ch22m518-project

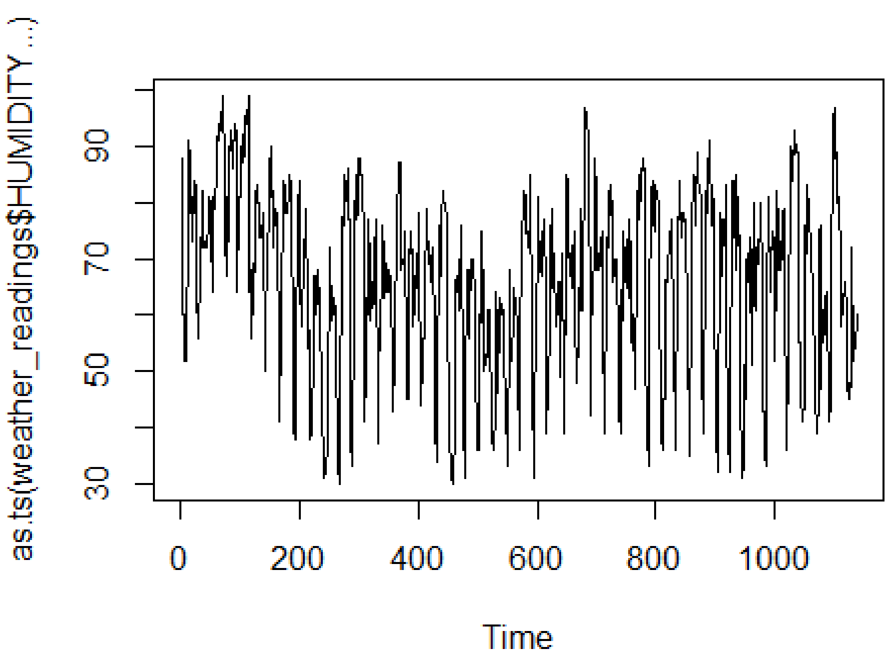
Husain Dehnuwala - ch22m518

```
library(lubridate)
library(padr)
library(imputeTS)
library(AnalyzeTS)
library(forecast)
library(Metrics)
```

1. Removing first four rows.

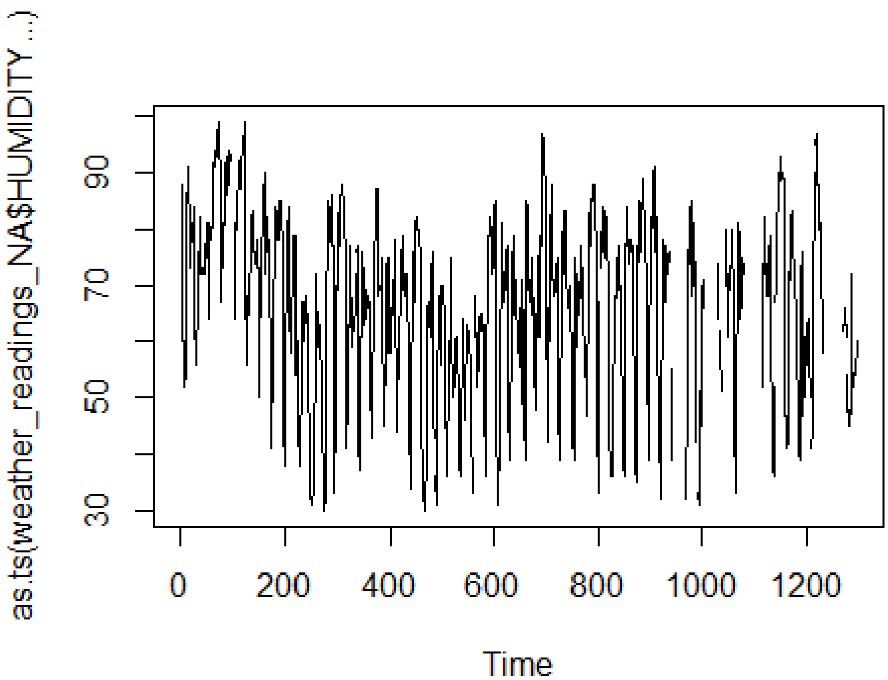
The first four columns of data are irrelevant since they do not represent any measurable quantity, we will remove them. This is because they simply convey information of the sensors, but not the sensor readings.

```
path = 'D:/Education/IITM/1 Trimester/Applied Time Series
Analysis/Project/SHAR_MAY15_JULY7.csv'
#Load entire data in a data frame
data = data.frame(read.csv(path))
#Remove first 4 column from the imported data and keep only remaining data
for analysis
weather_readings = data[,5:16]
plot(as.ts(weather_readings$HUMIDITY...))
```



