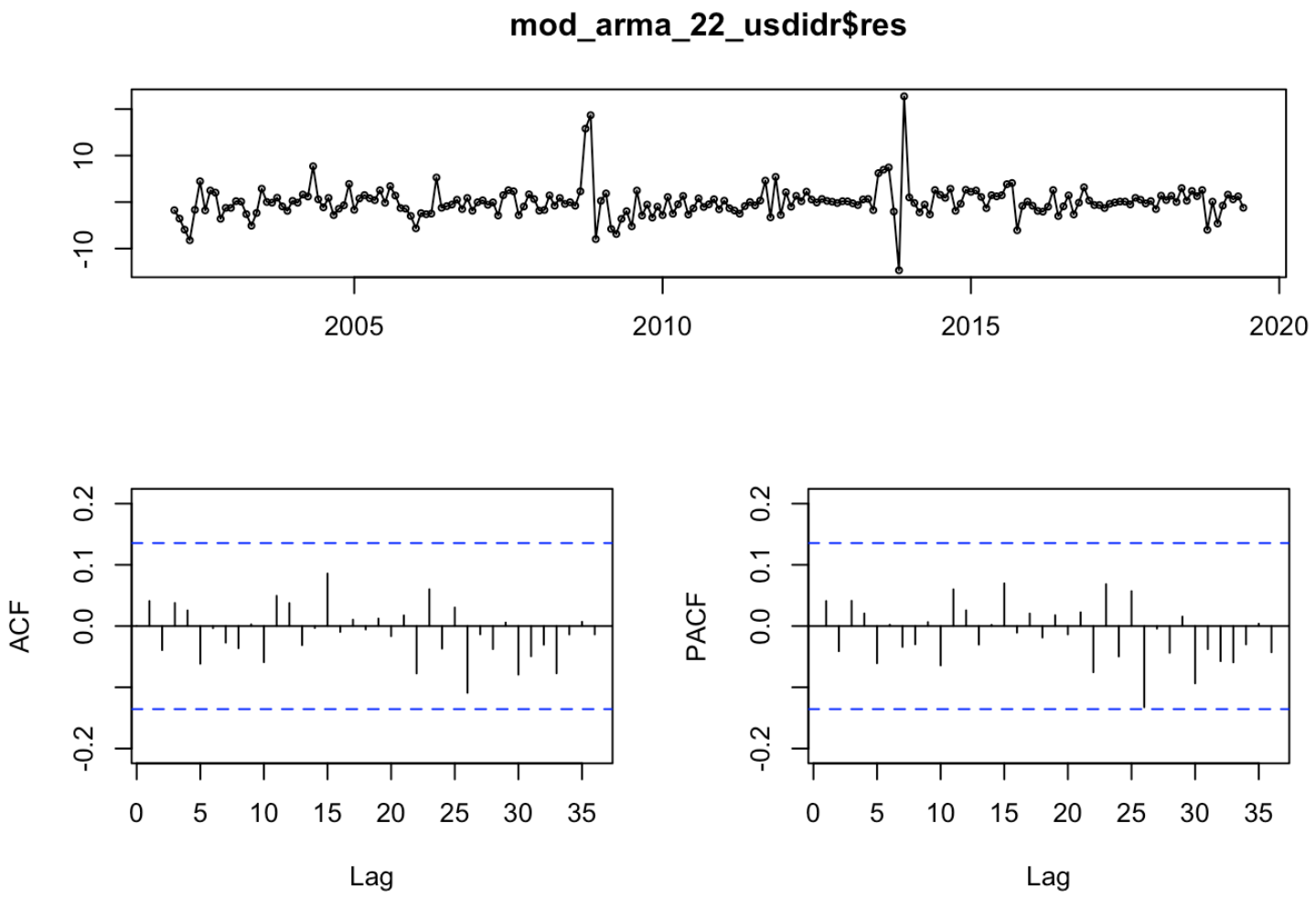
Analyzing GARCH(p,q)

Although our current model, ARMA(2,2), may be good, we still dont know if GARCH is needed to better fit the model.

GARCH(p,q) for USD/IDR

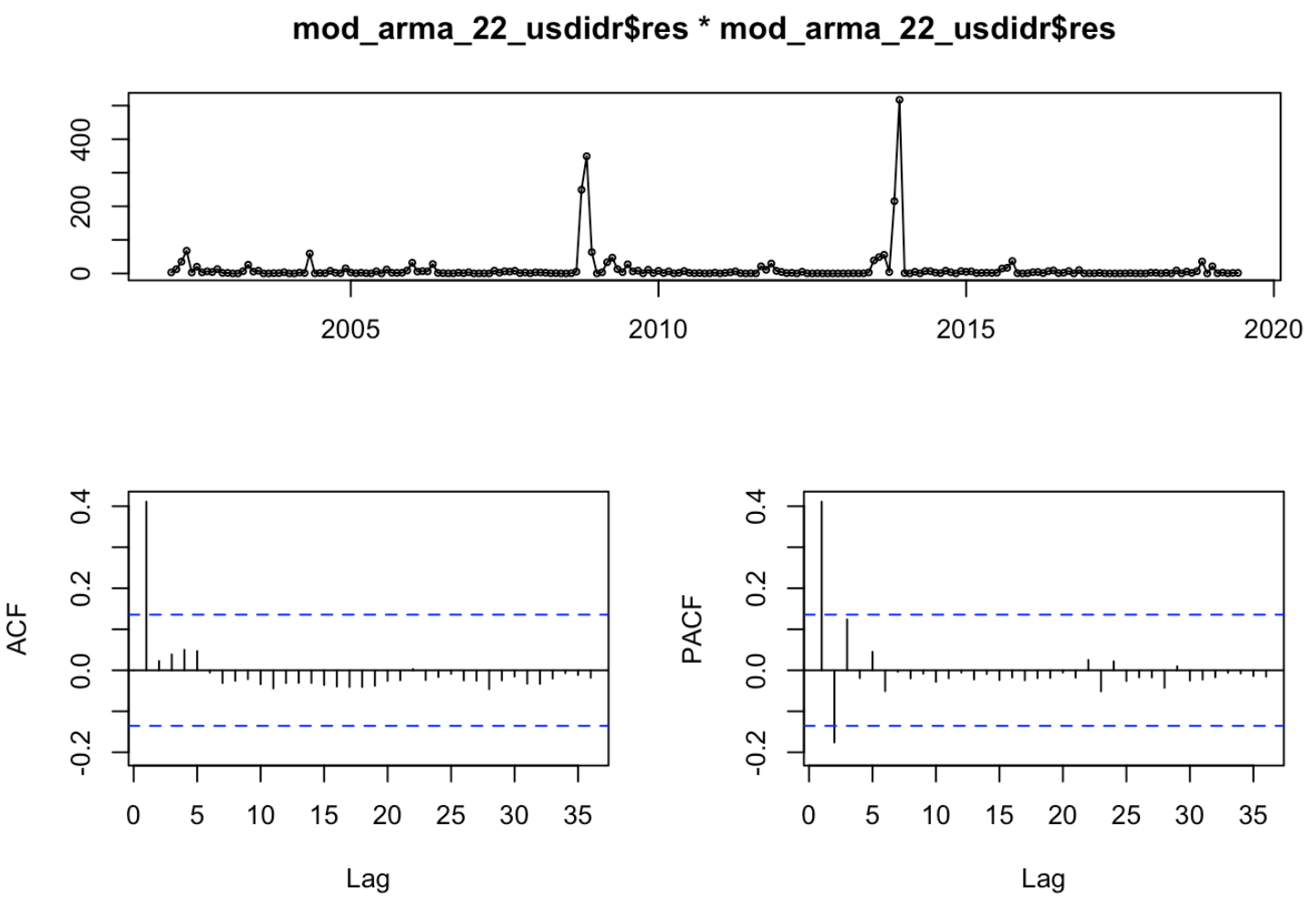
First, we look at the residuals of the USD/IDR ARMA(2,2),

```
tsdisplay(mod_arma_22_usdidr$res)
```



It seems that the residuals are all almost zero which is a good signal. Now, we square the residuals to detect any need for GARCH.

```
#Analyze GARCH
tsdisplay(mod_arma_22_usdidr$res*mod_arma_22_usdidr$res)
```



We can see that in the squared residuals, there are still spikes which indicates the need for GARCH model. We now find the best p and q for GARCH(p,q) using for loop and compare their MSE and MAE.

```
#Use for loop to find best GARCH(p,q)
m <- 6
garch_mse_mat <- matrix(NA, ncol = m, nrow = m)
garch_mae_mat <- matrix(NA, ncol = m, nrow = m)
for (i in 1:m){
  for(j in 1:m){
    model <- ugarchspec(
      variance.model = list(model = "sGARCH", garchOrder = c(i, j)),
      mean.model = list(armaOrder = c(2, 2), include.mean = TRUE),
      distribution.model = "sstd")

    modelfit <- ugarchfit(spec=model,data=usd_idr_ret)

    a <- attributes(modelfit)

    garch_mse_mat[i,j] <- mse(usd_idr_ret, a$fit$fitted.values)
    garch_mae_mat[i,j] <- mae(usd_idr_ret, a$fit$fitted.values)

  }
}

garch_mae_mat
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
## [1,] 2.141705 2.143322 2.147085 2.142597 2.144062 2.141844
## [2,] 2.141697 2.143326 2.145338 2.135026 2.137474 2.137897
## [3,] 2.140828 2.142434 2.145333 2.135015 2.137468 2.137894
## [4,] 2.132256 2.134233 2.140605 2.136185 2.137117 2.137661
## [5,] 2.143129 2.143130 2.142514 2.137220 2.137120 2.137661
## [6,] 2.143464 2.143450 2.141679 2.141675 2.141674 2.141663
```

```
garch_mse_mat
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
## [1,] 12.70991 12.71964 12.72589 12.72645 12.70598 12.73655
## [2,] 12.70985 12.71964 12.70153 12.66804 12.68567 12.69495
## [3,] 12.71091 12.72052 12.70145 12.66798 12.68557 12.69491
## [4,] 12.63939 12.67429 12.70355 12.67021 12.68697 12.69455
## [5,] 12.70683 12.70684 12.71190 12.67482 12.68702 12.69454
## [6,] 12.72171 12.72114 12.70481 12.70480 12.70482 12.70471
```

```
which(garch_mae_mat == min(garch_mae_mat), arr.ind = TRUE)
```

```
##      row col
## [1,]   4   1
```

```
which(garch_mse_mat == min(garch_mse_mat), arr.ind = TRUE)
```

```
##      row col
## [1,]   4   1
```

From the for loop summary, we find that the best value for p,q is 4 and 1 respectively. Thus, we use GARCH(4,1) for the model.

```
garch_mod=ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(4, 1)),
  mean.model = list(armaOrder = c(2, 2), include.mean = TRUE),
  distribution.model = "sstd")

# Fit the model
garchfit41_usdidr = ugarchfit(spec=garch_mod, data=usd_idr_ret)
garchfit41_usdidr
```

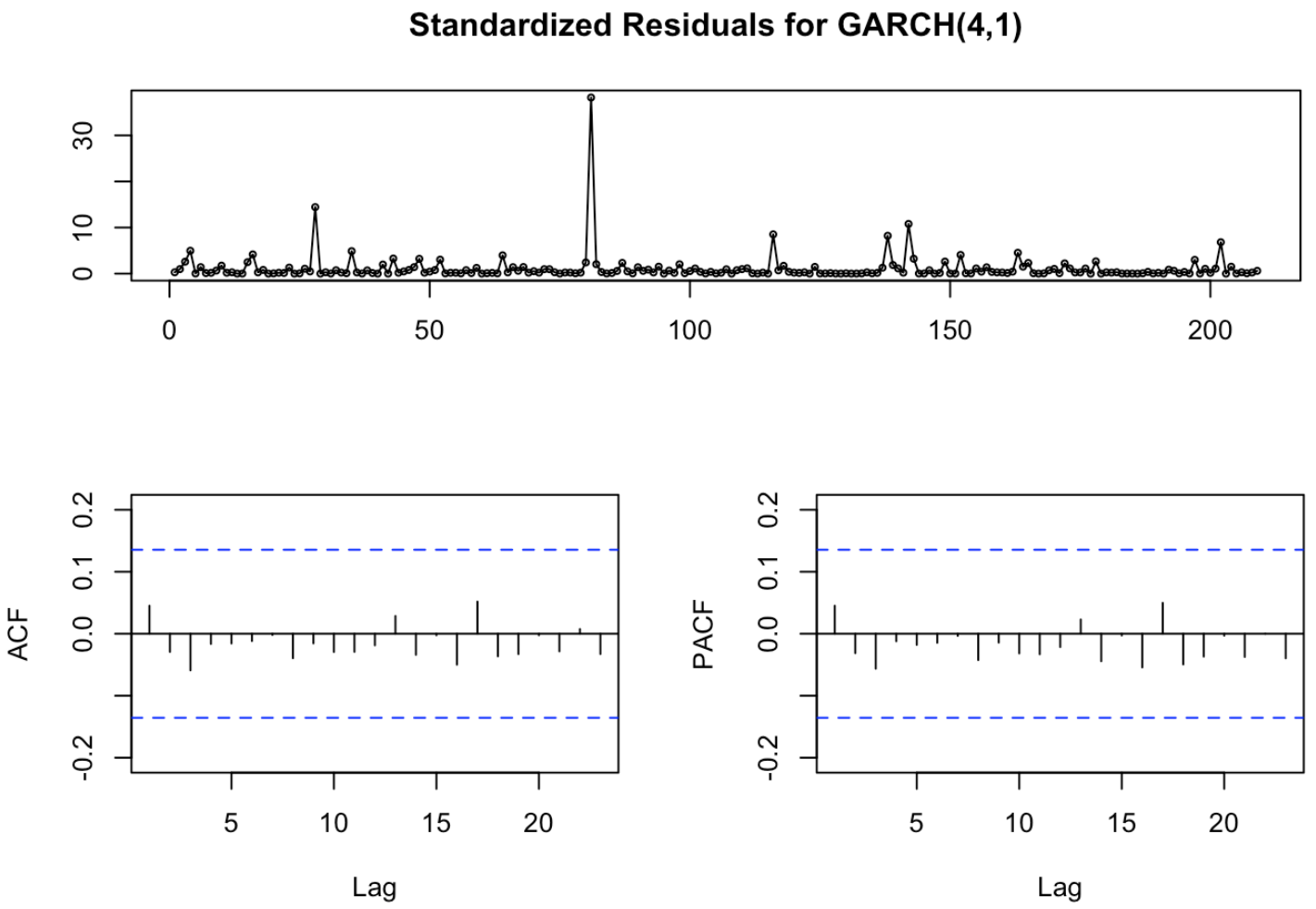
```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
```

```
## GARCH Model  : sGARCH(4,1)
## Mean Model   : ARFIMA(2,0,2)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.306626    0.138163  2.21930 0.026466
## ar1     0.733426    0.211868  3.46172 0.000537
## ar2    -0.598147    0.128974 -4.63771 0.000004
## ma1    -0.901693    0.193050 -4.67076 0.000003
## ma2     0.713115    0.111848  6.37574 0.000000
## omega   2.234147    0.483045  4.62513 0.000004
## alpha1  0.635960    0.252786  2.51581 0.011876
## alpha2  0.243913    0.136619  1.78535 0.074205
## alpha3  0.000000    0.059406  0.00000 1.000000
## alpha4  0.077037    0.088196  0.87348 0.382402
## beta1   0.000000    0.189660  0.00000 1.000000
## skew    1.220646    0.116382 10.48827 0.000000
## shape   4.212658    1.132621  3.71939 0.000200
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.306626    0.191101  1.60452 0.108599
## ar1     0.733426    0.255860  2.86651 0.004150
## ar2    -0.598147    0.154611 -3.86871 0.000109
## ma1    -0.901693    0.256699 -3.51265 0.000444
## ma2     0.713115    0.144473  4.93597 0.000001
## omega   2.234147    0.809318  2.76053 0.005771
## alpha1  0.635960    0.318412  1.99729 0.045794
## alpha2  0.243913    0.234121  1.04182 0.297494
## alpha3  0.000000    0.073321  0.00000 1.000000
## alpha4  0.077037    0.092587  0.83205 0.405379
## beta1   0.000000    0.127279  0.00000 1.000000
## skew    1.220646    0.147887  8.25393 0.000000
## shape   4.212658    1.092303  3.85667 0.000115
##
## LogLikelihood : -479.6776
##
## Information Criteria
## ------------------------------------
##
```

```
## Akaike       4.7146
## Bayes        4.9225
## Shibata      4.7075
## Hannan-Quinn 4.7987
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      3.180 0.07453
## Lag[2*(p+q)+(p+q)-1][11]    4.006 0.99990
## Lag[4*(p+q)+(p+q)-1][19]    6.775 0.92674
## d.o.f=4
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.4335  0.5103
## Lag[2*(p+q)+(p+q)-1][14]   1.7244  0.9954
## Lag[4*(p+q)+(p+q)-1][24]   2.8074  0.9998
## d.o.f=5
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[6]    0.0291 0.500 2.000  0.8646
## ARCH Lag[8]    0.2746 1.480 1.774  0.9560
## ARCH Lag[10]   0.4348 2.424 1.650  0.9893
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  2.4617
## Individual Statistics:
## mu     0.42784
## ar1    0.29881
## ar2    0.14182
## ma1    0.28443
## ma2    0.14477
## omega  0.32511
## alpha1 0.07268
## alpha2 0.04379
## alpha3 0.18787
```

```
## alpha4 0.07693
## beta1  0.04998
## skew   0.26416
## shape  0.11016
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.89 3.15 3.69
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## -----------------------------------
##                    t-value   prob sig
## Sign Bias           1.0077 0.3148
## Negative Sign Bias  0.4730 0.6368
## Positive Sign Bias  0.1611 0.8722
## Joint Effect        1.0779 0.7824
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     17.51       0.5555
## 2    30     32.00       0.3197
## 3    40     34.64       0.6691
## 4    50     44.35       0.6619
##
##
## Elapsed time : 0.3727169
```

```
att_garch_usdidr = attributes(garchfit41_usdidr)
 # standardized residuals
tsdisplay((att_garch_usdidr$fit$residuals*att_garch_usdidr$fit$resid
uals)/att_garch_usdidr$fit$var,
          main = "Standardized Residuals for GARCH(4,1)")
```

