The Analysis of Hourly Weather Data

Name: Juilee Sameer Thakur

Date: 2023-05-07

Name of the course: MA 641 - Time Series Analysis

Introduction

The weather data in a certain location can give information about a lot of aspects of the region. In this project, I run a thorough analysis on the hourly data of the relative humidity of a certain region and found a certain trend in the data. I analysed it to forecast the future climate of the region based on the same.

Importing the data-set

```
data <- read.csv("C:/Users/Julee/Downloads/archive (4)/Weather Data.csv")
head(data)
```

```
##       Date.Time Temp_C Dew.Point.Temp_C Rel.Hum_. Wind.Speed_km.h Visibility_km
## 1 1/1/2012 0:00   -1.8             -3.9        86               4           8.0
## 2 1/1/2012 1:00   -1.8             -3.7        87               4           8.0
## 3 1/1/2012 2:00   -1.8             -3.4        89               7           4.0
## 4 1/1/2012 3:00   -1.5             -3.2        88               6           4.0
## 5 1/1/2012 4:00   -1.5             -3.3        88               7           4.8
## 6 1/1/2012 5:00   -1.4             -3.3        87               9           6.4
##   Press_kPa              Weather
## 1    101.24                  Fog
## 2    101.24                  Fog
## 3    101.26 Freezing Drizzle,Fog
## 4    101.27 Freezing Drizzle,Fog
## 5    101.23                  Fog
## 6    101.27                  Fog
```

This data-set is a time-series data-set of weather which has per-hour information about the weather condition at a particular location. It records Temperature, dew Point temperature, Relative humidity, Wind speed, Visibility, Pressure and conditions.

The data-set contains following columns:

- `Date.Time` - This column contains the date and time information when the data was recorded.
- `Temp_C` - This column contains the temperature information in degree Celsius.
- `Dew.Point.Temp_C` - This column contains the temperature (in Celsius) to which air must be cooled or become saturated with water vapor, assuming the air pressure and water content is constant.
- `Rel.Hum_.` - This column contains the information of amount of water vapor present in air.
- `Wind.Speed_km.h` - This column contains the speed of wind information in kmph.
- `Visibility_km` - This contains th emeasure of the horizontal opacity of the atmosphere at the point of observation in km.
- `Press_kPa` - This contains the measure of air pressure in kPa.
- `Weather` - This column contains the overall weather of the hour.

Displaying the data-set information

Below, I summarized the data to see the values of each attribute of the data-set and check for any missing values in the data-set.

```
summary(data)
```

```
##   Date.Time             Temp_C        Dew.Point.Temp_C    Rel.Hum_.
##  Length:8784        Min.   :-23.300   Min.   :-28.500   Min.   : 18.00
##  Class :character   1st Qu.:  0.100   1st Qu.: -5.900   1st Qu.: 56.00
##  Mode  :character   Median :  9.300   Median :  3.300   Median : 68.00
##                     Mean   :  8.798   Mean   :  2.555   Mean   : 67.43
##                     3rd Qu.: 18.800   3rd Qu.: 11.800   3rd Qu.: 81.00
##                     Max.   : 33.000   Max.   : 24.400   Max.   :100.00
##  Wind.Speed_km.h Visibility_km     Press_kPa        Weather
##  Min.   : 0.00   Min.   : 0.20   Min.   : 97.52   Length:8784
##  1st Qu.: 9.00   1st Qu.:24.10   1st Qu.:100.56   Class :character
##  Median :13.00   Median :25.00   Median :101.07   Mode  :character
##  Mean   :14.95   Mean   :27.66   Mean   :101.05
##  3rd Qu.:20.00   3rd Qu.:25.00   3rd Qu.:101.59
##  Max.   :83.00   Max.   :48.30   Max.   :103.65
```

I found no missing values in the data-set. I am keen on knowing the precipitation in the region through-out the mentioned time period. This information could be extracted easily knowing the relative humidity data. Hence, the desired attribute of interest is “Rel. Hum_.”

Checking for seasonality

To check if the Relative Humidity data is seasonal or not, I run the combined test from the “seastests” library. The combined test can be used individually or using the *isSeasonal()* function. So, I ran both tests.

```
library(seastests)
```

```
## Warning: package 'seastests' was built under R version 4.2.3
```

```
# Combined test
combined_test(data$Rel.Hum_., freq = 12)
```

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

```
## Test used:  WO
##
## Test statistic:  1
## P-value:  1 1 0
```

```
# Seasonal or not Test
cat("\nThe result of isSeasonal test is",isSeasonal(data$Rel.Hum_., test = "combined", freq = 12))
```

```
##
## The result of isSeasonal test is TRUE
```

The combined test run WO test. By default, the WO-test combines the results of the QS-test and the kw-test. Here, the over-all p-value of the test is equal to 0 which means the data is seasonal.

The *isSeasonal()* test runs the combined-test to assess the seasonality of a time series and returns a boolean. In this case, it retured true which confirms the data is seasonal.

Exploratory Data Analysis

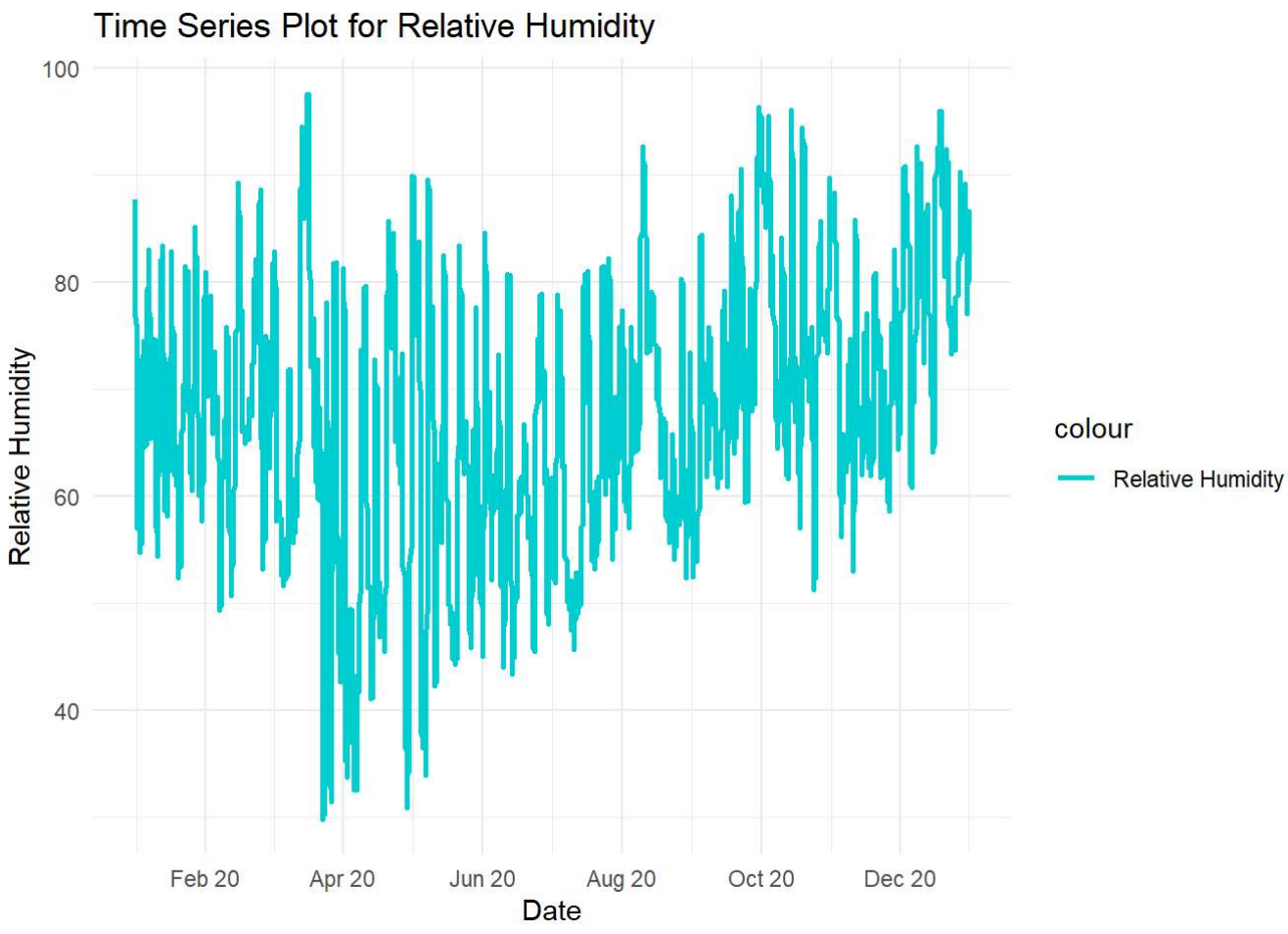
In order to check the data trend and seasonality, I plot the time series of the dataset

```
options(warn = -1)

library(ggplot2)
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

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