2. Identifying the missing data and filling it with 'NA'

```
weather_readings$Concat_Date_Time_IST=
mdy_hm(paste(weather_readings$DATE.IST., weather_readings$TIME.IST.))
#Filling the missing data with NA
weather_readings_NA = pad(weather_readings, interval = "hour" ,
by="Concat_Date_Time_IST")
# We plot the data here to see the effect of added NA values
plot(as.ts(weather_readings_NA$HUMIDITY...))
```



We can observe empty blocks in the graph representing the empty values

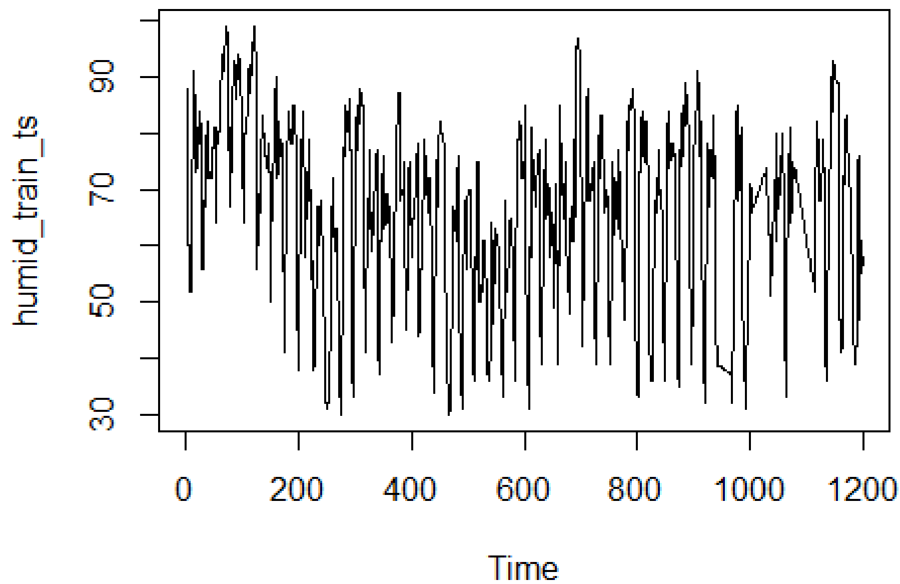
3. Filling the missing data (NA) using interpolation.

Interpolation is a process of determining the unknown values that lie in between the known data points.

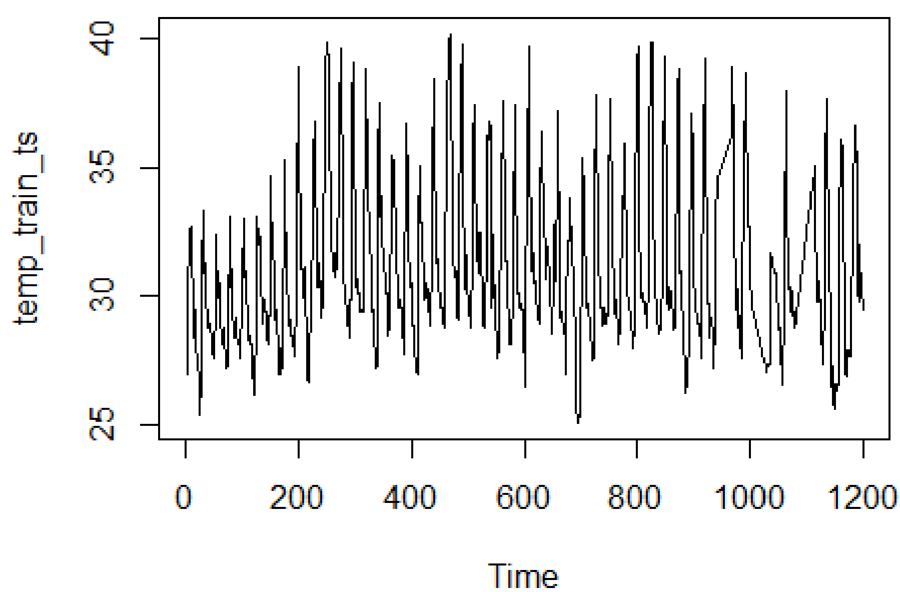
```
interpolated_data = na_interpolation(weather_readings_NA)
# Segregation of testing and training data leaving 96 values for test.
training_data = interpolated_data[1:1200,1:13]
test_data = interpolated_data[1201:1296,1:13]
```