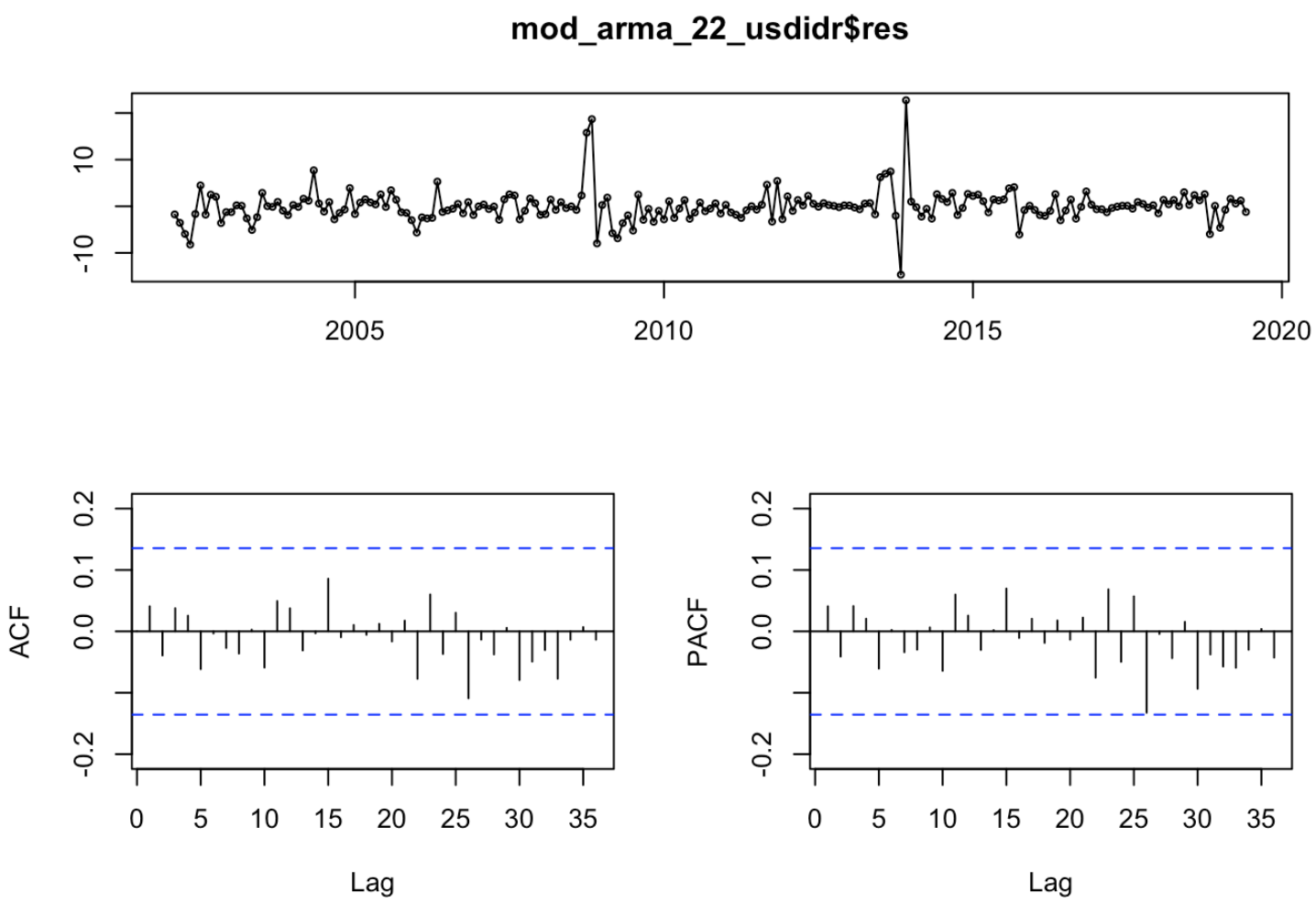


From the ACF PACF we can clearly see that the GARCH model eliminates the remaining spikes of the residual. Thus, we use GARCH(4,1)

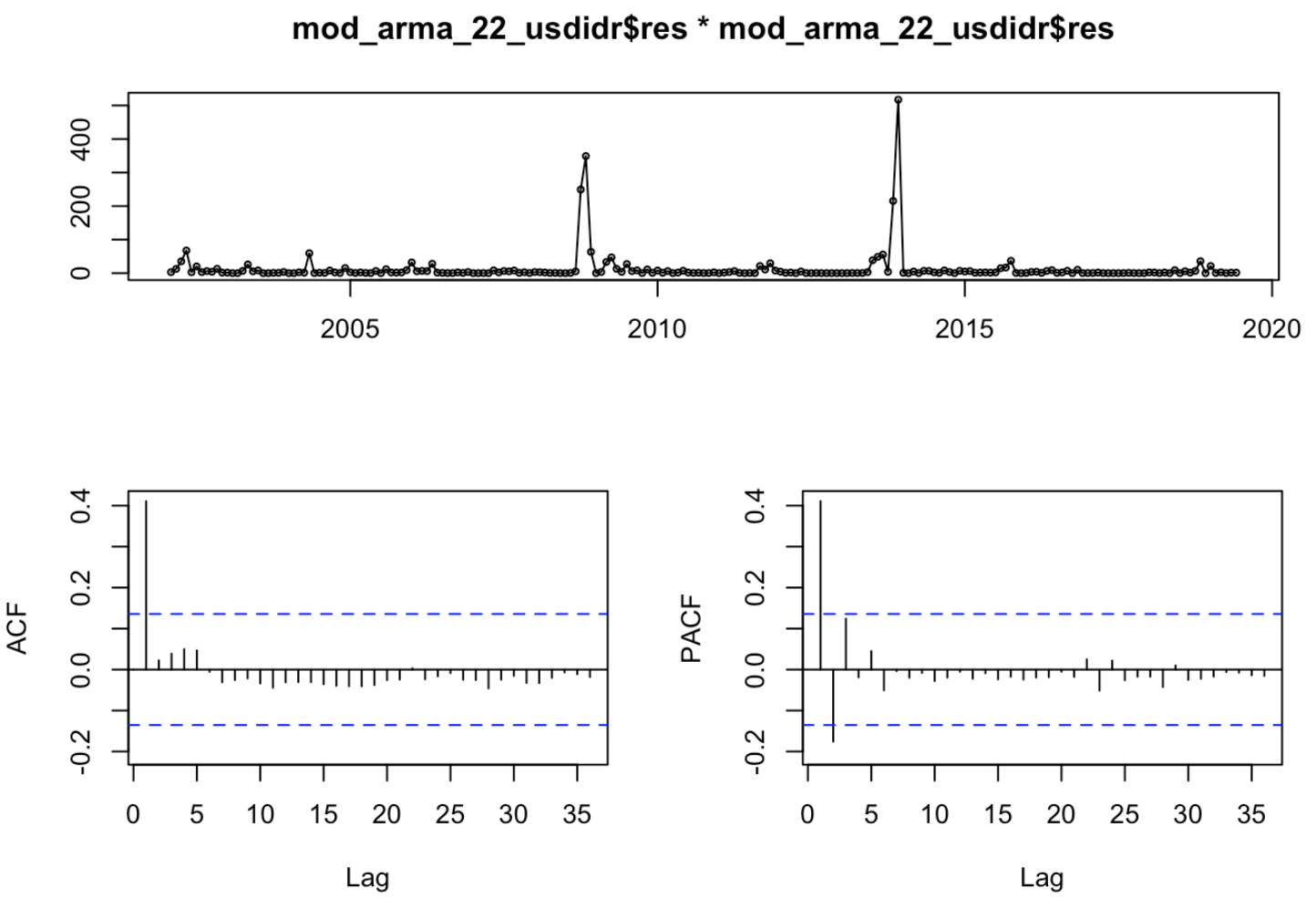
GARCH(p,q) for S&P500

Now, we will find the best GARCH(p,q) value for S&P500 using the same approach. First, we look at the residual and squared residuals.

```
tsdisplay(mod_arma_22_usdidr$res)
```




```
tsdisplay(mod_arma_22_usdidr$res*mod_arma_22_usdidr$res)
```



We see similar pattern as before, there are still spikes in the model. Thus, we look for GARCH.

```
m <- 6
garch_mse_mat <- matrix(NA, ncol = m, nrow = m)
garch_mae_mat <- matrix(NA, ncol = m, nrow = m)
for (i in 1:m){
  for(j in 1:m){
    model <- ugarchspec(
      variance.model = list(model = "sGARCH", garchOrder = c(i, j)),
      mean.model = list(armaOrder = c(2, 2), include.mean = TRUE),
      distribution.model = "sstd")

    modelfit <- ugarchfit(spec=model,data=sp500_ret)

    a <- attributes(modelfit)

    garch_mse_mat[i,j] <- mse(sp500_ret, a$fit$fitted.values)
    garch_mae_mat[i,j] <- mae(sp500_ret, a$fit$fitted.values)

  }
}

garch_mae_mat
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
## [1,] 2.938728 2.993345 2.944164 2.995949 2.999478 2.950865
## [2,] 2.976711 2.989926 2.933992 2.847566 2.944207 2.955649
## [3,] 2.950073 2.989711 2.989726 2.954399 2.933140 2.956397
## [4,] 2.989932 2.939767 2.949496 2.992386 2.956795 2.867436
## [5,] 2.991554 2.990544 2.990742 2.993137 2.973336 2.998061
## [6,] 2.999475 2.999769 2.999778 2.999835 2.995929 2.998026
```

```
garch_mse_mat
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
## [1,] 15.99192 16.55720 16.19694 16.58081 16.60049 16.14323
## [2,] 16.66957 16.52144 16.17350 16.06131 16.17168 16.20577
## [3,] 16.17167 16.52510 16.56646 16.17329 16.27138 16.17632
## [4,] 16.53594 15.97969 16.14722 16.54161 16.19954 15.89813
## [5,] 16.54964 16.53179 16.56862 16.54848 16.74078 16.59599
## [6,] 16.51894 16.51920 16.51922 16.51908 16.55183 16.59620
```

```
which(garch_mae_mat == min(garch_mae_mat), arr.ind = TRUE)
```

```
##      row col
## [1,]   2   4
```

```
which(garch_mse_mat == min(garch_mse_mat), arr.ind = TRUE)
```

```
##      row col
## [1,]   4   6
```

We can see that the smallest error in the analysis above is at GARCH(2,1) and GARCH(2,6). Thus, in this case we use GARCH(2,1) as it is a simpler model with less parameters.

```
garch_mod=ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(2, 1)),
  mean.model = list(armaOrder = c(2, 2), include.mean = TRUE),
  distribution.model = "sstd")

# Fit the model
garchfit21_sp500 = ugarchfit(spec=garch_mod, data=sp500_ret)
garchfit21_sp500
```