```
data$Date.Time <- as.Date(data$Date.Time, "%m/%d/%y")

data$rolling_avg <- rollapply(data$Rel.Hum_., width = 24, FUN = mean, fill = NA, align = "center", partial = TRUE)

ggplot(data, aes(x = Date.Time)) +
  geom_line(aes(y = rolling_avg, color = "Relative Humidity"), size = 1) +
  labs(x = "Date", y = "Relative Humidity",
       title = "Time Series Plot for Relative Humidity") +
  scale_x_date(date_breaks = "2 month", date_labels = "%b %y") +
  scale_color_manual(values = c("Relative Humidity" = "darkturquoise")) +
  theme_minimal()
```



We can see the seasonality in the above plot throughout the year data.

Checking the stationarity

To check the stationarity of the data, I run the Augmented Dickey-Fuller test.

Following the hypothesis for ADF test:

H_0 : The series is non-stationary

vs

H_1 : The series is stationary

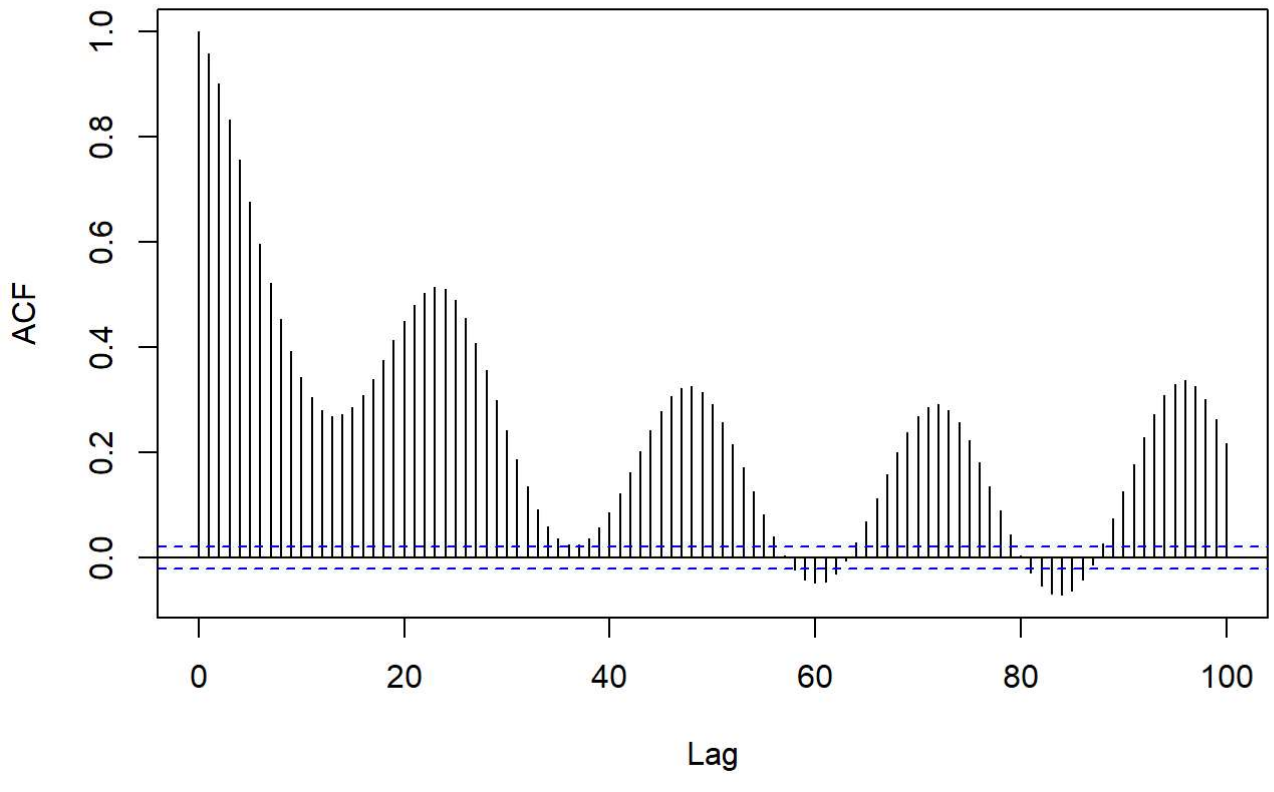
```
library(tseries)

adf.test(data$Rel.Hum_.)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data$Rel.Hum_.
## Dickey-Fuller = -9.0452, Lag order = 20, p-value = 0.01
## alternative hypothesis: stationary
```

The p-value displayed in 0.01 which is significantly lesser than 0.05. Hence, we reject the H_0 and say that the data is stationary.

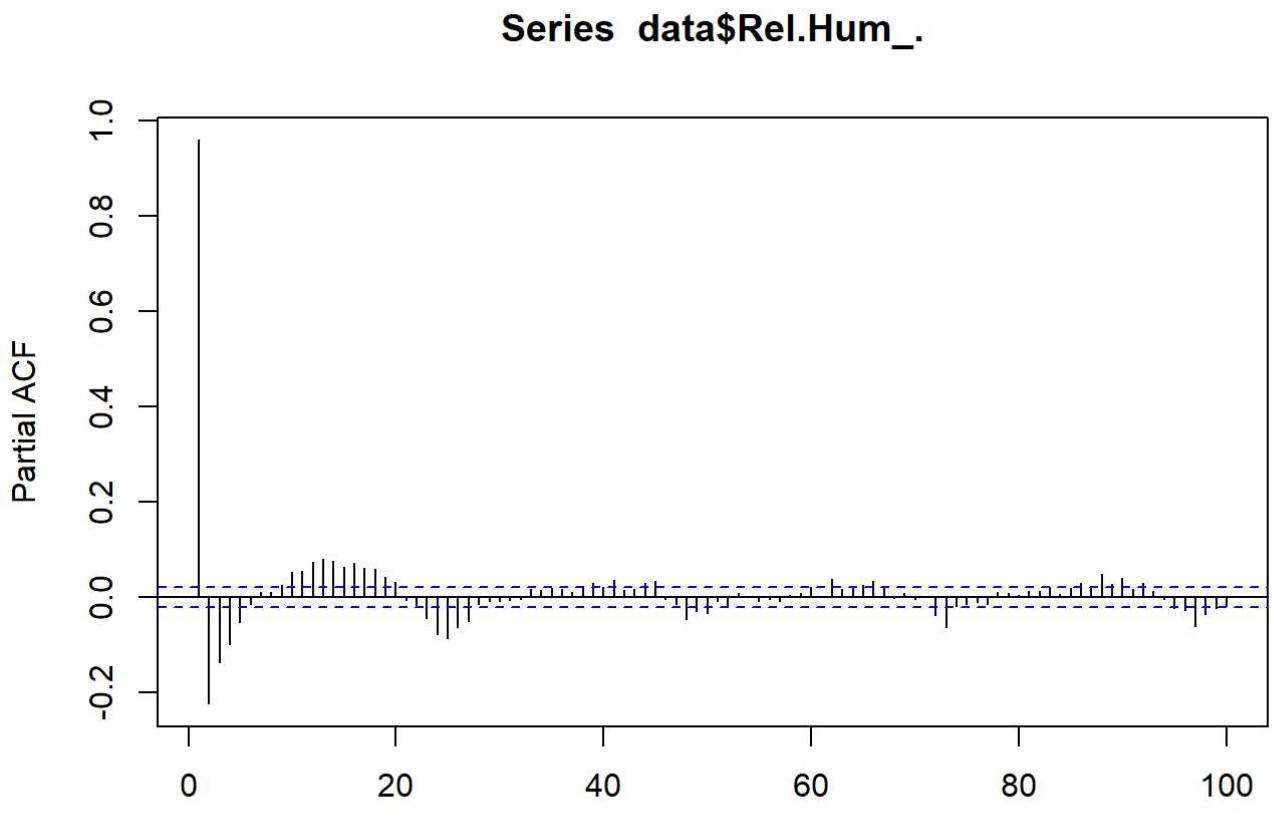
Plotting the ACF and PACF of the data

```
acf(data$Rel.Hum_., lag.max = 100)
```



Since, the lags are exponentially decaying we can consider the value to be 0 or possibly 1. Since, we do not use differencing, component d = 0 throughout.

```
pacf(data$Rel.Hum_., lag.max = 100)
```



The significant lags in the PACF plot seem to be in a range of values that can possibly give value of component p to be from 1 to 5.