4. Creating Time Series objects for RH and temperature variables, respectively

```
humid_train_ts = as.ts(training_data$HUMIDITY...)
temp_train_ts = as.ts(training_data$AIR_TEMP..C.)
humid_test_ts = as.ts(test_data$HUMIDITY...)
temp_test_ts = as.ts(test_data$AIR_TEMP..C.)
plot(humid_train_ts)
```



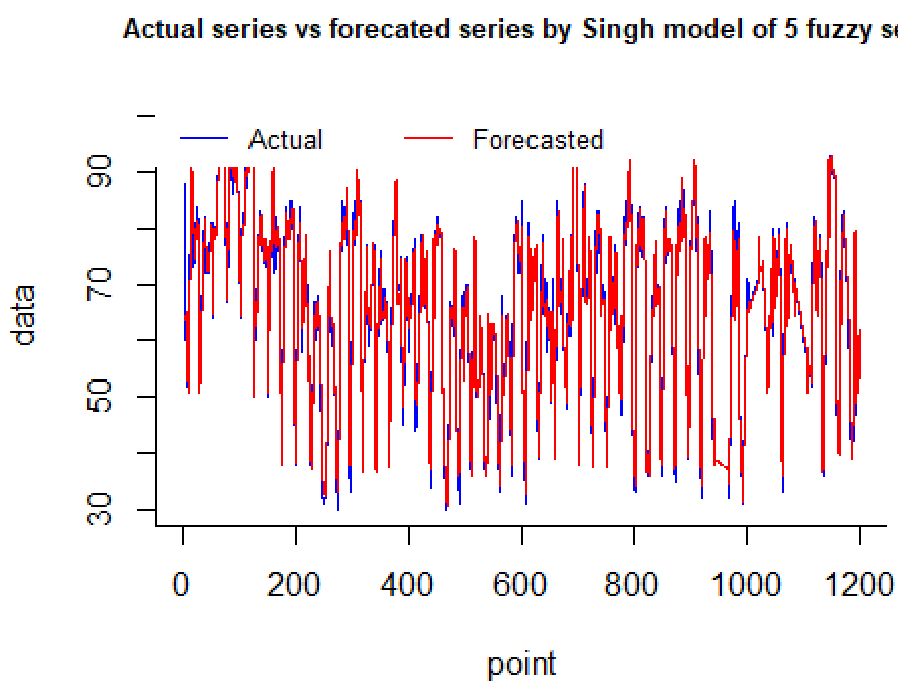

```
plot(temp_train_ts)
```



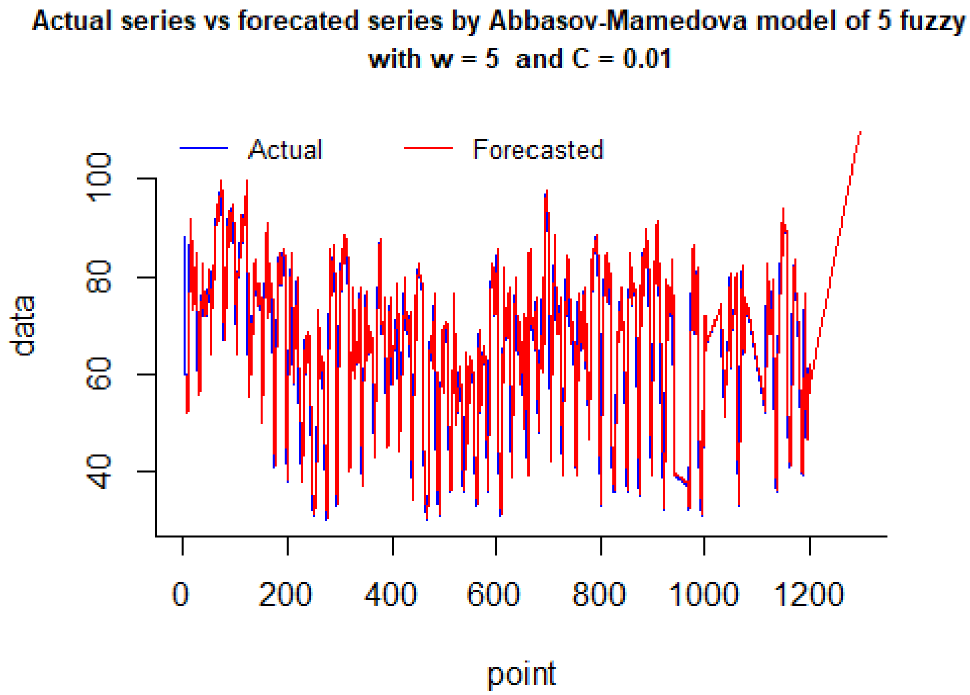
Notice that the ecmpty blocks observed in data after step 2 disappear since we have added values using interpolation.

5. Build Fuzzy Time Series model M1.

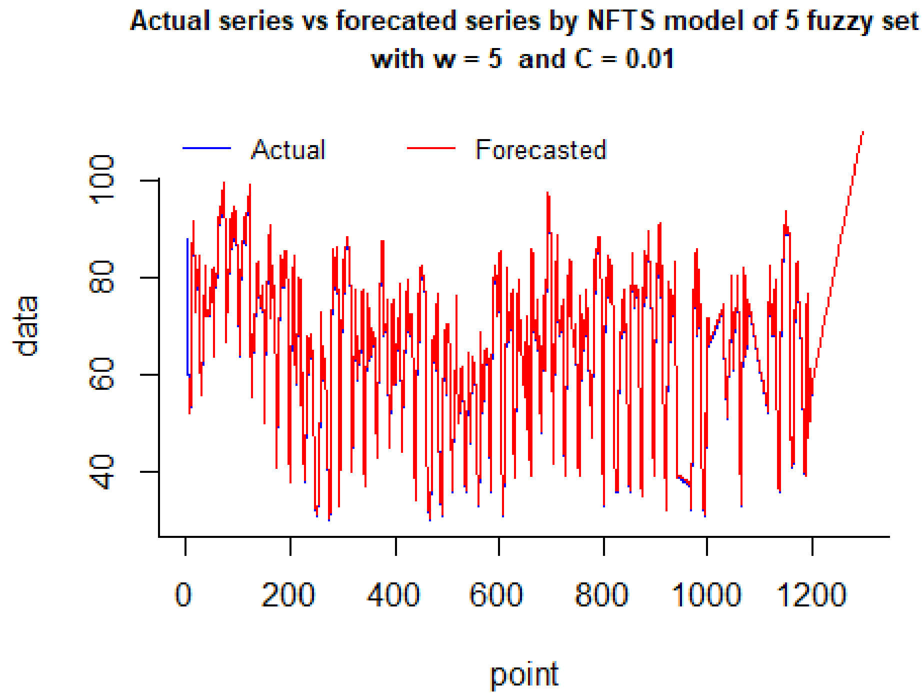
```
# Singh model
m1_singh=fuzzy.ts1 ( humid_train_ts, n=5, type="Singh", plot=TRUE)
```




```
#m1_singh_for=m1_singh$forecast
#Metrics::rmse(humid_test_ts, as.vector(m1_singh_for))

#Abbasov Mamedova model
m1_abma=fuzzy.ts2
( humid_train_ts ,n=5,w=5,C=0.01,forecast=96,plot=TRUE,type="Abbasov-
Mamedova", trace=FALSE)
```




```
m1_abma_for=m1_abma$forecast
Metrics::rmse(humid_test_ts, as.vector(m1_abma_for))

## [1] 31.18754

#NFTS model
m1_nfts=fuzzy.ts2
( humid_train_ts,n=5,w=5,C=0.01,forecast=96,plot=TRUE,type="NFTS",
trace=FALSE)
```




```
m1_nfts_for=m1_nfts$forecast
Metrics::rmse(humid_test_ts, as.vector(m1_nfts_for))

## [1] 31.52966
```