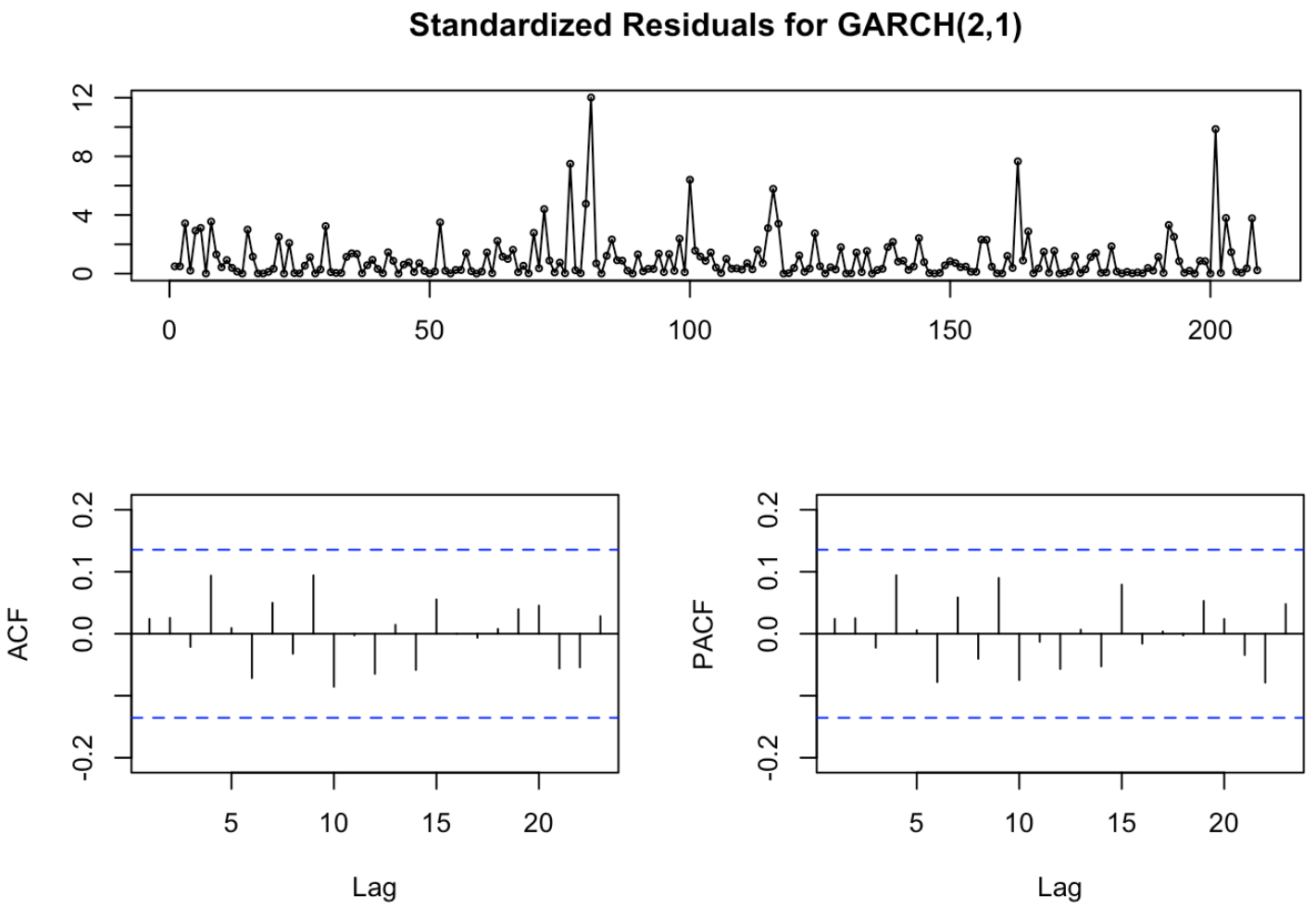
```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(2,1)
## Mean Model   : ARFIMA(2,0,2)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu       0.79487    0.195069   4.07483 0.000046
```

```
## ar1     -0.81081    0.029882 -27.13406 0.000000
## ar2     -0.96540    0.015283 -63.16700 0.000000
## ma1      0.79080    0.015268  51.79576 0.000000
## ma2      0.97784    0.013481  72.53251 0.000000
## omega    1.77279    0.764817   2.31793 0.020453
## alpha1   0.16942    0.074898   2.26205 0.023695
## alpha2   0.18916    0.138200   1.36871 0.171088
## beta1    0.53530    0.123605   4.33077 0.000015
## skew     0.61284    0.091453   6.70115 0.000000
## shape   60.00000  116.783338   0.51377 0.607411
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu       0.79487    0.177359   4.4817 0.000007
## ar1     -0.81081    0.024881 -32.5881 0.000000
## ar2     -0.96540    0.021824 -44.2359 0.000000
## ma1      0.79080    0.013402  59.0081 0.000000
## ma2      0.97784    0.012581  77.7208 0.000000
## omega    1.77279    0.783978   2.2613 0.023742
## alpha1   0.16942    0.075461   2.2452 0.024758
## alpha2   0.18916    0.160828   1.1761 0.239539
## beta1    0.53530    0.139497   3.8374 0.000124
## skew     0.61284    0.094312   6.4980 0.000000
## shape   60.00000   43.279430   1.3863 0.165643
##
## LogLikelihood : -553.2903
##
## Information Criteria
## ------------------------------------
##
## Akaike       5.3999
## Bayes        5.5758
## Shibata      5.3947
## Hannan-Quinn 5.4710
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1663  0.6834
## Lag[2*(p+q)+(p+q)-1][11]   2.6837  1.0000
## Lag[4*(p+q)+(p+q)-1][19]   6.1912  0.9625
## d.o.f=4
```

```
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1214  0.7276
## Lag[2*(p+q)+(p+q)-1][8]    2.0862  0.8454
## Lag[4*(p+q)+(p+q)-1][14]   4.6496  0.8115
## d.o.f=3
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[4]     1.836 0.500 2.000  0.1755
## ARCH Lag[6]     2.564 1.461 1.711  0.3786
## ARCH Lag[8]     3.087 2.368 1.583  0.5273
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  2.0786
## Individual Statistics:
## mu     0.05138
## ar1    0.11599
## ar2    0.20436
## ma1    0.03729
## ma2    0.15394
## omega  0.03231
## alpha1 0.03855
## alpha2 0.03259
## beta1  0.09433
## skew   0.09130
## shape  0.33371
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.49 2.75 3.27
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value    prob sig
## Sign Bias           0.9132 0.36222
## Negative Sign Bias  1.1148 0.26623
```

```
## Positive Sign Bias  2.0091 0.04585  **
## Joint Effect        6.4854 0.09024   *
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     10.62       0.9360
## 2    30     25.69       0.6421
## 3    40     34.64       0.6691
## 4    50     44.35       0.6619
##
##
## Elapsed time : 0.415601
```

```
att_garch_sp500 = attributes(garchfit21_sp500)
# standardized residuals
tsdisplay((att_garch_sp500$fit$residuals*att_garch_sp500$fit$residua
ls)/att_garch_sp500$fit$var,
          main = "Standardized Residuals for GARCH(2,1)")
```

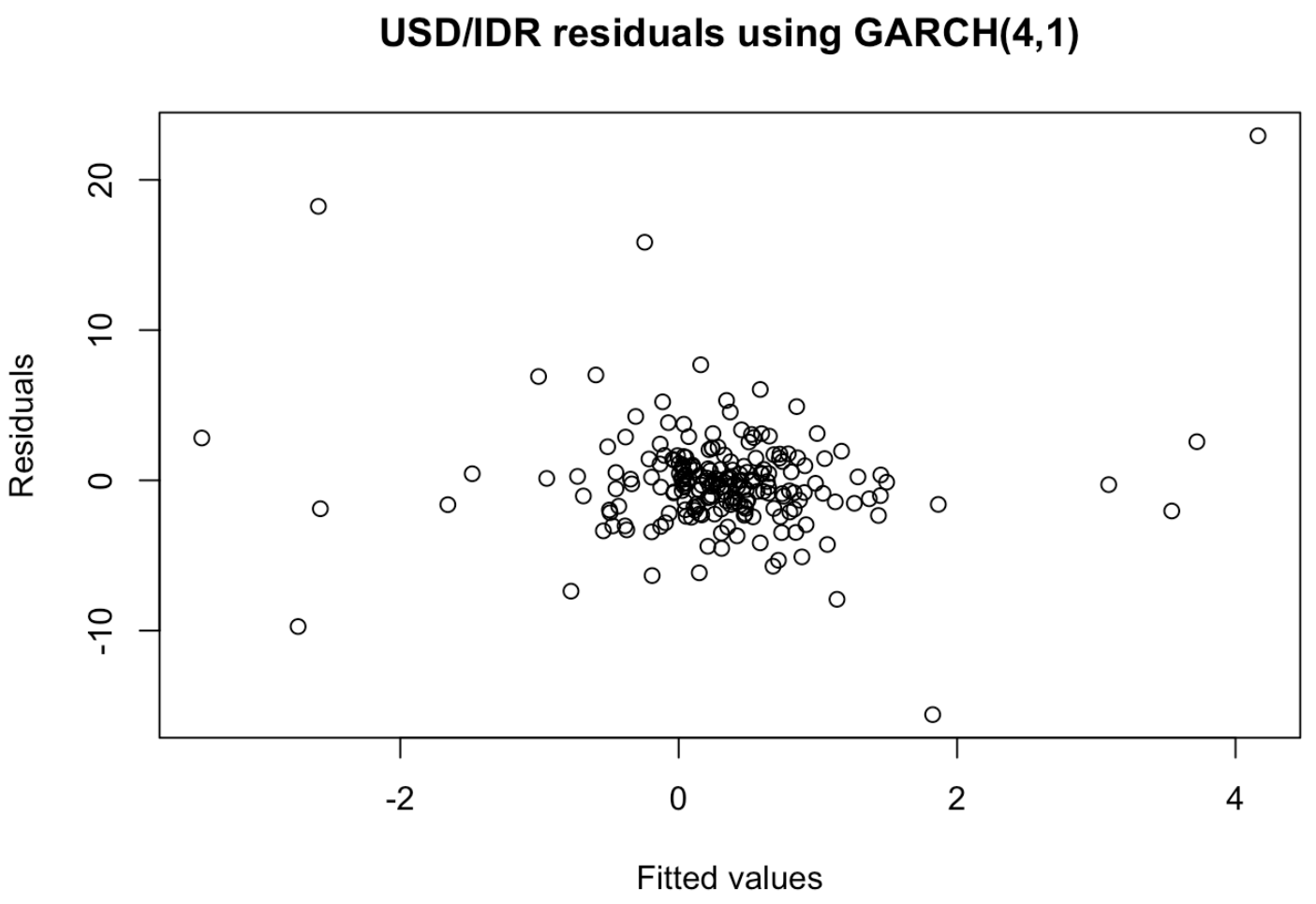


From this, we can also see that the standardized squared residuals is white noise. Thus, we use GARCH(2,1).

c. Respective Residuals

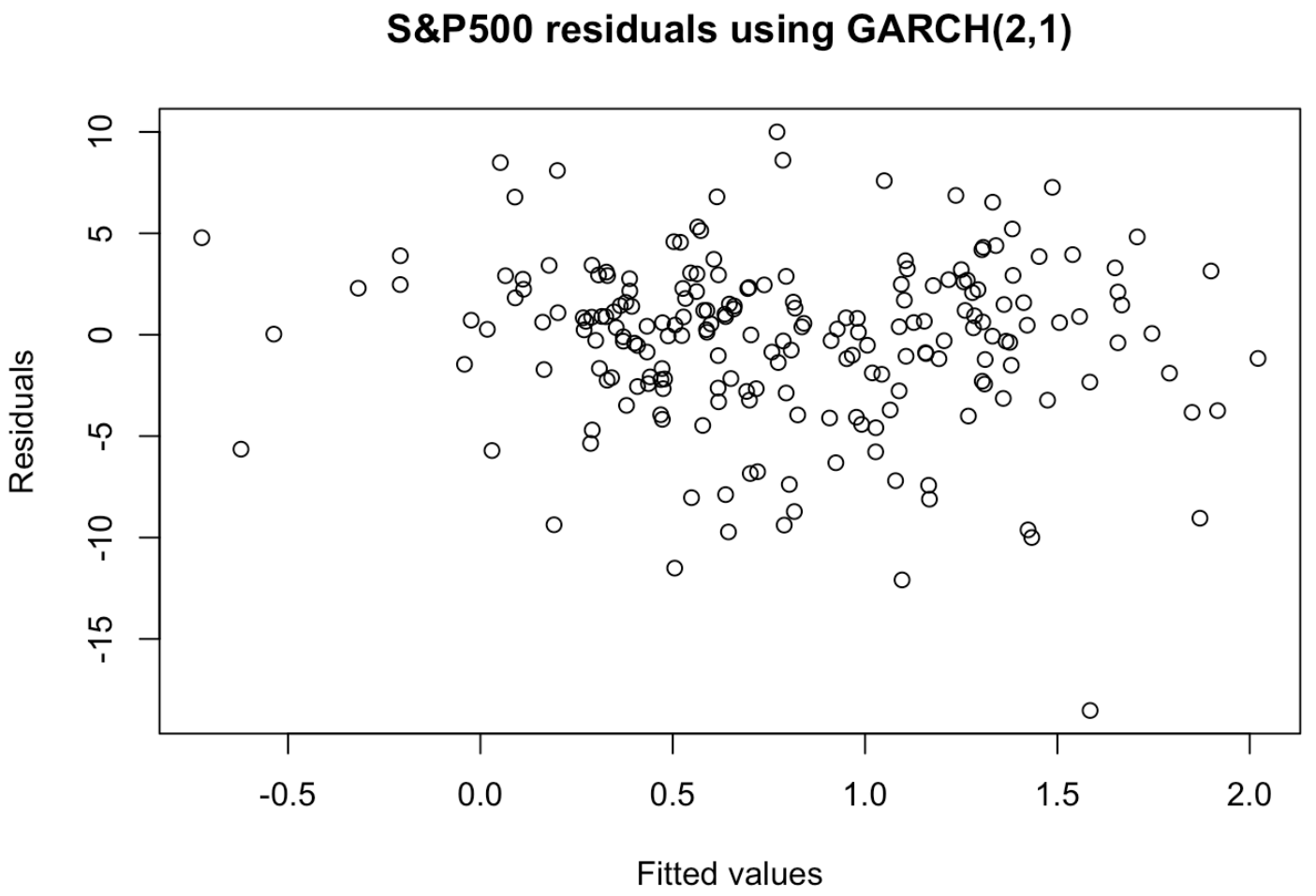
USD/IDR model residuals

```
plot(att_garch_usdidr$fit$residuals ~ att_garch_usdidr$fit$fitted.va
lues,
     main = "USD/IDR residuals using GARCH(4,1)",
     xlab = "Fitted values",
     ylab = "Residuals")
```



S&P500 model residuals

```
plot(att_garch_sp500$fit$residuals ~ att_garch_sp500$fit$fitted.valu
es,
     main = "S&P500 residuals using GARCH(2,1)",
     xlab = "Fitted values",
     ylab = "Residuals")
```