```
library(TSA)
```

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

```
##
## Attaching package: 'TSA'
```

```
## The following objects are masked from 'package:stats':
##
##     acf, arima
```

```
## The following object is masked from 'package:utils':
##
##     tar
```

```
eacf(data$Rel.Hum_.)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x x x x x x x x  x  x  x
## 1 x x x x x x x o x x x  x  x  x
## 2 x o o o o o x o o o o  o  o  o
## 3 x x o o x o o o o o o  o  o  o
## 4 x x x o o o o o o o o  o  o  o
## 5 x x x o o o o o o o o  o  o  o
## 6 x x x x x o o o o o o  o  o  o
## 7 x x o x x x o o o o o  o  o  o
```

The EACF of the data suggests the model to be ARMA (2,1) or ARMA(3,1).

Thus models suggested are:

- Model 1: ARMA (5,0)
- Model 2: ARMA (4,0)
- Model 3: ARMA (3,0)
- Model 4: ARMA (2,0)
- Model 5: ARMA (1,0)

Parameter Estimation

```
options(warn = -1)

library(forecast)

model_1 <- Arima(data$Rel.Hum_., order = c(5,0,0))
model_2 <- Arima(data$Rel.Hum_., order = c(4,0,0))
model_3 <- Arima(data$Rel.Hum_., order = c(3,0,0))
model_4 <- Arima(data$Rel.Hum_., order = c(2,0,0))
model_5 <- Arima(data$Rel.Hum_., order = c(1,0,0))
```

```
cat("For Model 1, AIC:", model_1$aic, "and BIC :", model_1$bic)
```

```
## For Model 1, AIC: 51755.34 and BIC : 51804.9
```

```
cat("\nFor Model 2, AIC:", model_2$aic, "and BIC :", model_2$bic)
```

```
##
## For Model 2, AIC: 51776.99 and BIC : 51819.47
```

```
cat("\nFor Model 3, AIC:", model_3$aic, "and BIC :", model_3$bic)
```

```
##
## For Model 3, AIC: 51861.62 and BIC : 51897.03
```

```
cat("\nFor Model 4, AIC:", model_4$aic, "and BIC :", model_4$bic)
```

```
##
## For Model 4, AIC: 52021.93 and BIC : 52050.25
```

```
cat("\nFor Model 5, AIC:", model_5$aic, "and BIC :", model_5$bic)
```

```
##
## For Model 5, AIC: 52469.43 and BIC : 52490.67
```

Model 1 i.e. ARMA(5,0) has the lowest AIC and BIC values. Hence, I stick that with the non-seaonal part of the SARMA and run different models changing the seasonal order.

```
sarma_1 <- Arima(data$Rel.Hum_., order = c(5,0,0), seasonal = list(order = c(5,0,0), period = 12))
sarma_2 <- Arima(data$Rel.Hum_., order = c(5,0,0), seasonal = list(order = c(4,0,0), period = 12))
sarma_3 <- Arima(data$Rel.Hum_., order = c(5,0,0), seasonal = list(order = c(3,0,0), period = 12))
sarma_4 <- Arima(data$Rel.Hum_., order = c(5,0,0), seasonal = list(order = c(2,0,0), period = 12))
sarma_5 <- Arima(data$Rel.Hum_., order = c(5,0,0), seasonal = list(order = c(1,0,0), period = 12))
```

```
cat("For SARMA Model 1, AIC:", sarma_1$aic, "and BIC :", sarma_1$bic)
```

```
## For SARMA Model 1, AIC: 51543.06 and BIC : 51628.02
```

```
cat("\nFor SARMA Model 2, AIC:", sarma_2$aic, "and BIC :", sarma_2$bic)
```

```
##
## For SARMA Model 2, AIC: 51550.76 and BIC : 51628.65
```

```
cat("\nFor SARMA Model 3, AIC:", sarma_3$aic, "and BIC :", sarma_3$bic)
```

```
##
## For SARMA Model 3, AIC: 51603.49 and BIC : 51674.3
```

```
cat("\nFor SARMA Model 4, AIC:", sarma_4$aic, "and BIC :", sarma_4$bic)
```

```
##
## For SARMA Model 4, AIC: 51611.1 and BIC : 51674.82
```

```
cat("\nFor SARMA Model 5, AIC:", sarma_5$aic, "and BIC :", sarma_5$bic)
```

```
##
## For SARMA Model 5, AIC: 51742.63 and BIC : 51799.28
```

The SARMA model 1 has lowest AIC and BIC. Therefore, our best model is $SARIMA(5,0,0)(5,0,0)_{12}$.

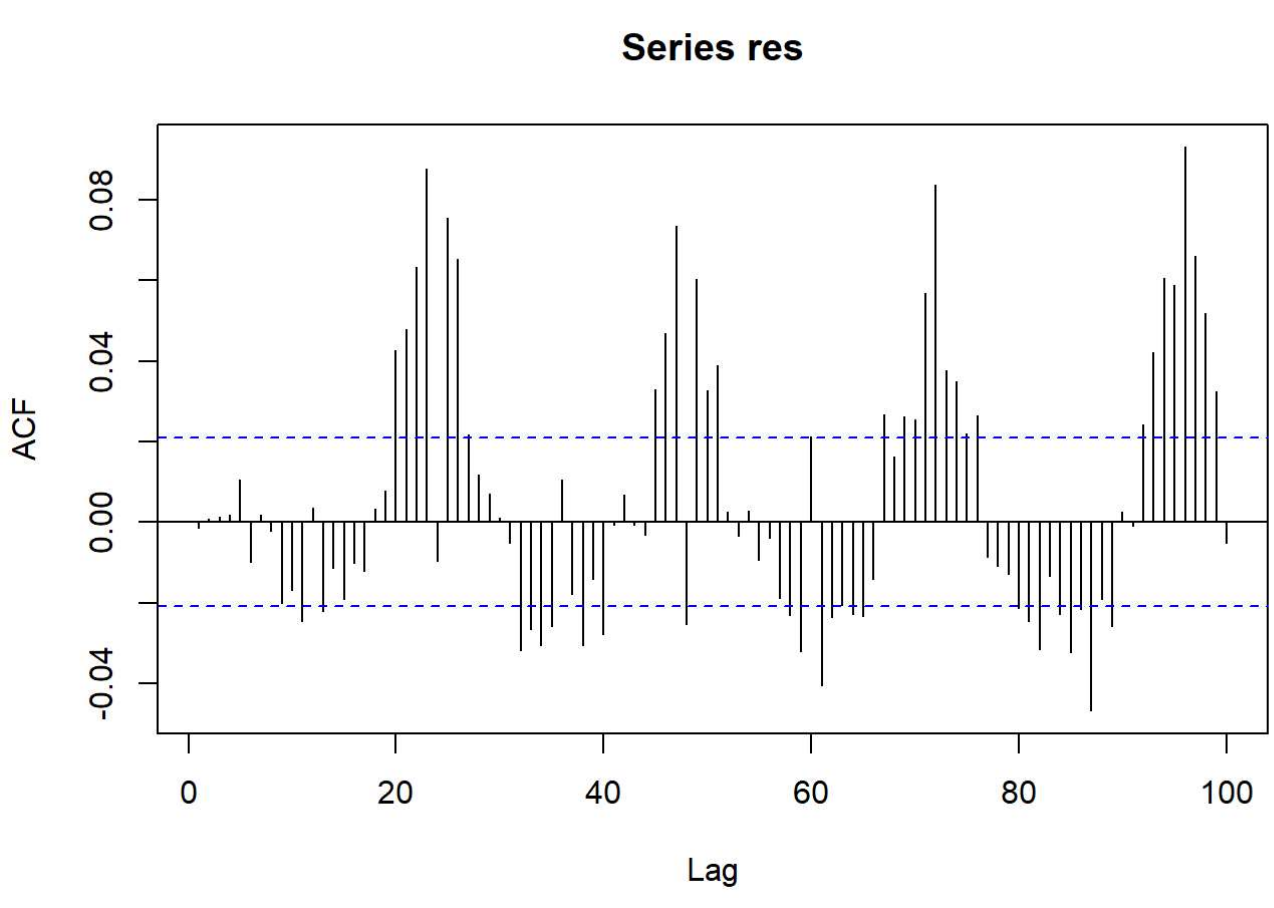
```
summary(sarma_1)
```