Accuracy for Singh

```
        ME   MAE    MPE  MAPE    MSE  RMSE         U
Singh 0.048 2.694 -0.168 4.384 11.637 3.411 0.5011703
```

Accuracy for Abbasov Mamedova model

```
                    ME   MAE    MPE  MAPE    MSE  RMSE         U
Abbasov.Mamedova -0.503 4.452 -1.435 7.401 45.446 6.741 0.9920568
```

accuracy for NFTS

```
        ME   MAE    MPE  MAPE  MSE  RMSE          U
NFTS -0.499 0.525 -0.747 0.799 0.37 0.608 0.08955515
```

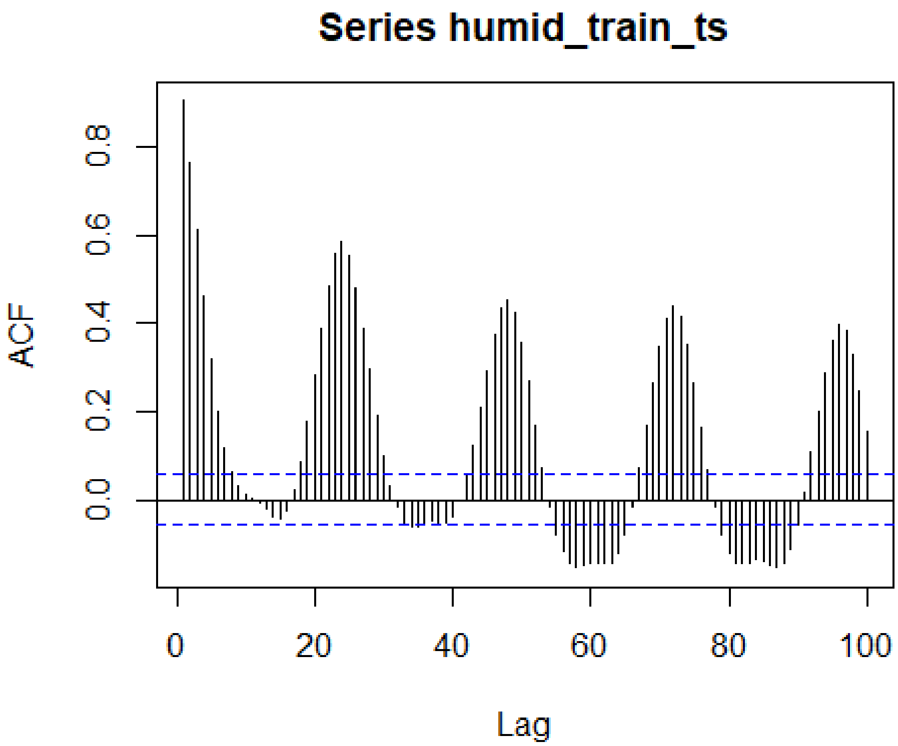
We find the best RMSE for NFTS fuzzy model from AnalyzeTS package- 0.608 on the training data.

However, for testing data, we get approximately same RMSE for all M1 models, i.e. 31. The forecast metrics are added above for reference, we conclude that NFTS is the best fuzzy model from AnalyzeTS package.