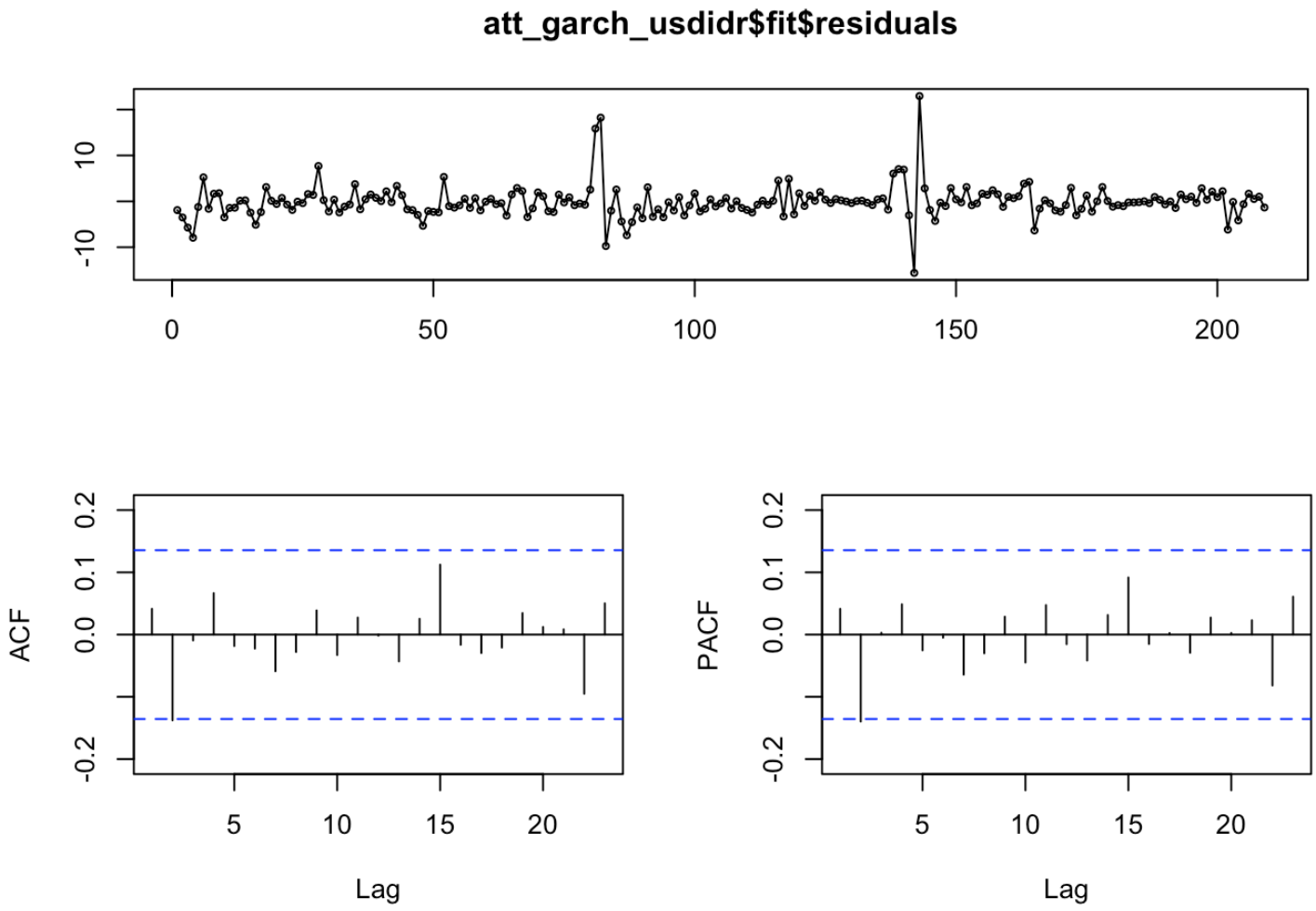


The residuals for both data are pretty much randomly distributed. So our model is considerably good. Next, we can do further analysis on the ACF and PACF of the residuals.

e. ACF and PACF of residuals

USD/IDR residuals

```
tsdisplay(att_garch_usdidr$fit$residuals)
```



There is no significant spikes on both the ACF and PACF which show that the residuals follows white noise model. We can also check the residuals using Ljung-Box test with the null defined by:

$$H_0 : \rho_1 = \rho_2 = \ldots = \rho_k = 0$$

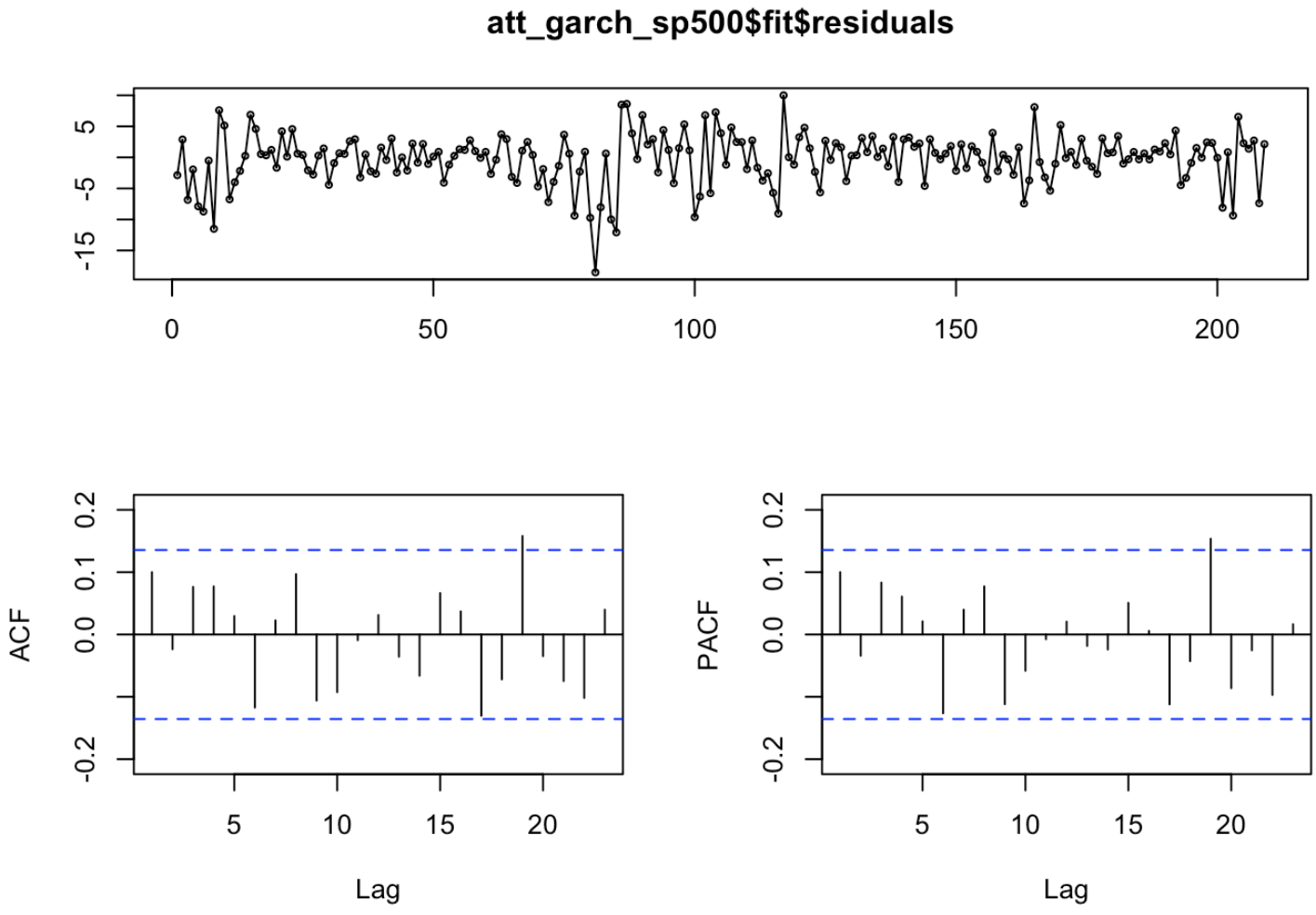
```
Box.test(att_garch_usdidr$fit$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  att_garch_usdidr$fit$residuals
## X-squared = 0.36249, df = 1, p-value = 0.5471
```

We failed to reject the null hypothesis. So there isn't enough evidence to say that the residuals are not white noise.

S&P 500 residuals

```
tsdisplay(att_garch_sp500$fit$residuals)
```



Looking at ACF and PACF, it seems there are only single spikes at lag 19. However, it is small so it should be safe to ignore it. For completeness, we run the Ljung-Box test.

```
Box.test(att_garch_sp500$fit$residuals, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  att_garch_sp500$fit$residuals
## X-squared = 2.1199, df = 1, p-value = 0.1454
```

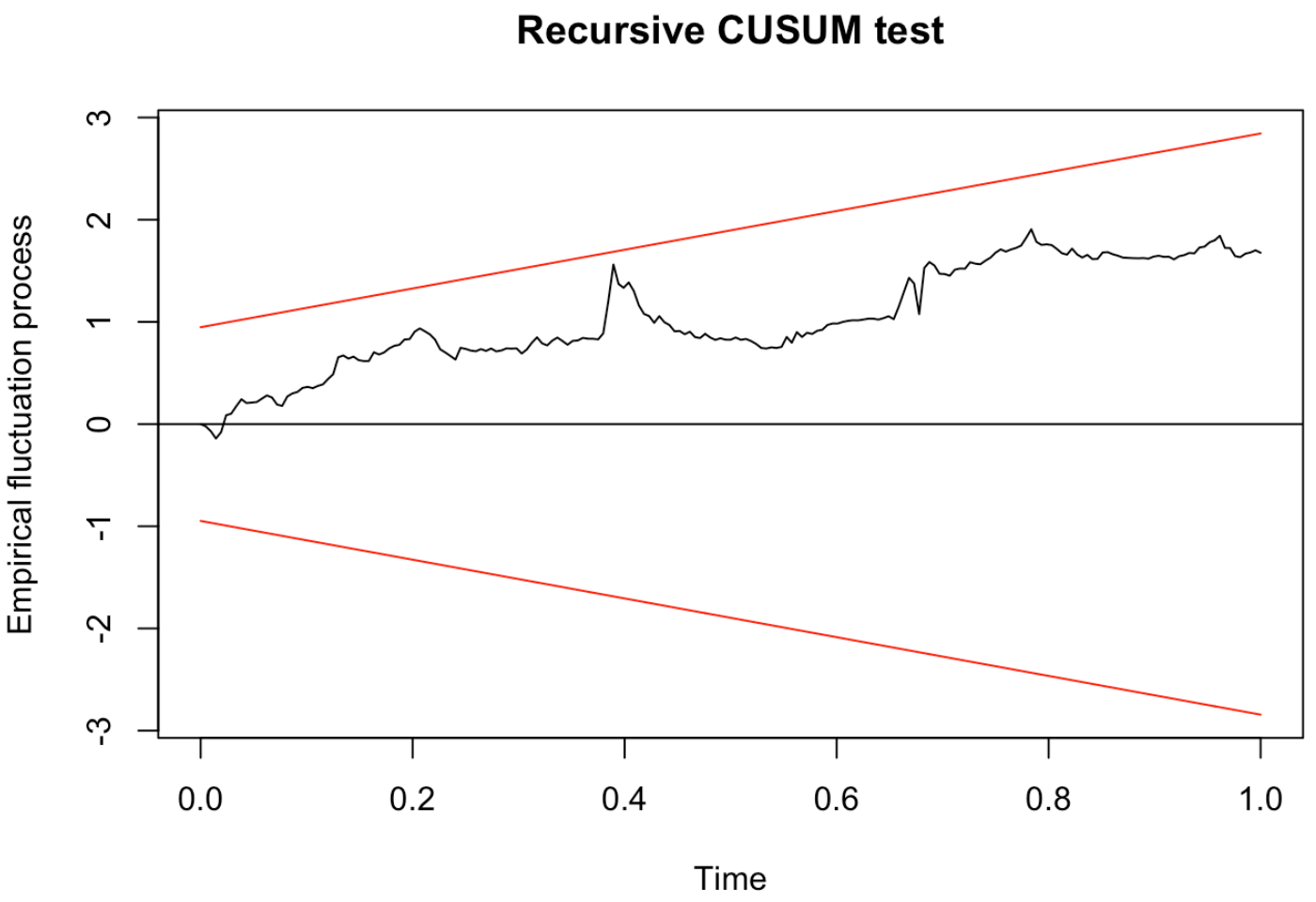
Using alpha 0.1, we failed to reject the null hypothesis, both the residuals from our model follow white noise.

f. CUSUM

For testing the parameter stability, we calculate the CUSUM plot. The red band is the interval of the stability of model.