```
## Series: data$Rel.Hum_.
## ARIMA(5,0,0)(5,0,0)[12] with non-zero mean
##
## Coefficients:
##          ar1      ar2      ar3      ar4      ar5     sar1    sar2     sar3
##       1.0756  -0.0558  -0.0154  -0.0289  -0.0353  -0.0494  0.1216  -0.0314
## s.e.     NaN      NaN      NaN   0.0025   0.0014      NaN  0.0012      NaN
##         sar4     sar5     mean
##       0.0801  -0.0323  67.4318
## s.e.  0.0018      NaN   0.8893
##
## sigma^2 = 20.66:  log likelihood = -25759.53
## AIC=51543.06   AICc=51543.09   BIC=51628.02
##
## Training set error measures:
##                       ME     RMSE      MAE        MPE    MAPE      MASE
## Training set 0.00022863 4.542689 3.161664 -0.5926368 5.10978 0.9401074
##                      ACF1
## Training set -0.00150183
```

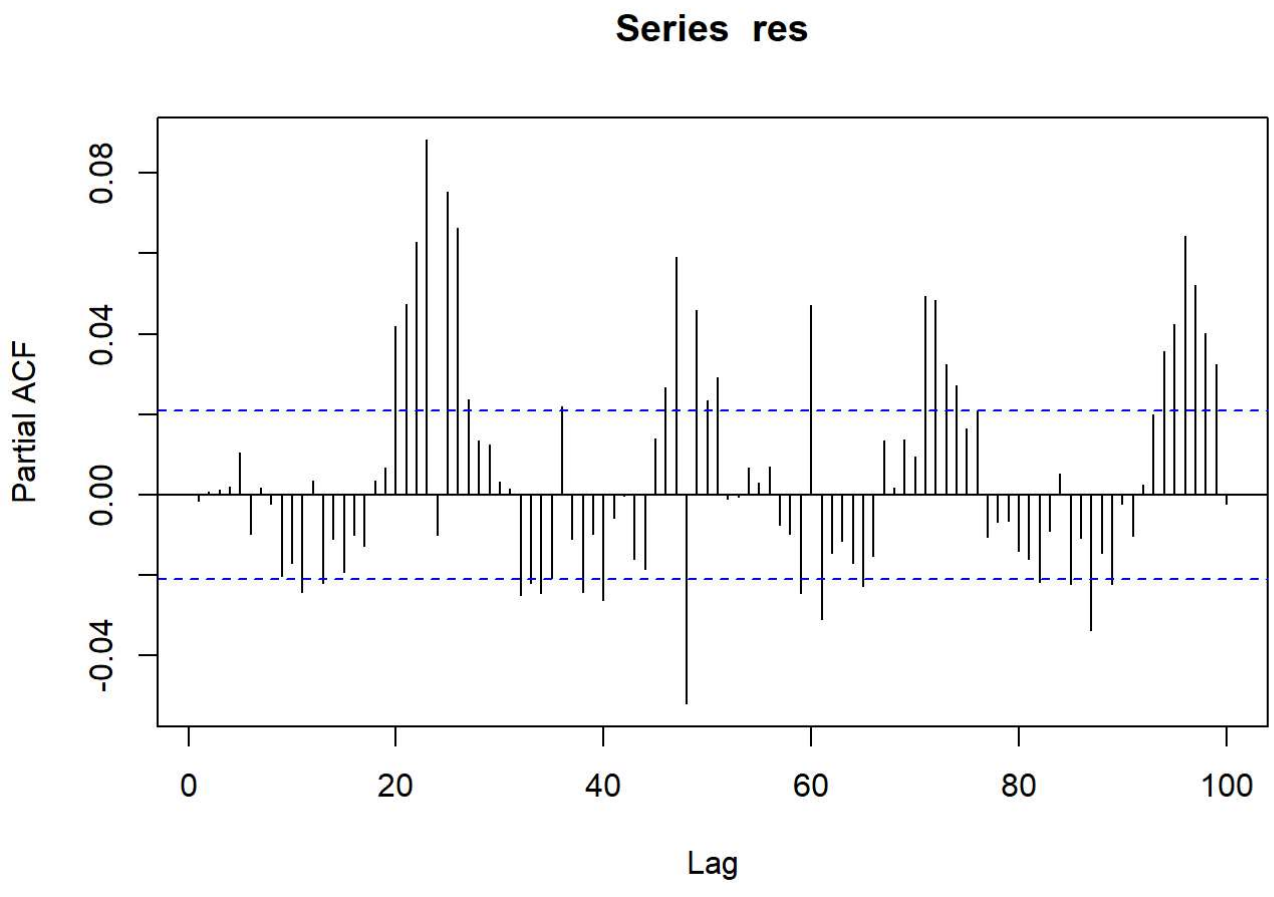
Residual Analysis

Using the above obtained model, I performed the residual analysis and plotted the same along with its ACF and PACF.

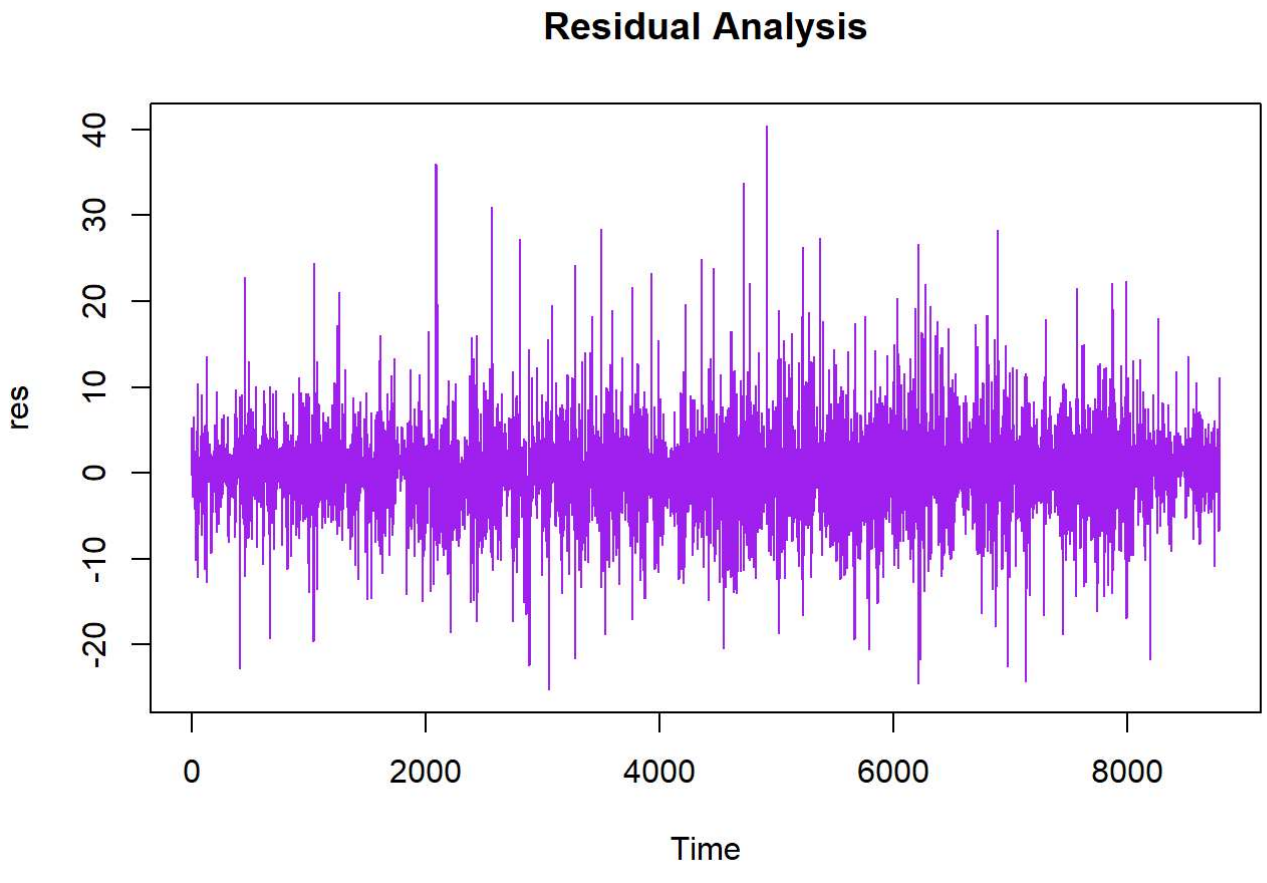
```
res <- sarma_1$residuals
acf(res,lag.max = 100)
```




```
pacf(res, lag.max = 100)
```




```
ts.plot(res,lwd=1,col="purple",main='Residual Analysis')
```



The ACF and PACF plot do no indicate white noise a such and hence, it is not ideal to forecast using the same. Thus, I decide to run the GARCH model.

GARCH Modelling

I ran the sGARCH model on orders (5,0), (4,0) and (3,0) and plotted the summaries and plots.