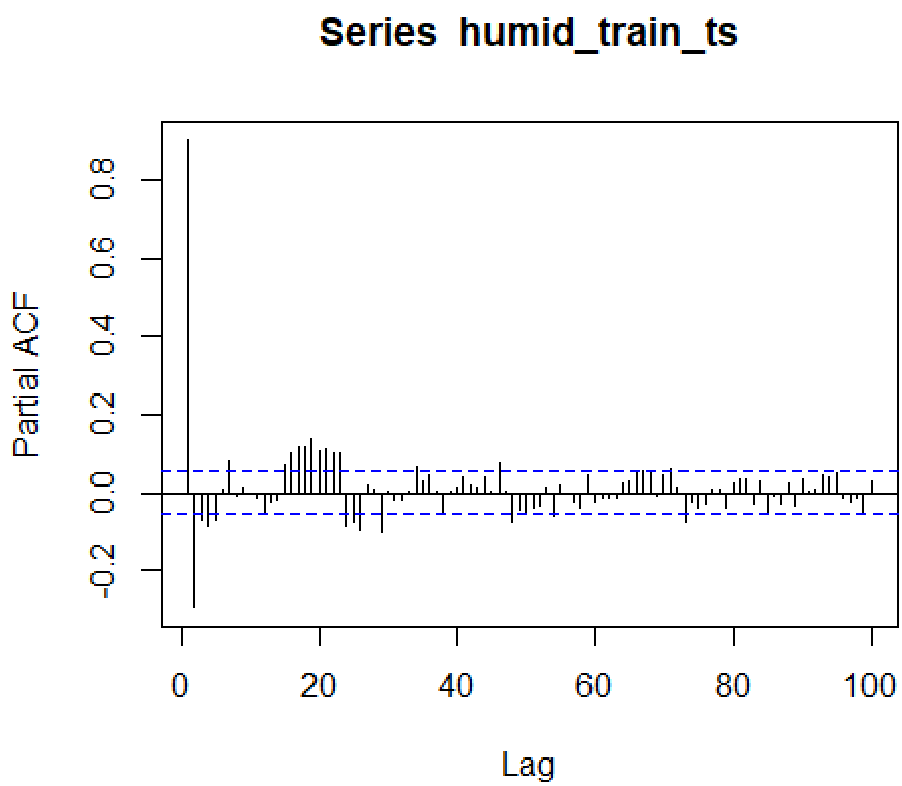
6. SARIMA model with temperature as the exogenous input to RH

```
temp_humid_ts = as.ts(interpolated_data[c(5, 9)])
```

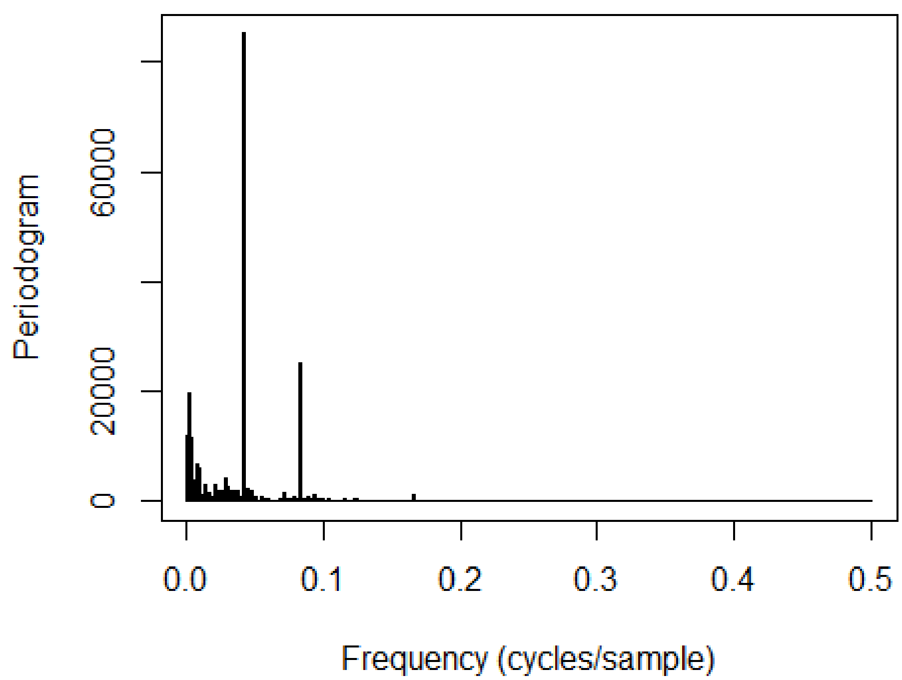
```
par(mfrow=c(1,1), mai = c(0.8,1,0.6,0.2))
acf(humid_train_ts,lag.max=100)
```