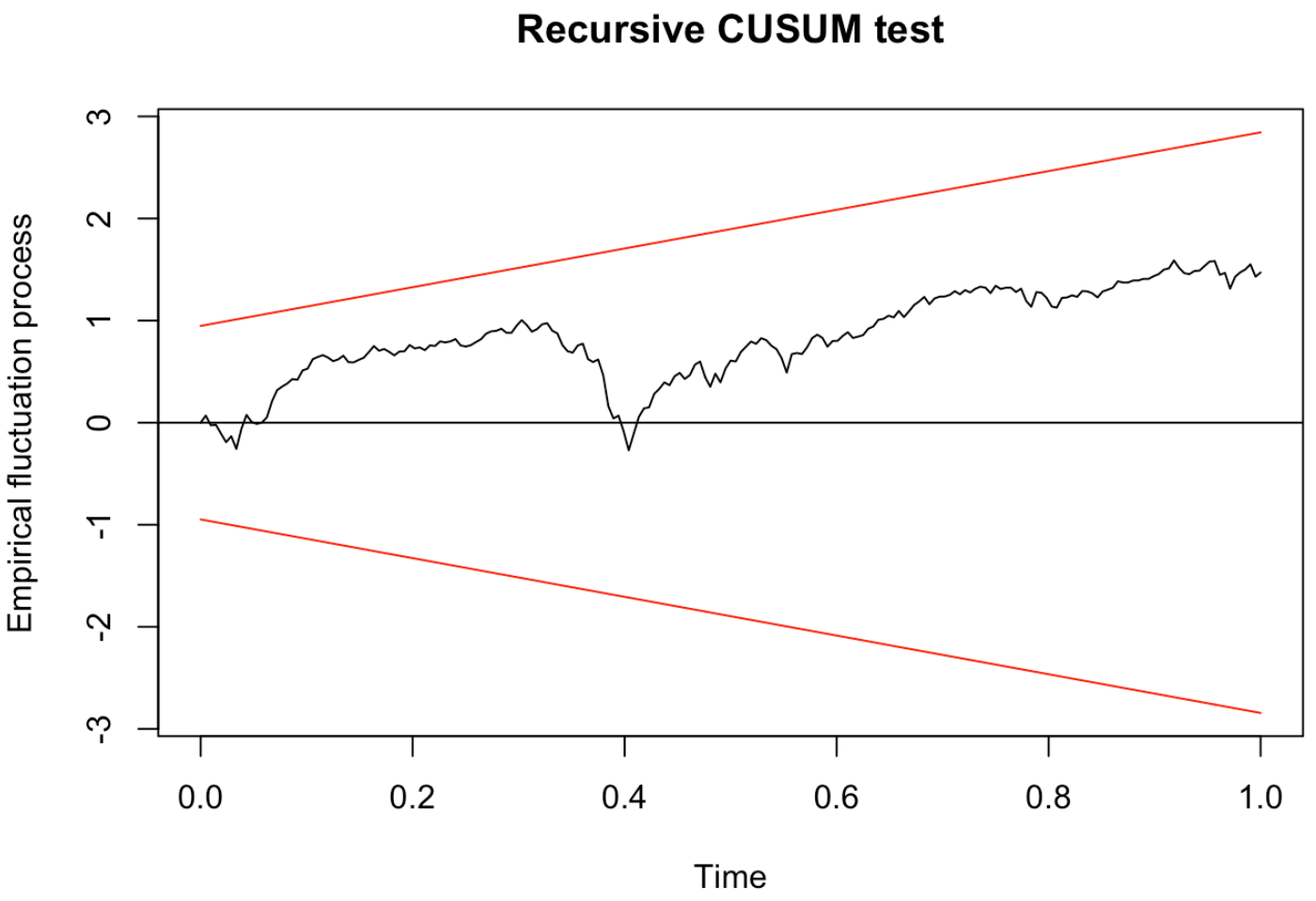
USD/IDR

```
plot(efp(att_garch_usdidr$fit$res ~ 1))
```



S&P 500

```
plot(efp(att_garch_sp500$fit$res ~ 1))
```

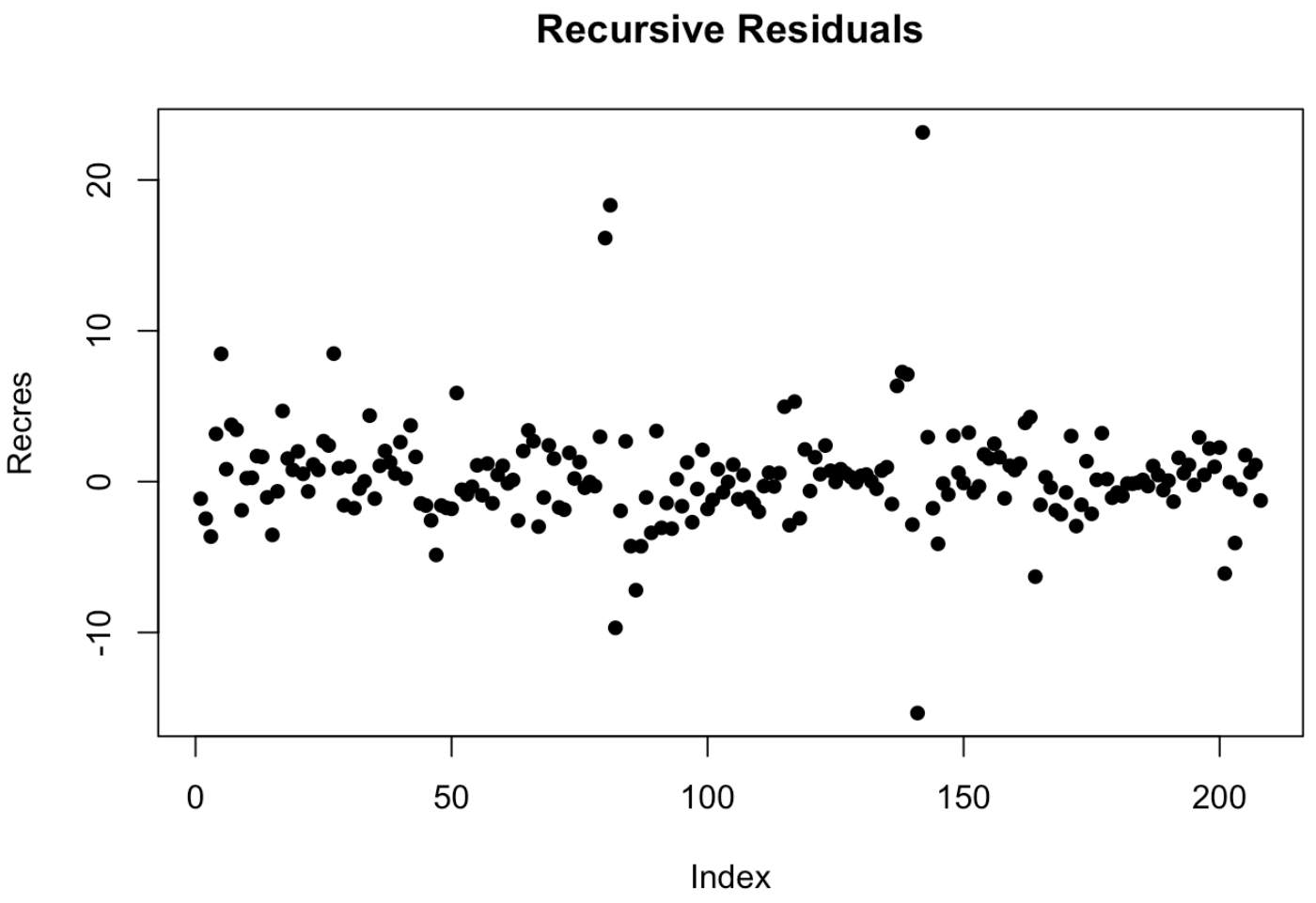


g. Recursive Residuals

The other thing to consider in validating the model is to look at its recursive residuals.

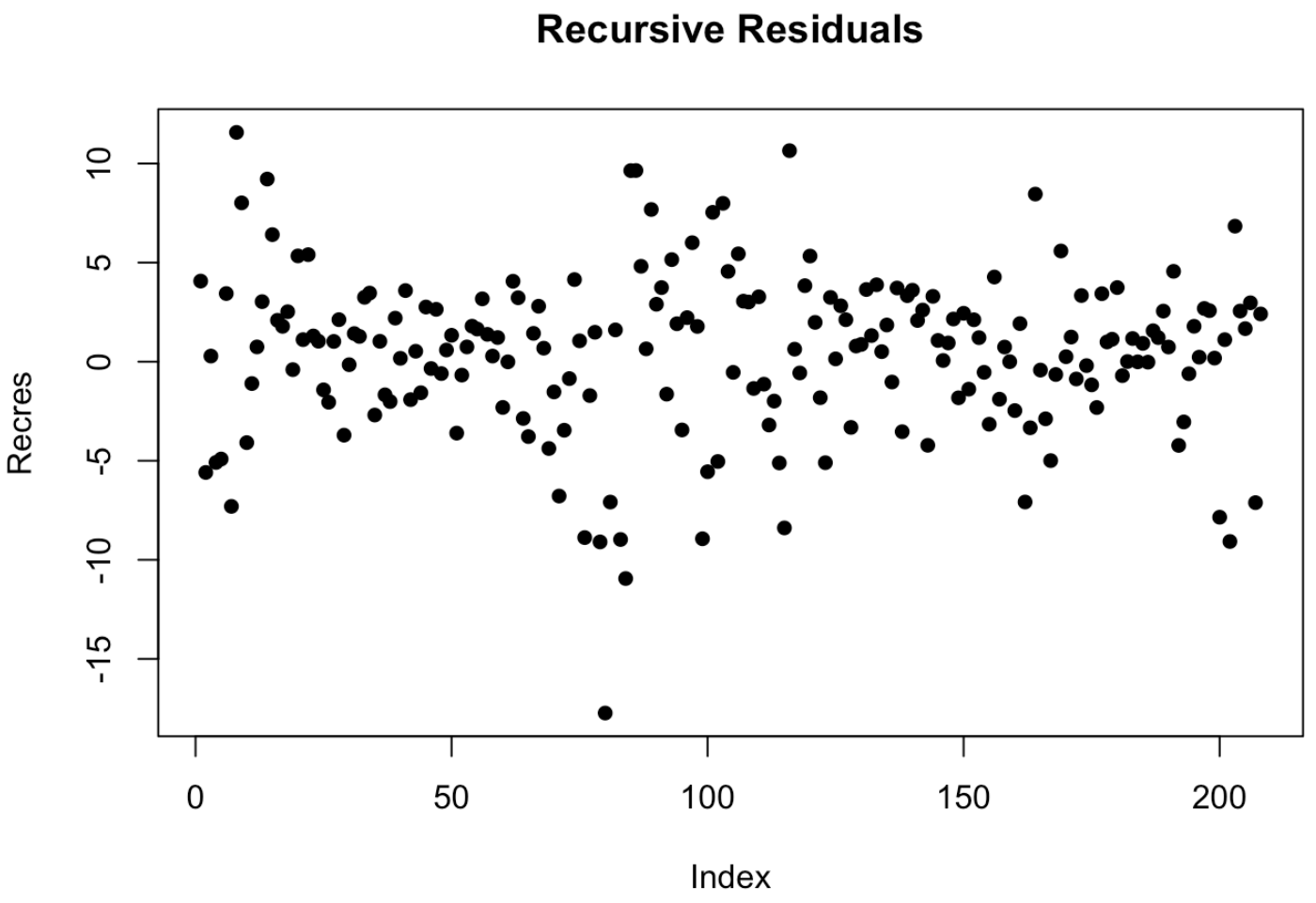
USD/IDR

```
rec_usdidr = recresid(att_garch_usdidr$fit$res ~ 1)
plot(rec_usdidr, pch = 16, main = "Recursive Residuals", ylab = "Rec
res")
```



S&P 500

```
rec_sp500 = recresid(att_garch_sp500$fit$res ~ 1)
plot(rec_sp500, pch = 16, ylab = "Recres", main = "Recursive Residua
ls")
```



h. Diagnostic statistics

In conclusion, we derive ARMA (2,2) as our best model. This, in fact is better than ARMA (2,2) with S-AR (1) which was generated using auto.arima. This is because the AIC for ARMA (2,2) is smaller and the coefficient parameter for S-AR(1) is not statistically significant. The next thing we do is to fit the best GARCH model because the squared residuals still have spikes in ACF and PACF.

By comparing AIC of all the combination p and q in GARCH (p,q), we get that GARCH(4,1) is the best for USD/IDR return and GARCH (2,1) is the best for S&P 500 returns. Finally, the standardized residuals show no significant spikes in the plot of ACF and PACF.

Seeing at the CUSUM, we see that the plot of graph for both S&P500 and USD/IDR doesn't break the band. Which implies that our model is consistent. Furthermore, the recursive residuals also tell use that our model is valid.

i. Forecast

Forecast returns

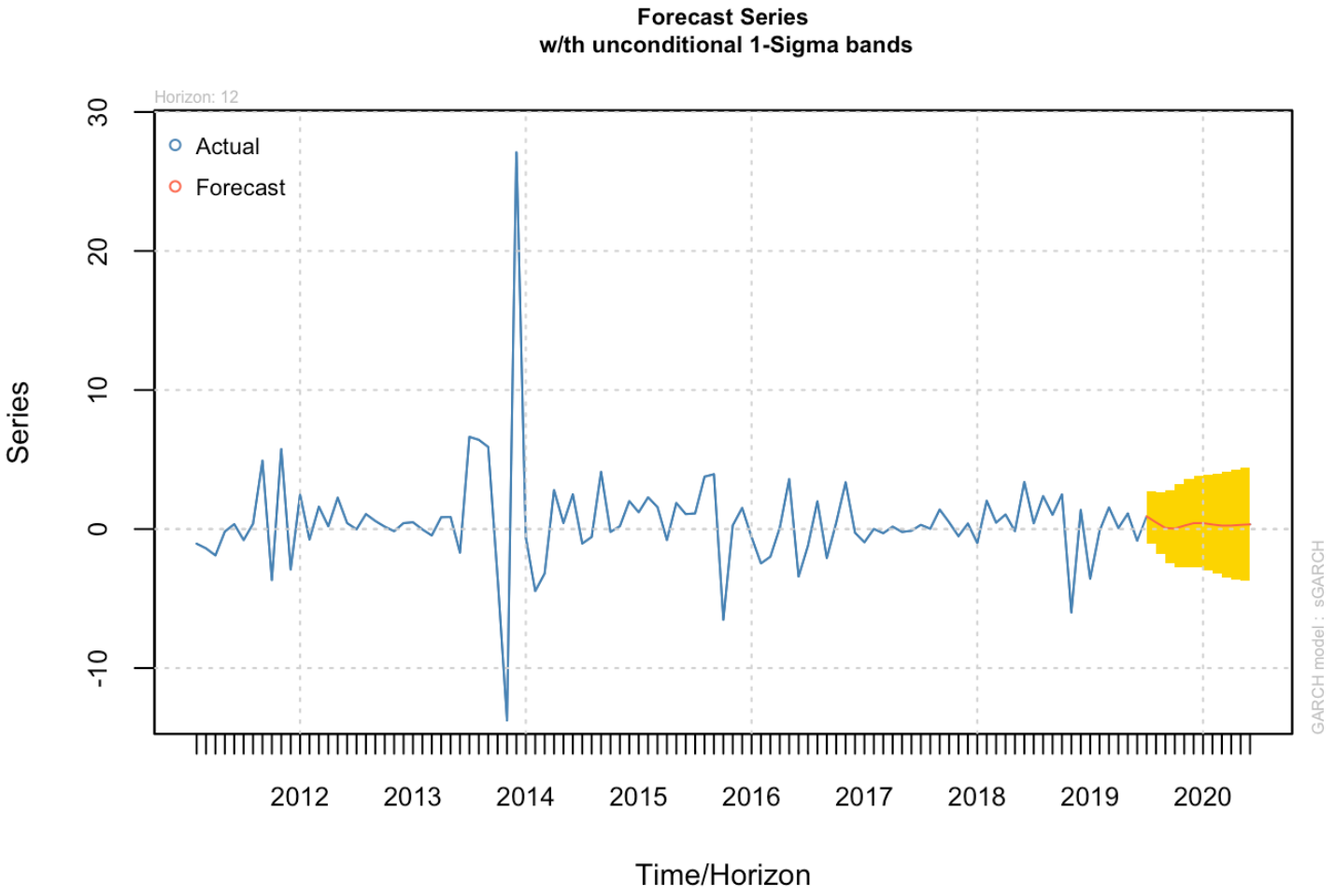
USD/IDR forecast

```
mod_for_usdidr = ugarchforecast(garchfit41_usdidr, data = NULL, n.ah
ead = 12, n.roll = 0, out.sample = 0)

# prediction
mod_for_usdidr
```

```
##
## *----------------------------------*
## *       GARCH Model Forecast       *
## *----------------------------------*
## Model: sGARCH
## Horizon: 12
## Roll Steps: 0
## Out of Sample: 0
##
## 0-roll forecast [T0=Jun 2019]:
##       Series Sigma
## T+1  0.91194 1.958
## T+2  0.48287 2.265
## T+3  0.07382 2.551
## T+4  0.03046 2.786
## T+5  0.24333 3.009
## T+6  0.42539 3.206
## T+7  0.43159 3.388
## T+8  0.32724 3.556
## T+9  0.24700 3.711
## T+10 0.25056 3.856
## T+11 0.30117 3.992
## T+12 0.33616 4.119
```