```
library(rugarch)
```

```
## Loading required package: parallel
```

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

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

```
return = diff(log(data$Rel.Hum_.))

s_norm <- ugarchspec(mean.model = list(armaOrder = c(5,0)),
                     variance.model = list(model = 'sGARCH'),
                     distribution.model = 'norm')

m_norm <- ugarchfit(data = return, spec = s_norm, solver ='hybrid')
m_norm
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(5,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.002111    0.000990  2.13239 0.032975
## ar1     0.242757    0.013315 18.23123 0.000000
## ar2     0.121713    0.012923  9.41826 0.000000
## ar3     0.054564    0.012480  4.37202 0.000012
## ar4    -0.004469    0.011963 -0.37357 0.708726
## ar5    -0.049573    0.011325 -4.37729 0.000012
## omega   0.000389    0.000046  8.42988 0.000000
## alpha1  0.280838    0.021995 12.76811 0.000000
## beta1   0.709597    0.020166 35.18765 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.002111    0.001038  2.03344 0.042008
## ar1     0.242757    0.020444 11.87406 0.000000
## ar2     0.121713    0.016424  7.41069 0.000000
## ar3     0.054564    0.013045  4.18262 0.000029
## ar4    -0.004469    0.014656 -0.30493 0.760423
## ar5    -0.049573    0.015169 -3.26795 0.001083
## omega   0.000389    0.000183  2.12114 0.033910
## alpha1  0.280838    0.068269  4.11368 0.000039
## beta1   0.709597    0.075003  9.46096 0.000000
##
## LogLikelihood : 10781.82
##
## Information Criteria
## ------------------------------------
##
## Akaike       -2.4531
## Bayes        -2.4459
## Shibata      -2.4531
## Hannan-Quinn -2.4506
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1709  0.6793
## Lag[2*(p+q)+(p+q)-1][14]  111.6389  0.0000
## Lag[4*(p+q)+(p+q)-1][24]  287.9170  0.0000
## d.o.f=5
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.08793 0.76682
## Lag[2*(p+q)+(p+q)-1][5]   5.46366 0.11996
## Lag[4*(p+q)+(p+q)-1][9]   9.90705 0.05251
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]     3.256 0.500 2.000 0.07116
## ARCH Lag[5]     7.312 1.440 1.667 0.02934
## ARCH Lag[7]     9.324 2.315 1.543 0.02636
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  3.1906
## Individual Statistics:
## mu     0.13074
## ar1    0.67172
## ar2    0.31254
## ar3    0.09501
## ar4    0.08285
## ar5    0.24101
```

```
## omega  0.86643
## alpha1 0.41189
## beta1  0.57872
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.1 2.32 2.82
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## -----------------------------------
##                    t-value   prob sig
## Sign Bias           0.6377 0.5237
## Negative Sign Bias  0.6839 0.4941
## Positive Sign Bias  0.3954 0.6926
## Joint Effect        1.0705 0.7842
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## -----------------------------------
##   group statistic p-value(g-1)
## 1    20     808.1   4.073e-159
## 2    30     851.8   1.517e-160
## 3    40     893.0   4.846e-162
## 4    50     945.0   3.737e-166
##
##
## Elapsed time : 1.481713
```

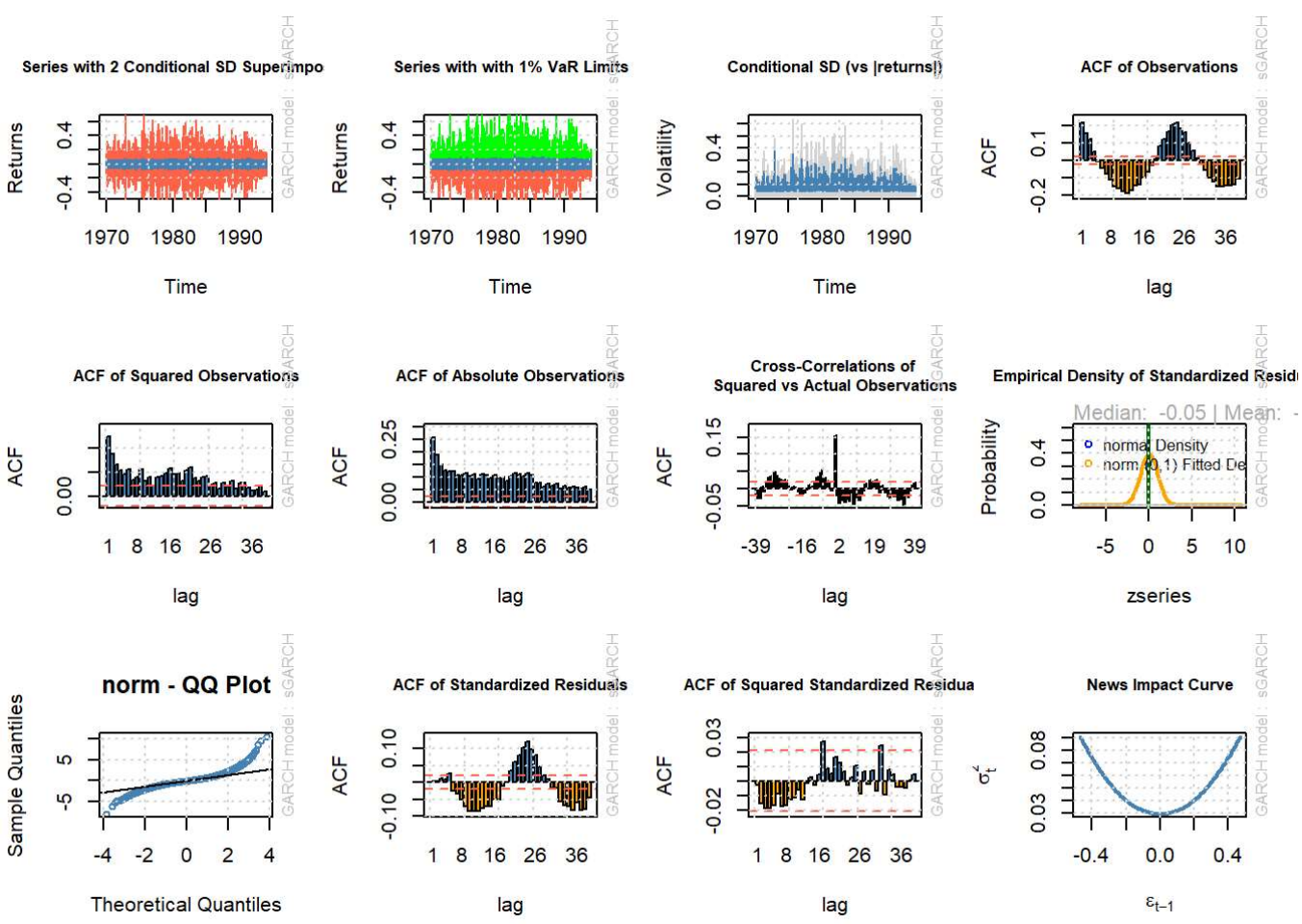
For Model with orders (5,0):

AIC : -2.4531

BIC : -2.4459

```
plot(m_norm, which = 'all')
```

```
##
## please wait...calculating quantiles...
```




```
s_norm_2 <- ugarchspec(mean.model = list(armaOrder = c(4,0)),
                       variance.model = list(model = 'sGARCH'),
                       distribution.model = 'norm')

m_norm_2 <- ugarchfit(data = return, spec = s_norm_2, solver ='hybrid')
m_norm_2
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(4,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001929    0.001044   1.8475 0.064671
## ar1     0.243241    0.013339  18.2348 0.000000
## ar2     0.118928    0.012934   9.1949 0.000000
## ar3     0.047800    0.012432   3.8449 0.000121
## ar4    -0.014286    0.011791  -1.2116 0.225679
## omega   0.000396    0.000047   8.4030 0.000000
## alpha1  0.285352    0.022451  12.7102 0.000000
## beta1   0.705850    0.020490  34.4489 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001929    0.001133   1.7025 0.088666
## ar1     0.243241    0.020462  11.8873 0.000000
## ar2     0.118928    0.016666   7.1362 0.000000
## ar3     0.047800    0.013208   3.6191 0.000296
## ar4    -0.014286    0.014973  -0.9541 0.340034
## omega   0.000396    0.000189   2.0954 0.036137
## alpha1  0.285352    0.070354   4.0559 0.000050
## beta1   0.705850    0.076995   9.1675 0.000000
##
## LogLikelihood : 10772.7
##
## Information Criteria
## ------------------------------------
##
## Akaike       -2.4513
## Bayes        -2.4448
## Shibata      -2.4513
## Hannan-Quinn -2.4491
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.4101  0.5219
## Lag[2*(p+q)+(p+q)-1][11]   53.7372  0.0000
## Lag[4*(p+q)+(p+q)-1][19]  206.8419  0.0000
## d.o.f=4
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.09356 0.75970
## Lag[2*(p+q)+(p+q)-1][5]   5.57080 0.11330
## Lag[4*(p+q)+(p+q)-1][9]   9.91264 0.05237
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]     3.394 0.500 2.000 0.06542
## ARCH Lag[5]     7.288 1.440 1.667 0.02973
## ARCH Lag[7]     9.287 2.315 1.543 0.02688
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  2.974
## Individual Statistics:
## mu     0.12538
## ar1    0.66682
## ar2    0.30242
## ar3    0.09128
## ar4    0.10144
## omega  0.87259
## alpha1 0.40493
## beta1  0.56235
```