Above is the 12 step-ahead forecast. To see what's those number really meant, plotting is nessesary.

```
plot(mod_for_usdidr, which = 1)
```

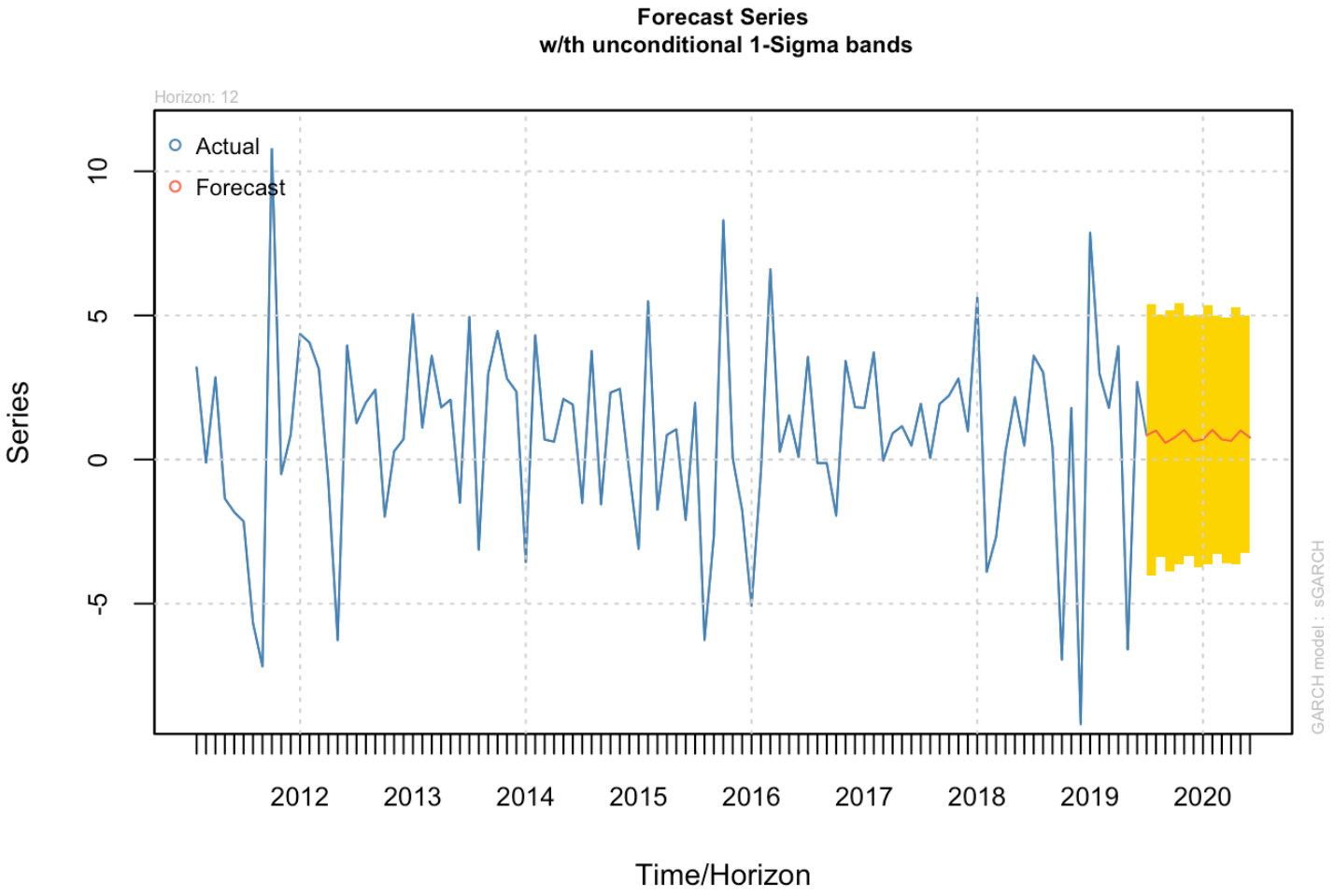


S&P 500 forecast

```
mod_for_sp500 = ugarchforecast(garchfit21_sp500, data = NULL, n.ahea
d = 12, n.roll = 0, out.sample = 0)

# prediction
mod_for_sp500
```

```
##
## *-----------------------------------*
## *       GARCH Model Forecast        *
## *-----------------------------------*
## Model: sGARCH
## Horizon: 12
## Roll Steps: 0
## Out of Sample: 0
##
## 0-roll forecast [T0=Jun 2019]:
##      Series Sigma
## T+1  0.8414 4.860
## T+2  1.0085 4.391
## T+3  0.5767 4.453
## T+4  0.7655 4.404
## T+5  1.0293 4.380
## T+6  0.6332 4.355
## T+7  0.6997 4.332
## T+8  1.0282 4.311
## T+9  0.6976 4.292
## T+10 0.6485 4.274
## T+11 1.0074 4.258
## T+12 0.7638 4.243
```

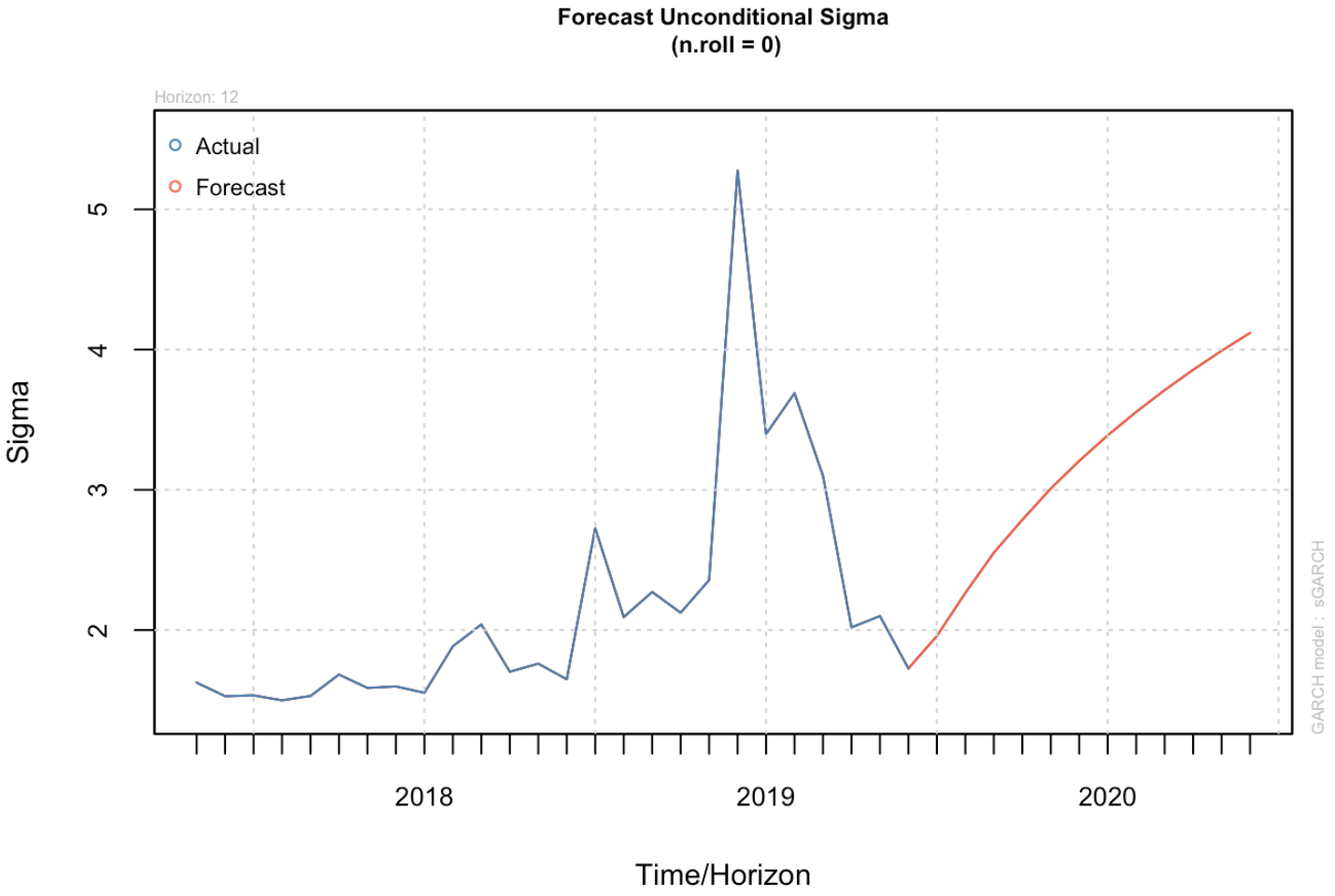
```
# plot
plot(mod_for_sp500, which = 1)
```



Forecast volatility

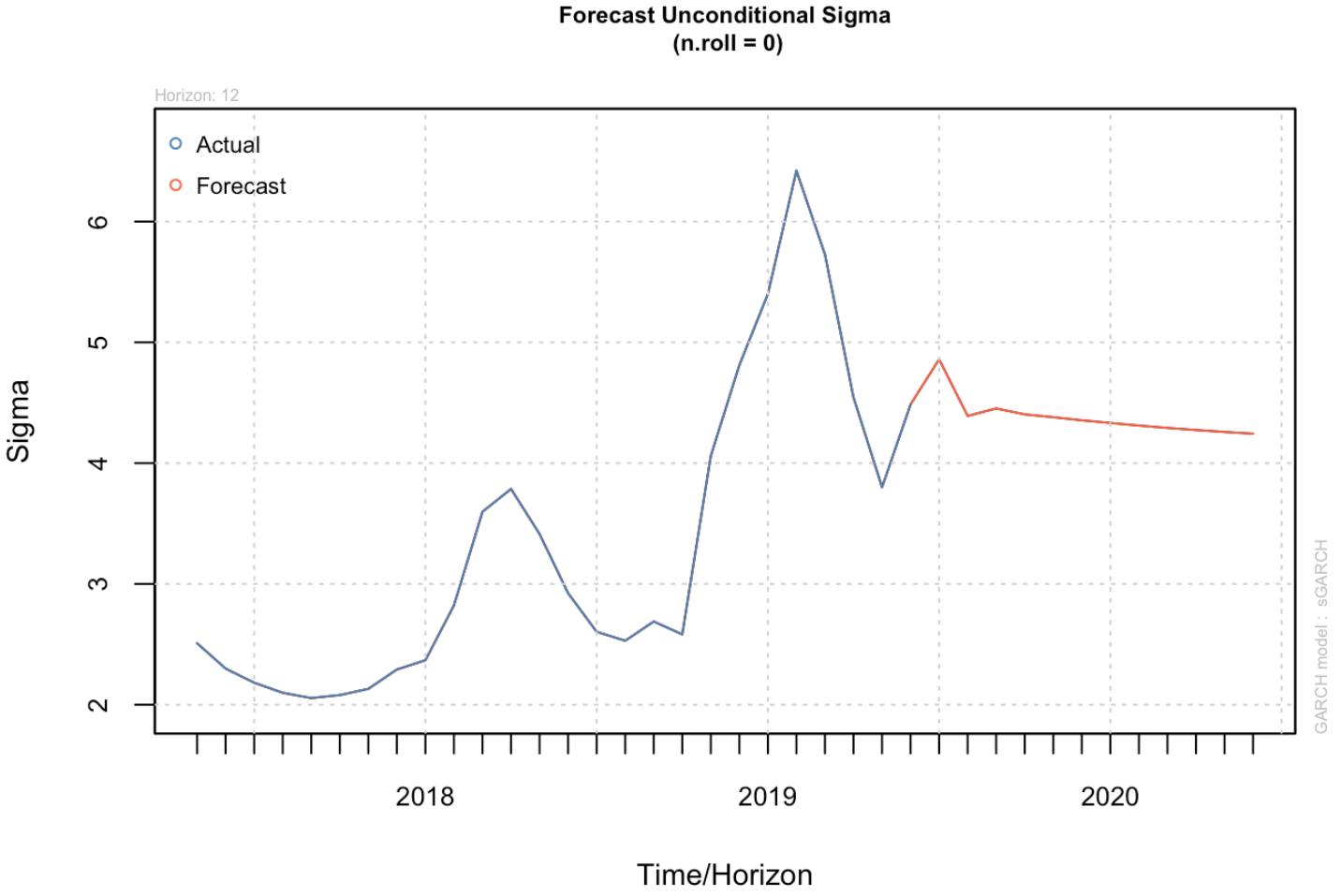
Since we're using GARCH model for both series, we can forecast their volatility. ##### Volatility for USD/IDR

```
plot(mod_for_usdidr, which = 3)
```



Volatility for S&P 500

```
plot(mod_for_sp500, which = 3)
```



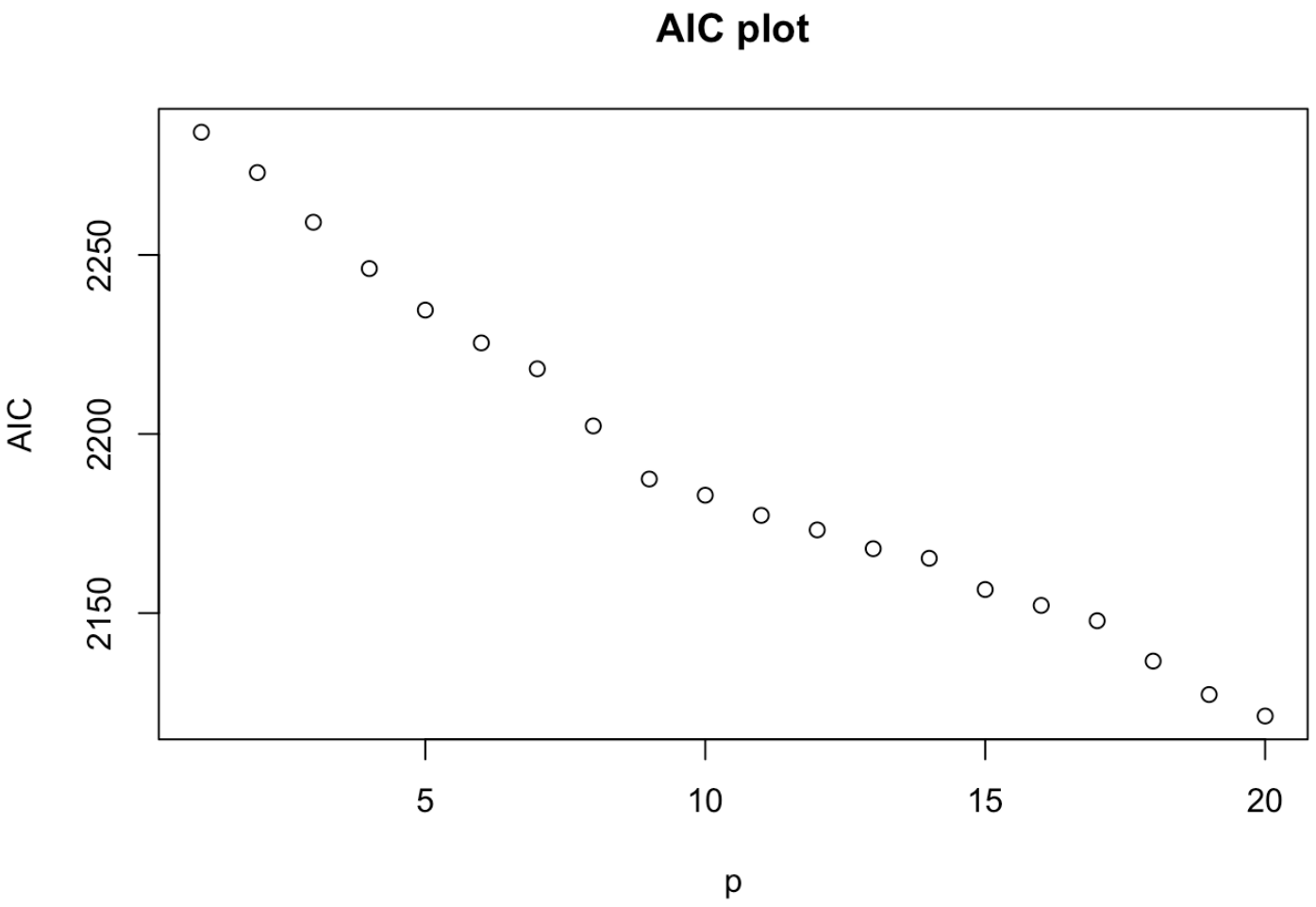
i. VAR models

Now we are considering VAR models to test our hypothesis about relationship between data. We first find the best parameter p for VAR(p) using AIC and BIC.

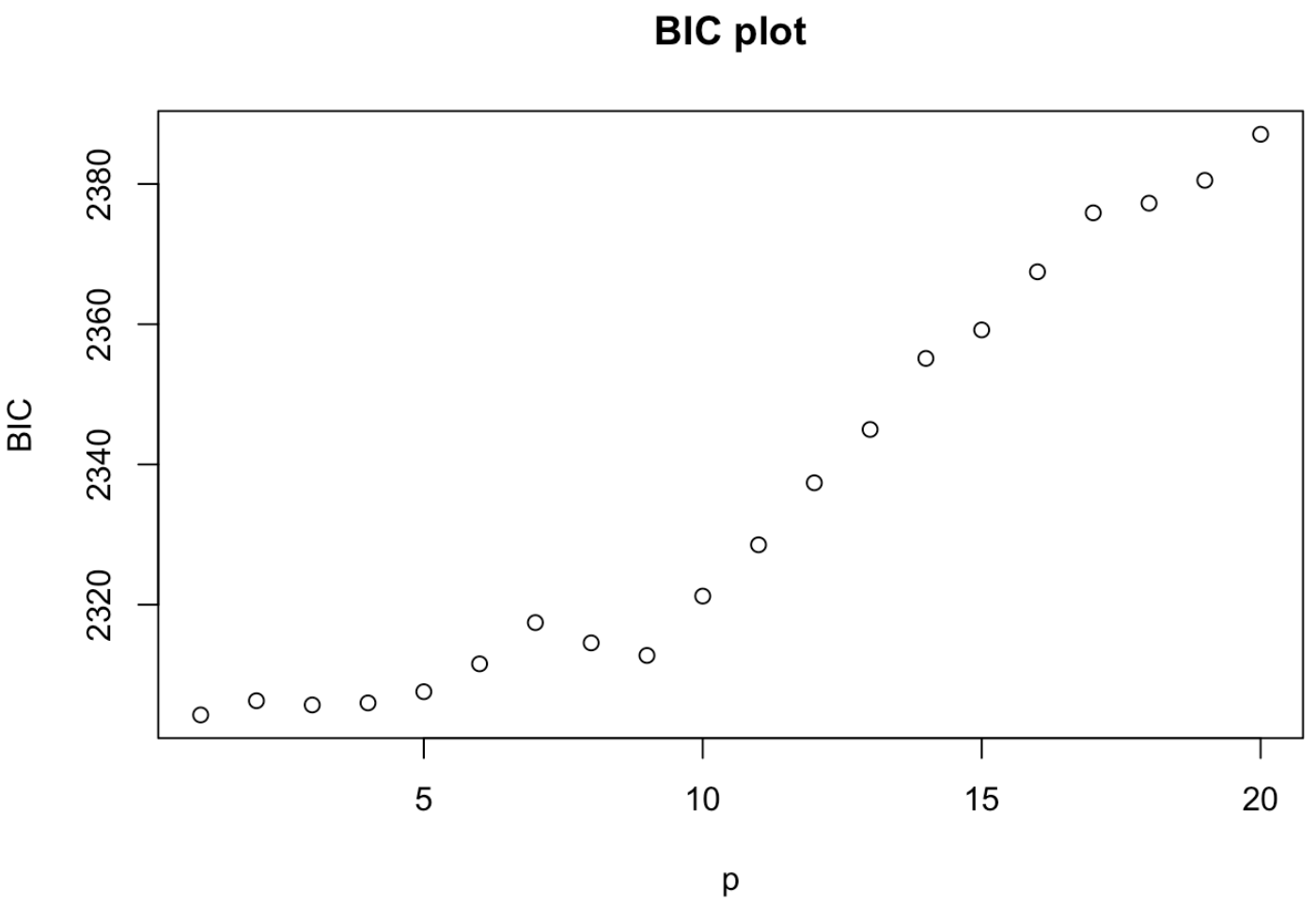
```
#VAR
varbind = cbind(sp500_ret, usd_idr_ret)

#Choose Best model by AIC BIC
row = 20
aicbic = matrix(NA, ncol=3, nrow=row)
colnames(aicbic) = c("p", "AIC", "BIC")

for(i in 1:row){
  aicbic[i, 1] = i
  aicbic[i, 2] = AIC(VAR(varbind, p=i))
  aicbic[i, 3] = BIC(VAR(varbind, p=i))
}

aicbic
```

```
##         p      AIC      BIC
##  [1,]   1 2284.256 2304.281
##  [2,]   2 2272.977 2306.304
##  [3,]   3 2259.113 2305.703
##  [4,]   4 2246.179 2305.993
##  [5,]   5 2234.595 2307.593
##  [6,]   6 2225.425 2311.568
##  [7,]   7 2218.199 2317.447
##  [8,]   8 2202.242 2314.554
##  [9,]   9 2187.427 2312.763
## [10,] 10 2182.909 2321.227
## [11,] 11 2177.282 2328.542
## [12,] 12 2173.225 2337.385
## [13,] 13 2167.973 2344.991
## [14,] 14 2165.290 2355.124
## [15,] 15 2156.580 2359.187
## [16,] 16 2152.139 2367.477
## [17,] 17 2147.861 2375.886
## [18,] 18 2136.595 2377.263
## [19,] 19 2127.253 2380.521
## [20,] 20 2121.267 2387.090
```

```
plot(aicbic[,1], aicbic[,2], xlab = "p", ylab="AIC", main="AIC plot"
)
```




```
plot(aicbic[,1], aicbic[,3], xlab = "p", ylab="BIC", main="BIC plot"
)
```



From the AIC and BIC plots, we can see that the AIC plots are always decreasing even after p>20, thus we disregard this. Looking at the BIC plot, the smallest BIC is when p=1. Therefore, we use VAR(1).

Then, we can make a VAR(1) model,

```
#Choose VAR(1)
varmod = VAR(varbind, p=1)
summary(varmod)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: sp500_ret, usd_idr_ret
## Deterministic variables: const
## Sample size: 208
## Log Likelihood: -1136.128
## Roots of the characteristic polynomial:
## 0.1138 0.07596
```

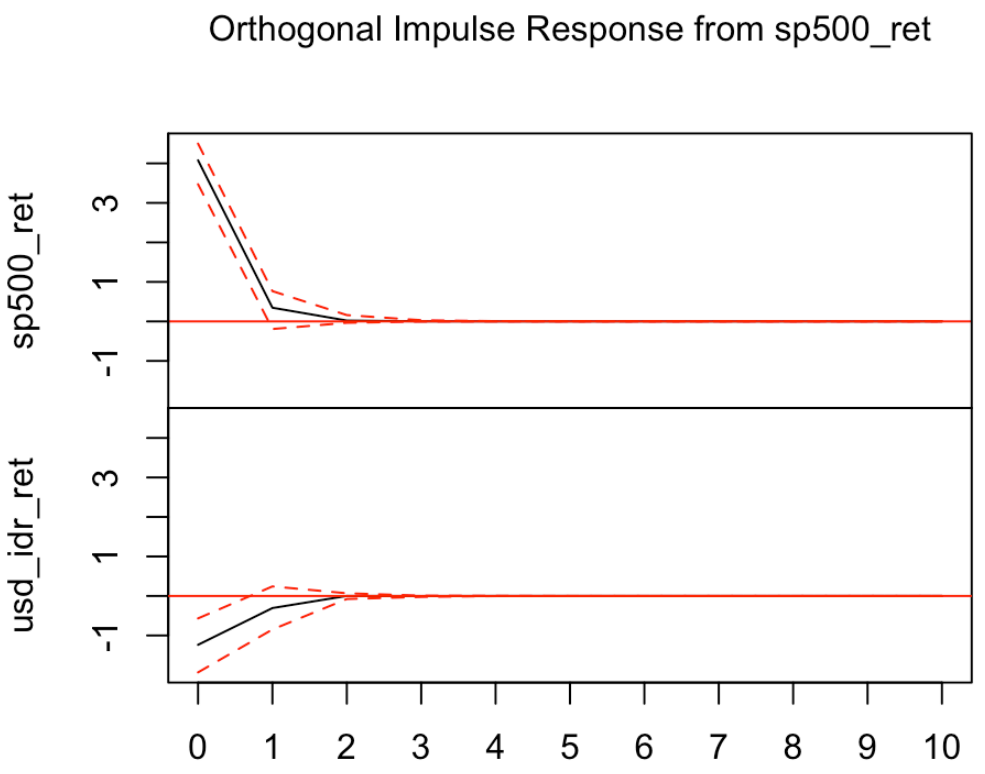
```
## Call:
## VAR(y = varbind, p = 1)
##
##
## Estimation results for equation sp500_ret:
## ==========================================
## sp500_ret = sp500_ret.l1 + usd_idr_ret.l1 + const
##
##                Estimate Std. Error t value Pr(>|t|)
## sp500_ret.l1    0.09610    0.07400   1.299   0.1955
## usd_idr_ret.l1  0.03659    0.08212   0.446   0.6564
## const           0.47837    0.28675   1.668   0.0968 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 4.078 on 205 degrees of freedom
## Multiple R-Squared: 0.00816, Adjusted R-squared: -0.001517
## F-statistic: 0.8433 on 2 and 205 DF,  p-value: 0.4318
##
##
## Estimation results for equation usd_idr_ret:
## ============================================
## usd_idr_ret = sp500_ret.l1 + usd_idr_ret.l1 + const
##
##                Estimate Std. Error t value Pr(>|t|)
## sp500_ret.l1   -0.11558    0.06619  -1.746   0.0823 .
## usd_idr_ret.l1 -0.13395    0.07345  -1.824   0.0697 .
## const           0.31793    0.25649   1.240   0.2166
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 3.648 on 205 degrees of freedom
## Multiple R-Squared: 0.02265, Adjusted R-squared: 0.01312
## F-statistic: 2.376 on 2 and 205 DF,  p-value: 0.09549
##
##
##
## Covariance matrix of residuals:
##             sp500_ret usd_idr_ret
## sp500_ret      16.632      -5.043
```

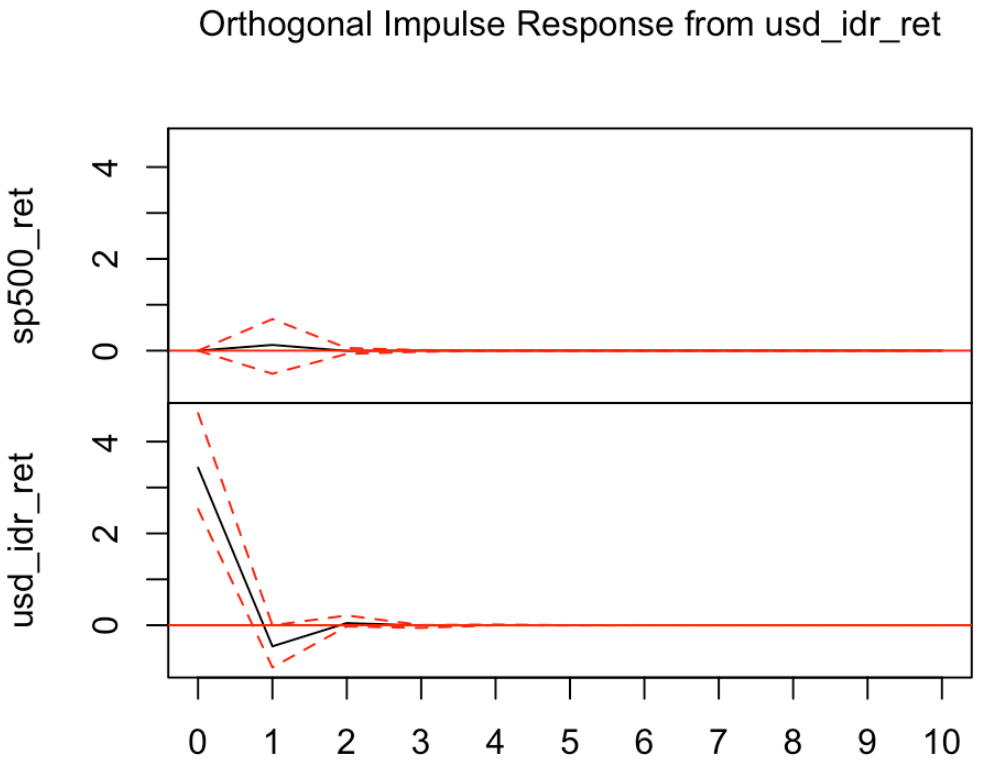
```
## usd_idr_ret    -5.043      13.307
##
## Correlation matrix of residuals:
##             sp500_ret usd_idr_ret
## sp500_ret       1.000      -0.339
## usd_idr_ret    -0.339       1.000
```

Looking at the summary, using USD/IDR Return to explain S&P500 return, we can see that the parameters are not significant. Using S&P500 to explain USD/IDR, we can only see a little bit of significance at α=0.1.