```
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.89 2.11 2.59
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## -----------------------------------
##                    t-value   prob sig
## Sign Bias           0.9065 0.3647
## Negative Sign Bias  0.8912 0.3728
## Positive Sign Bias  0.2589 0.7957
## Joint Effect        1.5172 0.6783
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     804.9   1.931e-158
## 2    30     856.0   1.964e-161
## 3    40     905.3   1.356e-164
## 4    50     949.8   3.768e-167
##
##
## Elapsed time : 1.129262
```

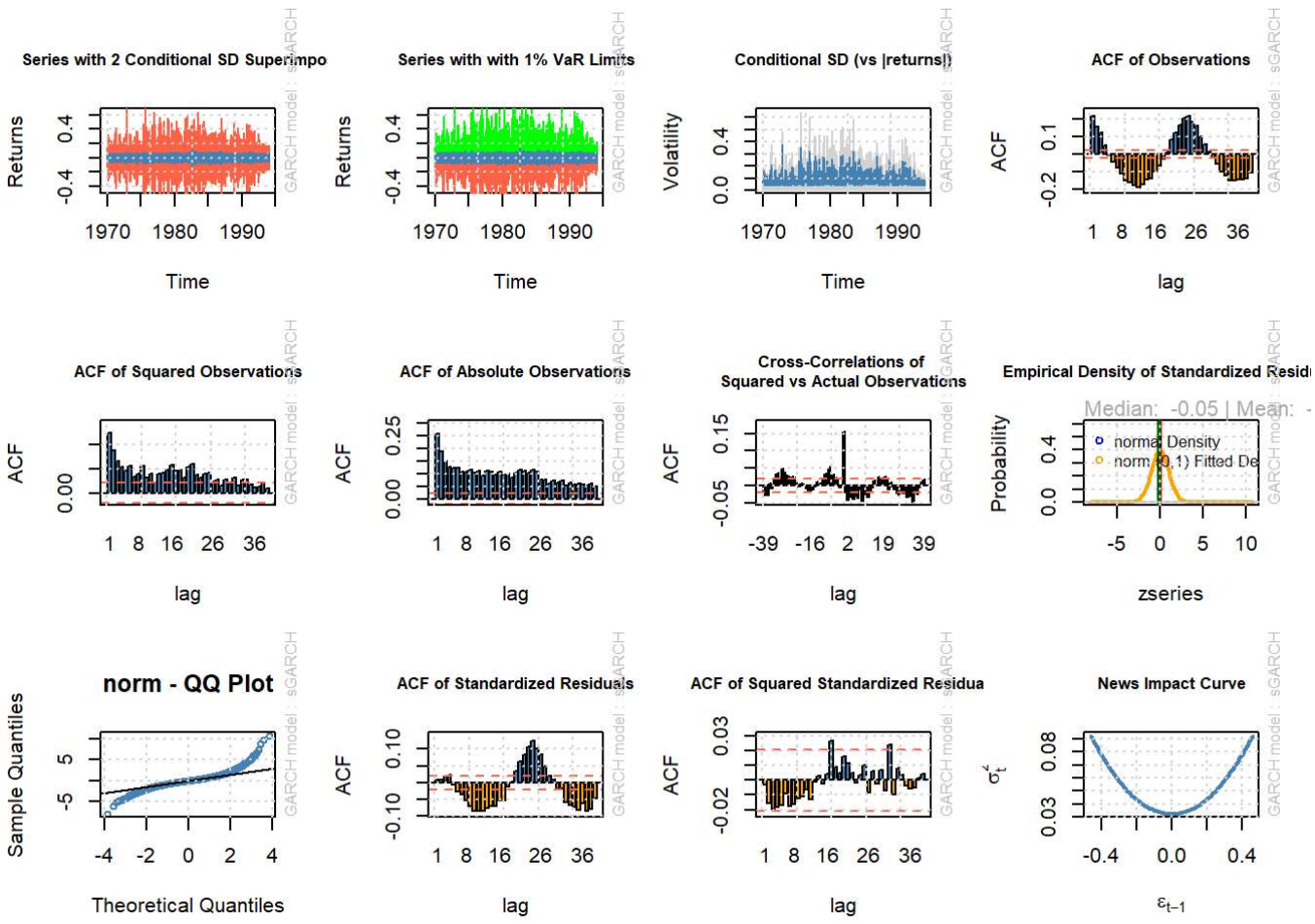
For Model with orders (4,0):

AIC : -2.4513

BIC : -2.4448

```
plot(m_norm_2, which = 'all')
```

```
##
## please wait...calculating quantiles...
```




```
s_norm_3 <- ugarchspec(mean.model = list(armaOrder = c(3,0)),
                       variance.model = list(model = 'sGARCH'),
                       distribution.model = 'norm')

m_norm_3 <- ugarchfit(data = return, spec = s_norm_3, solver ='hybrid')
m_norm_3
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(3,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001871    0.001061   1.7635 0.077815
## ar1     0.242656    0.013349  18.1783 0.000000
## ar2     0.117160    0.012830   9.1316 0.000000
## ar3     0.045113    0.012252   3.6822 0.000231
## omega   0.000402    0.000048   8.4551 0.000000
## alpha1  0.287691    0.022598  12.7306 0.000000
## beta1   0.703180    0.020615  34.1103 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.001871    0.001143   1.6374 0.101546
## ar1     0.242656    0.020578  11.7920 0.000000
## ar2     0.117160    0.016454   7.1206 0.000000
## ar3     0.045113    0.013178   3.4233 0.000619
## omega   0.000402    0.000190   2.1164 0.034311
## alpha1  0.287691    0.070149   4.1012 0.000041
## beta1   0.703180    0.076843   9.1509 0.000000
##
## LogLikelihood : 10772.44
##
## Information Criteria
## ------------------------------------
##
## Akaike       -2.4514
## Bayes        -2.4458
## Shibata      -2.4514
## Hannan-Quinn -2.4495
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.5125  0.4741
## Lag[2*(p+q)+(p+q)-1][8]    14.0154  0.0000
## Lag[4*(p+q)+(p+q)-1][14]  113.0290  0.0000
## d.o.f=3
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1103 0.73982
## Lag[2*(p+q)+(p+q)-1][5]    5.6155 0.11063
## Lag[4*(p+q)+(p+q)-1][9]    9.9053 0.05256
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]     3.402 0.500 2.000 0.06513
## ARCH Lag[5]     7.274 1.440 1.667 0.02996
## ARCH Lag[7]     9.224 2.315 1.543 0.02779
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  2.8381
## Individual Statistics:
## mu     0.12381
## ar1    0.65461
## ar2    0.29863
## ar3    0.09481
## omega  0.87998
## alpha1 0.40588
## beta1  0.56738
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.69 1.9 2.35
```

```
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.8763 0.3809
## Negative Sign Bias  0.9212 0.3570
## Positive Sign Bias  0.2935 0.7692
## Joint Effect        1.5273 0.6760
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     814.7   1.603e-160
## 2    30     858.1   7.090e-162
## 3    40     904.4   2.072e-164
## 4    50     936.0   2.652e-164
##
##
## Elapsed time : 0.9274321
```

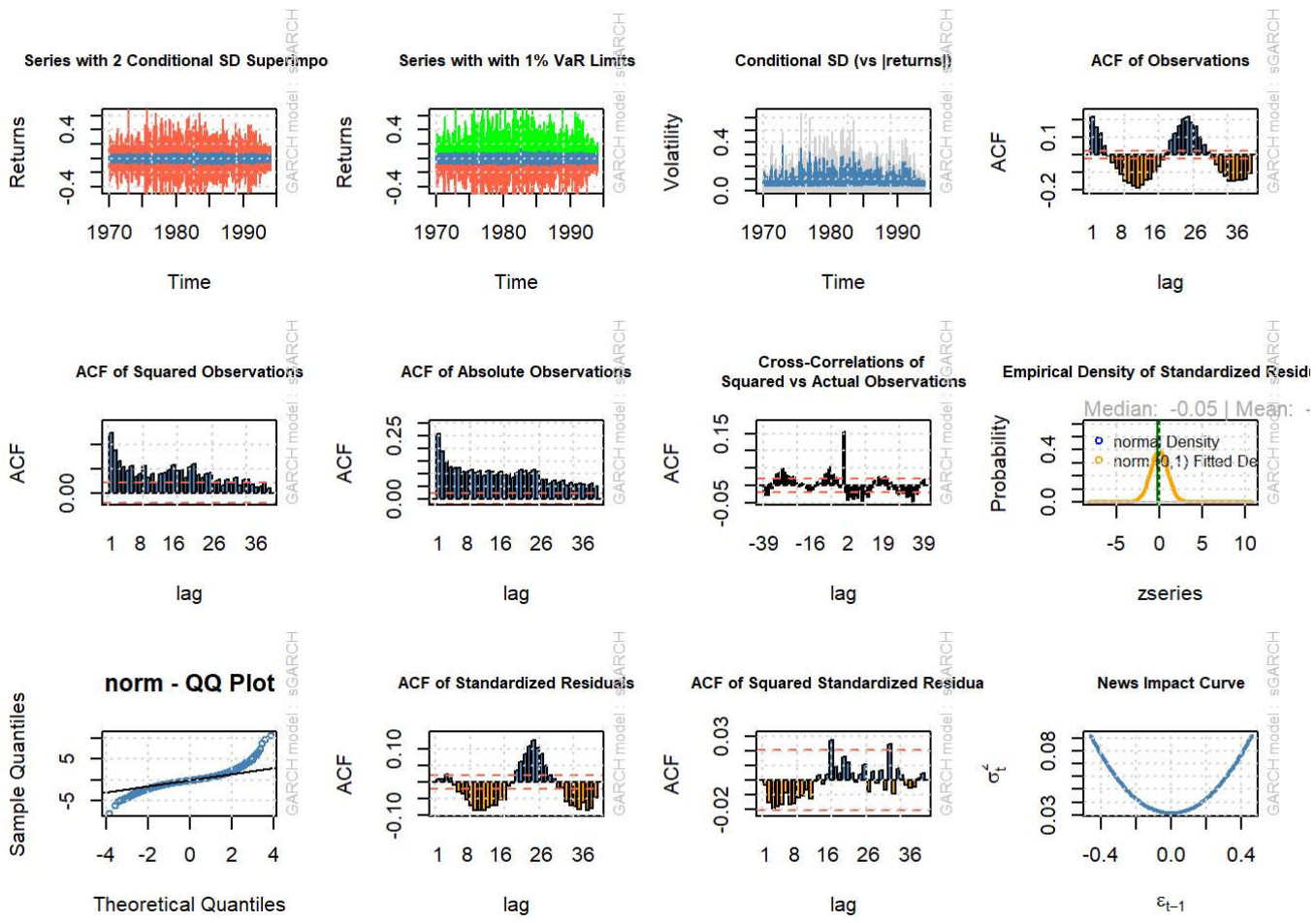
For Model with orders (4,0):

AIC : -2.4514

BIC : -2.4458

```
plot(m_norm_3, which = 'all')
```

```
##
## please wait...calculating quantiles...
```

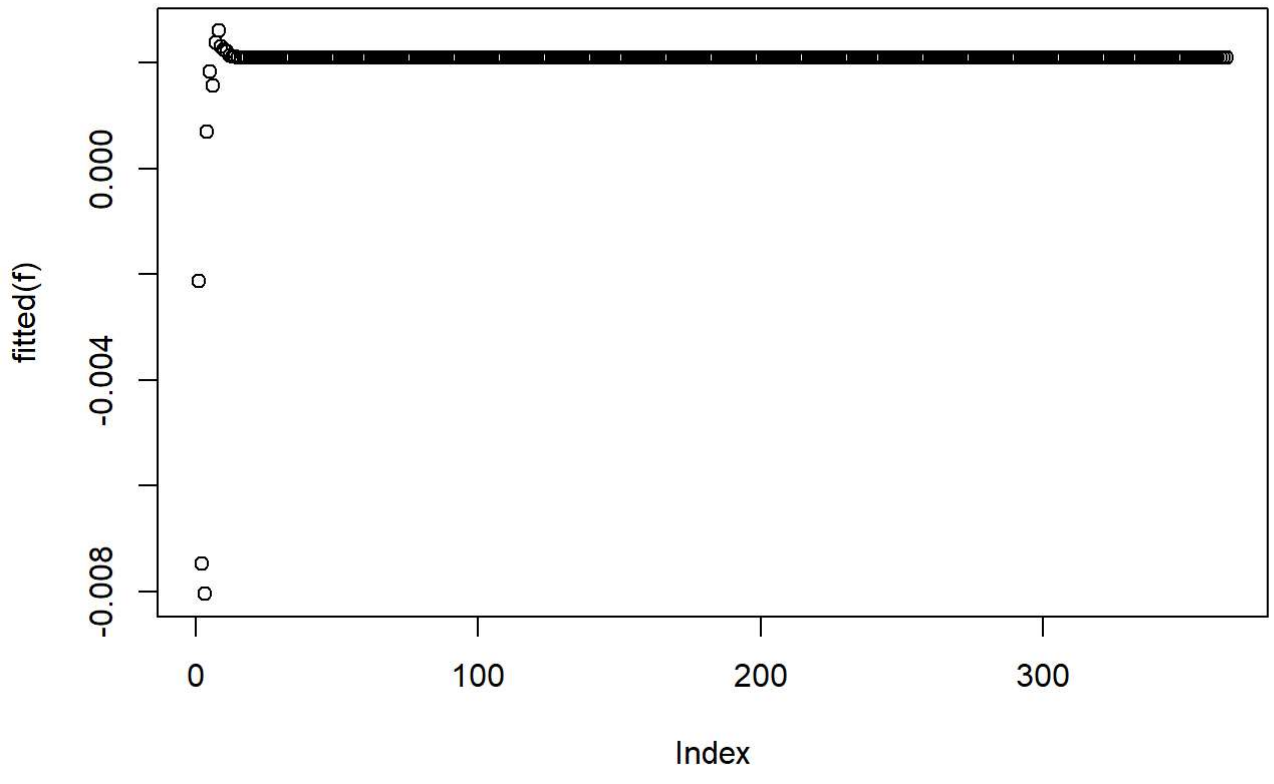


The lowest AIC and BIC values were of Model with orders (5,0). Hence, that is the best model that can be fitted.

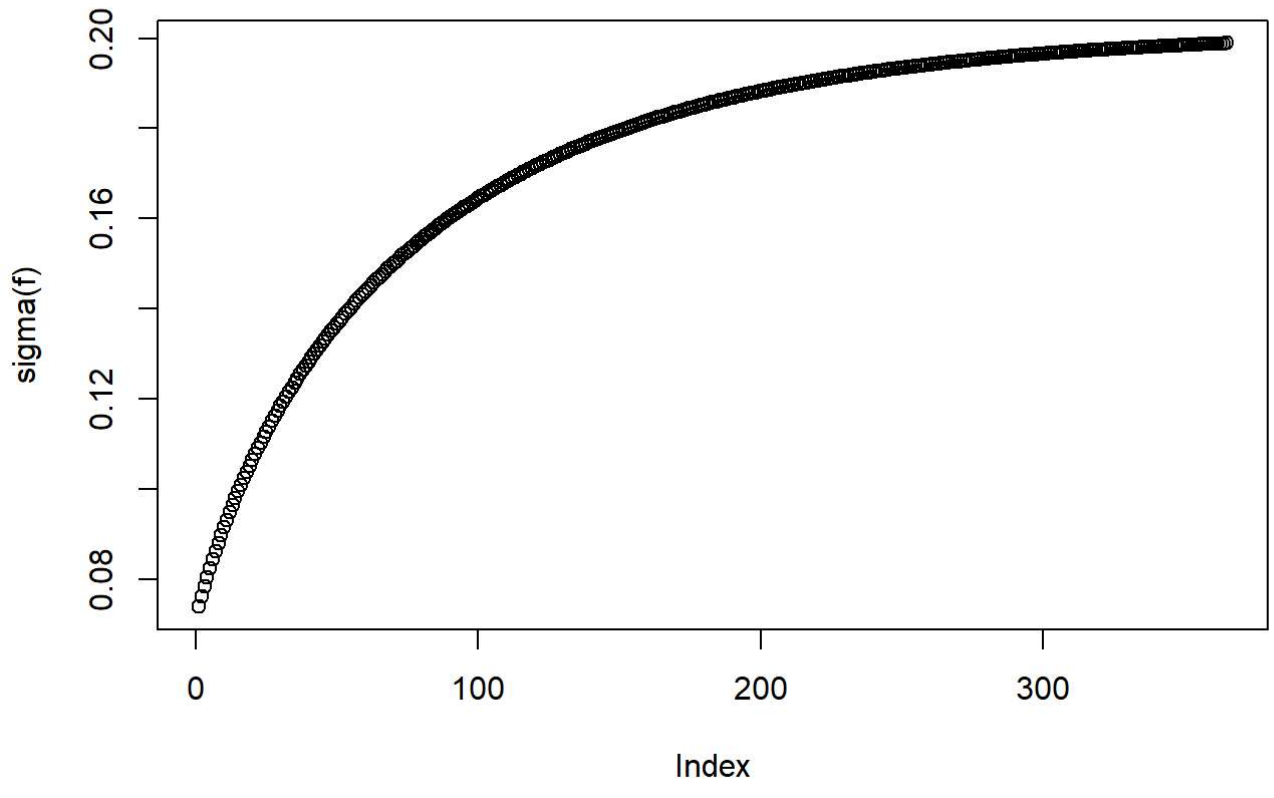
Forecasting

The best fit we get from the sGARCH model with orders (5,0). Hence, we use this model to forecast the year ahead data i.e. 365 days ahead.