Then, we look at the Impulse Response Function (IRF) of the datas,

```
#IRF
irf_mod=irf(varmod)
plot(irf_mod)
```


Orthogonal Impulse Response from sp500_ret
sp500_ret
usd_idr_ret
-1
1
3
0
1
2
3
4
5
6
7
8
9
10
95 % Bootstrap CI, 100 runs


Orthogonal Impulse Response from usd_idr_ret
sp500_ret
usd_idr_ret
0
2
4
0
1
2
3
4
5
6
7
8
9
10

In the first graph, we can see that the effect of S&P500 return shock on itself has a high effect at first, but decays quickly to zero until lag=2. The effect of S&P500 return shock on USD/IDR return has some effect at first but again, decays to zero quickly.

In the second graph, we can also observe that USD/IDR return shock does not have any significant effect at all on S&P500 return. It has a high effect for itself but decays quickly to zero as well.

k. Granger-Causality

To analyze the causal effect of each variable to another, we perform a Granger-Causality test,

```
#Granger test 1 (H0 = USD/IDR does not explain SP500)
grangertest(sp500_ret ~ usd_idr_ret, order=1)
```

```
## Granger causality test
##
## Model 1: sp500_ret ~ Lags(sp500_ret, 1:1) + Lags(usd_idr_ret, 1:1
)
## Model 2: sp500_ret ~ Lags(sp500_ret, 1:1)
##   Res.Df Df      F Pr(>F)
## 1    205
## 2    206 -1 0.1985 0.6564
```

```
#Granger test 2 (H0 = SP500 does not explain USD/IDR)
grangertest(usd_idr_ret ~ sp500_ret, order=1)
```

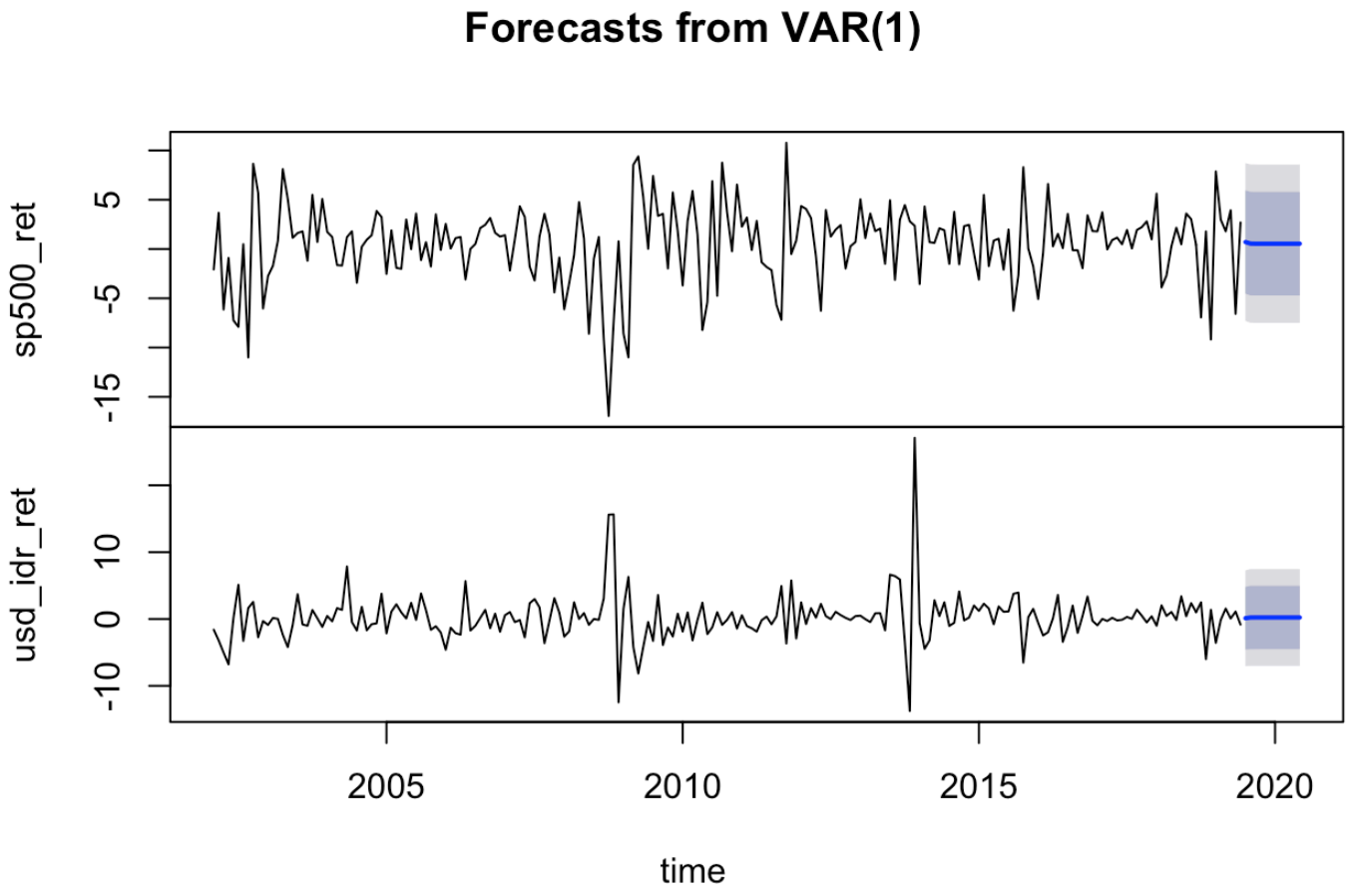
```
## Granger causality test
##
## Model 1: usd_idr_ret ~ Lags(usd_idr_ret, 1:1) + Lags(sp500_ret, 1
:1)
## Model 2: usd_idr_ret ~ Lags(usd_idr_ret, 1:1)
##   Res.Df Df      F  Pr(>F)
## 1    205
## 2    206 -1 3.0488 0.08229 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Seeing both results, the granger test cause does not have any significant in any H_0. Although there is some significant in H_0: S&P500 return does not explain USD/IDR return, it is only at α=0.1 which is not considered strongly significant.

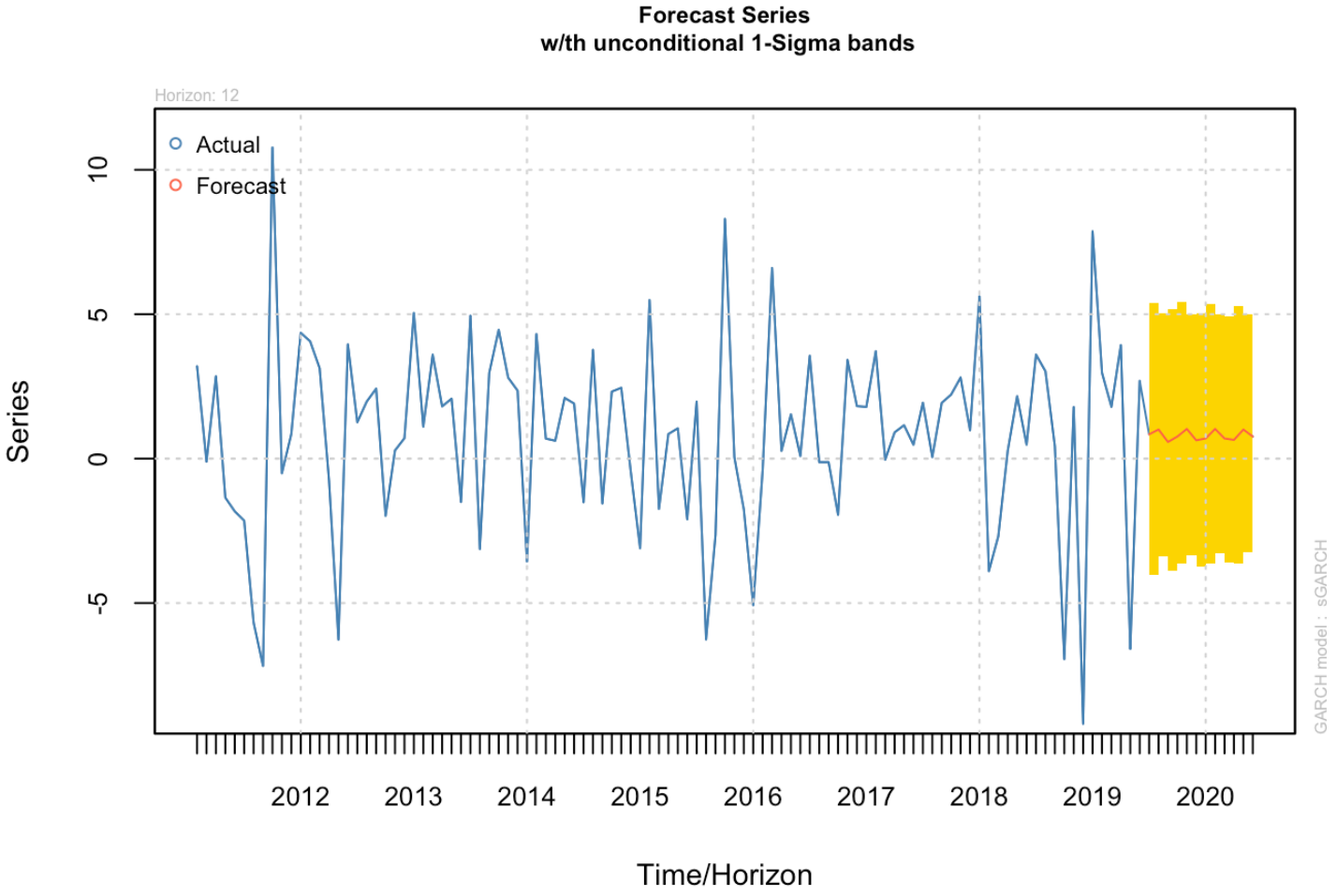
I. VAR vs. ARIMA

Comparing the predictions of VAR and GARCH which includes ARIMA.

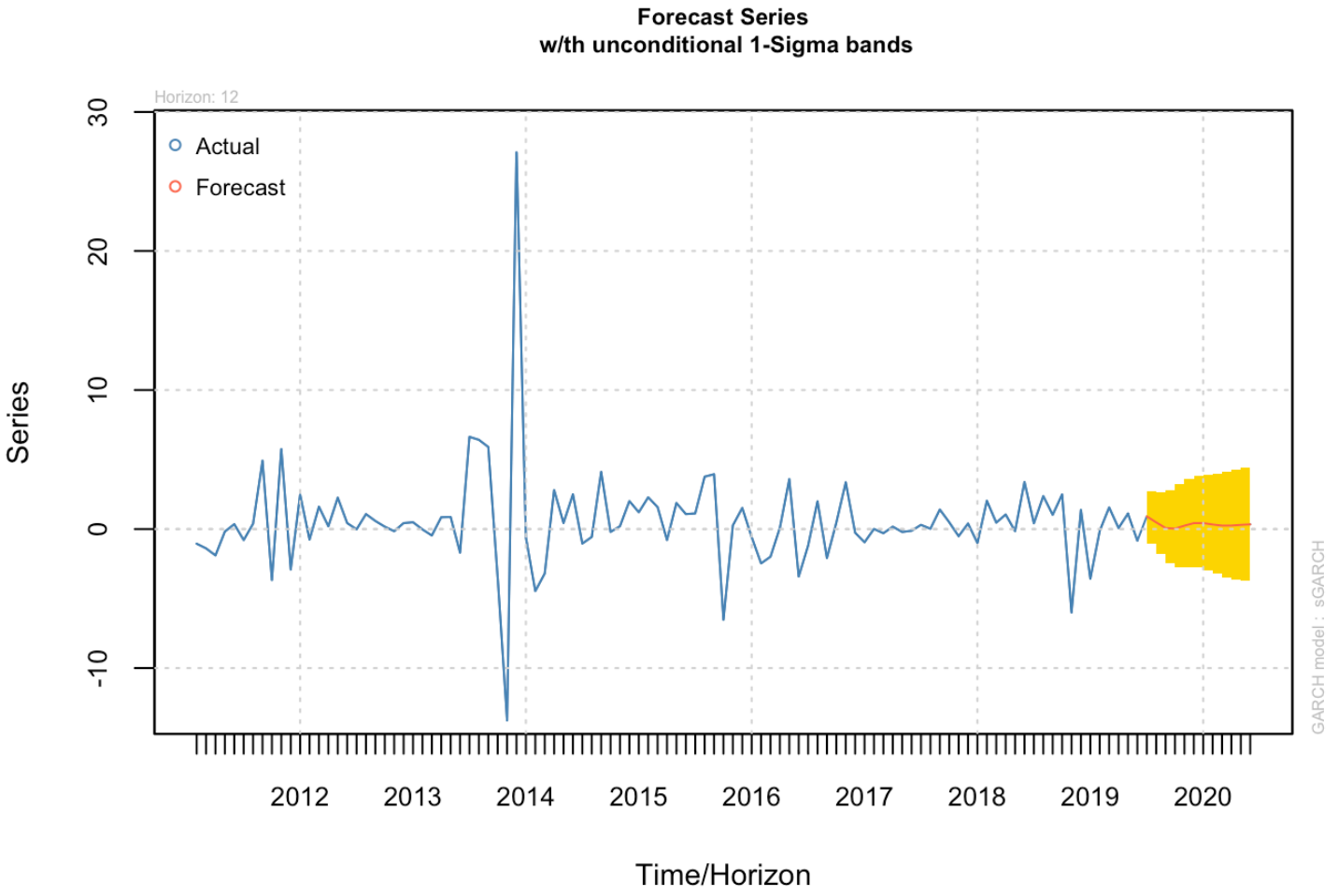
```
#Forecast using VAR
plot(forecast(varmod, h=12))
```




```
#Forecast using GARCH(4,1) and GARCH(2,1) with ARMA(2,2)
plot(mod_for_sp500, which = 1)
```




```
plot(mod_for_usdidr, which = 1)
```



III. Conclusion

Looking at the graphs, we can see more fluctuations in the GARCH forecast. This may be because the VAR model is not that good as USD/IDR return and S&P500 return does not really explain each other. This is contradictory to our initial prediciton about the returns. S&P500 return does not explain fully the USD/IDR return because there are still a lot of macroeconomics tools that we missed in this project. Furthermore, S&P500 return itself may not be a perfect representation of US Gross Domestic Product (GDP) which may be more accurate in comparing with USD/IDR return.

IV. Reference

All datas were downloaded from Yahoo Finance.