```
s2final <-  ugarchspec(variance.model=list(model="sGARCH"),
                mean.model=list(armaOrder=c(5,0)),distribution.model="norm")

m2final <- ugarchfit(data = return, spec = s2final)

f <- ugarchforecast(fitORspec = m2final, n.ahead = 365)
plot(fitted(f))
```



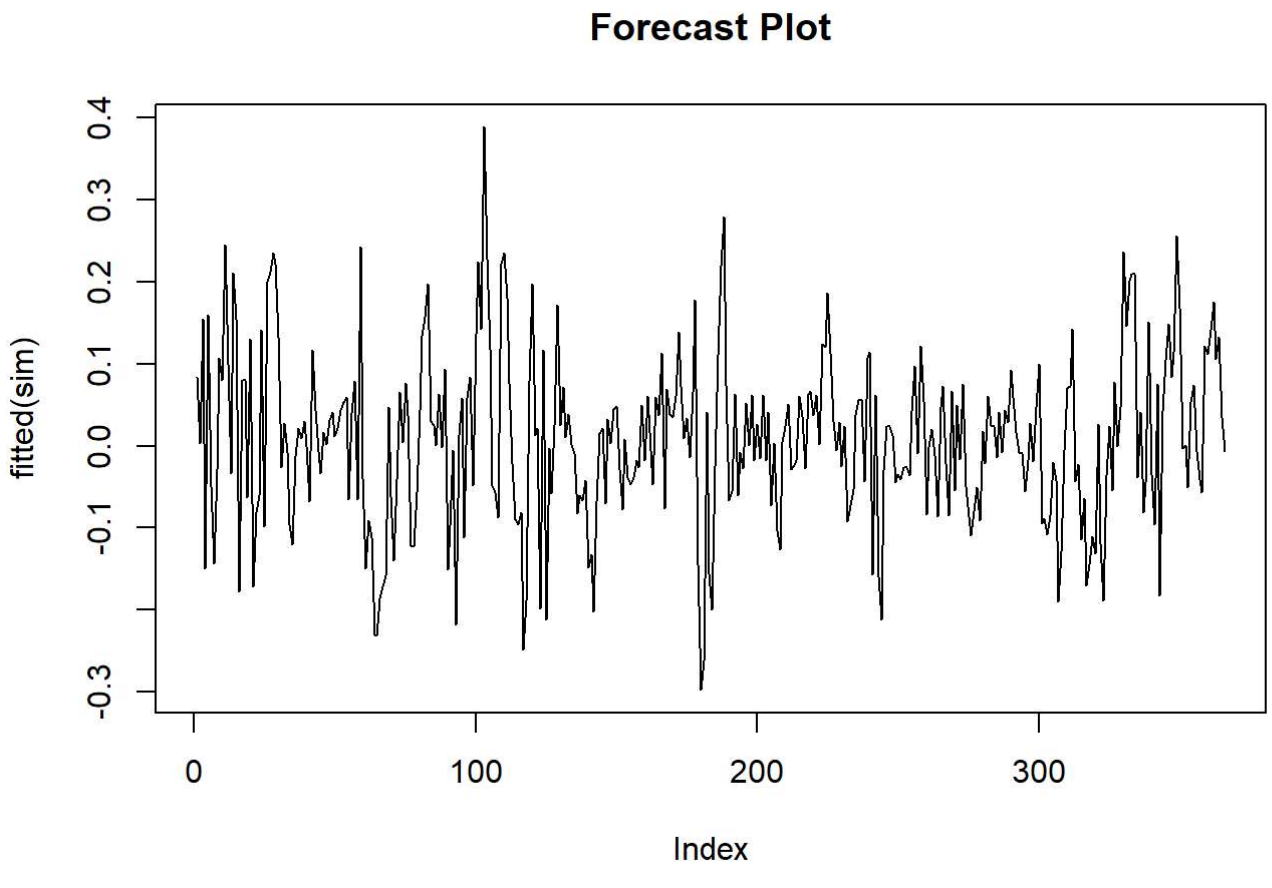

```
plot(sigma(f))
```




```
library(zoo)

sfinal <- s_norm

setfixed(sfinal) <- as.list(coef(m_norm))
coef(m_norm)
```

```
##            mu           ar1           ar2           ar3           ar4
##  0.0021112880  0.2427569649  0.1217127031  0.0545640276 -0.0044689340
##           ar5         omega        alpha1         beta1
## -0.0495731175  0.0003886226  0.2808376922  0.7095974168
```

```
sim <- ugarchpath(spec = sfinal,
                  m.sim = 1,
                  n.sim = 1*366,
                  rseed = 16)
plot.zoo(fitted(sim), main ="Forecast Plot")
```



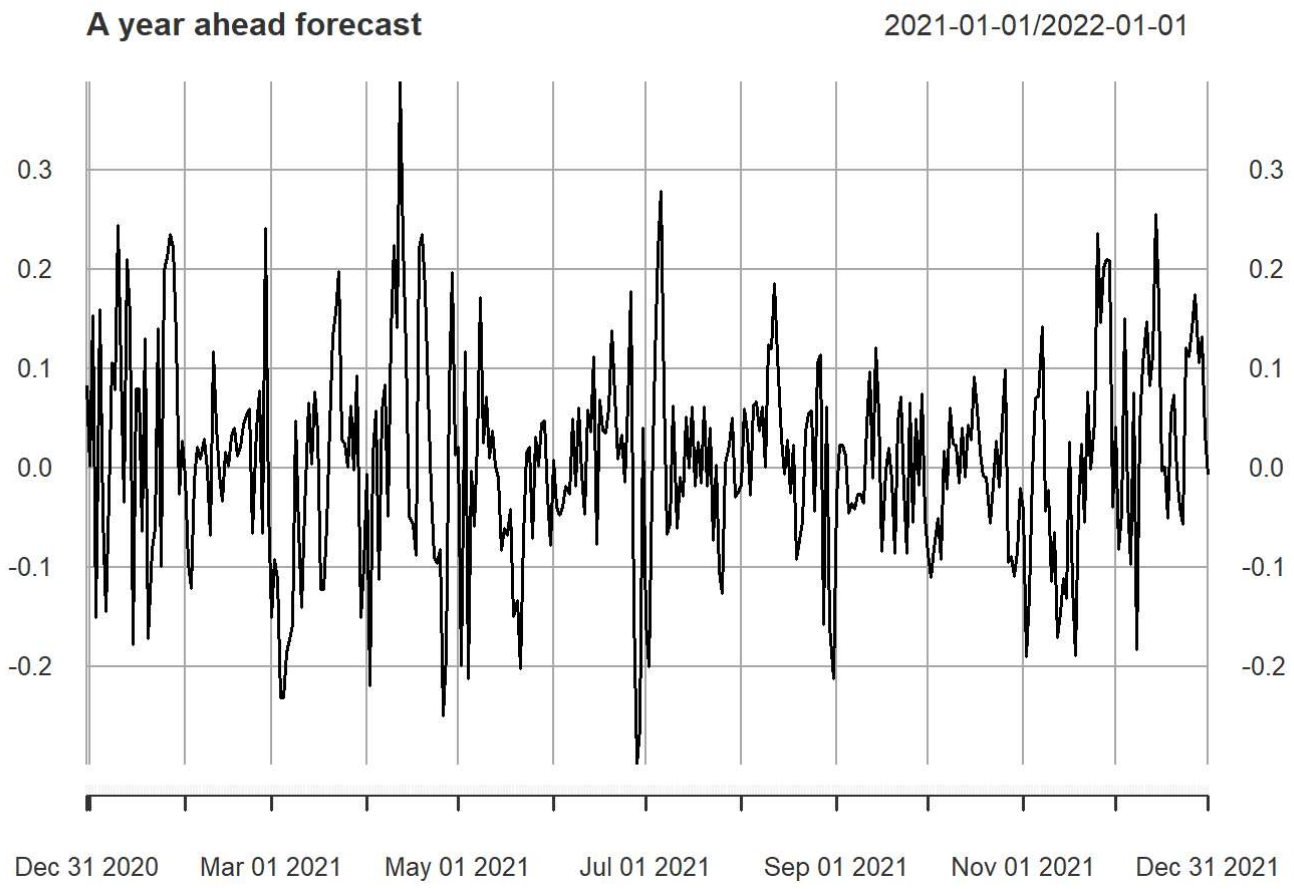
Thus, above is the plot of forecasted values. I checked for the last date of the data.

```
tail(data$Date.Time, n = 1)
```

```
## [1] "2020-12-31"
```

As we can see above, the last date of the data is December 31. Now, I plot the final plot of the forecast using the dates for better understanding.

```
library(ggplot2)
library(xts)

p_r <- xts(sim@path$seriesSim, order.by = as.Date("2021-01-01") + 0:(366-1))
plot(p_r, main = "A year ahead forecast", lwd = 1.5)
```



In the above plot, we can see that the maximum precipitation can be expected in the month of April around the second week due to the highest peak in the plot. During the month of June, the relative humidity may fall to the lowest and July but the year approaches to the month of September, the variation in the relative humidity itself lowers. Thus, we forecasted the relative humidity for the next year using the best model i.e. sGARCH(5,0).