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

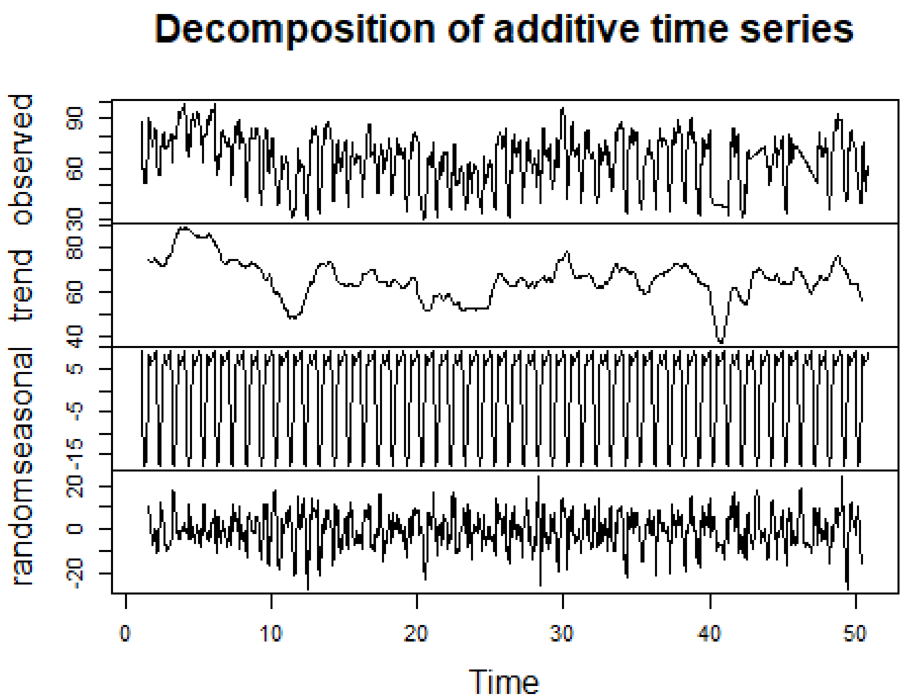



```
periodogram(humid_train_ts,ylab="Periodogram",xlab="Frequency
(cycles/sample)")
```



Depicted frequencey is 24 as per the periodogram.

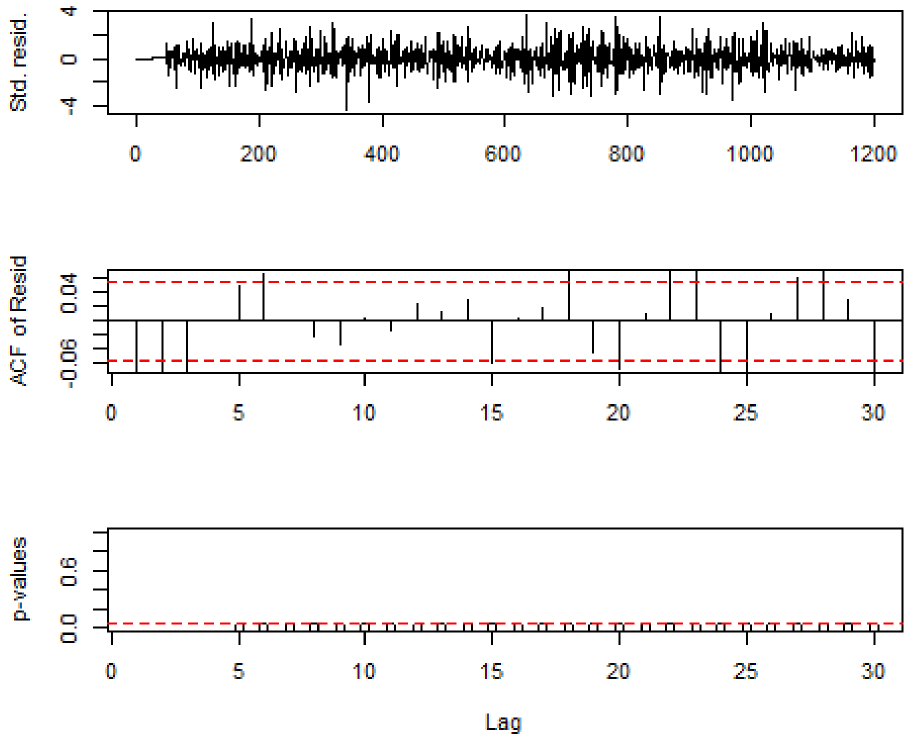
```
decomposed_data <- decompose(ts(humid_train_ts, frequency=24))
plot(decomposed_data)
```



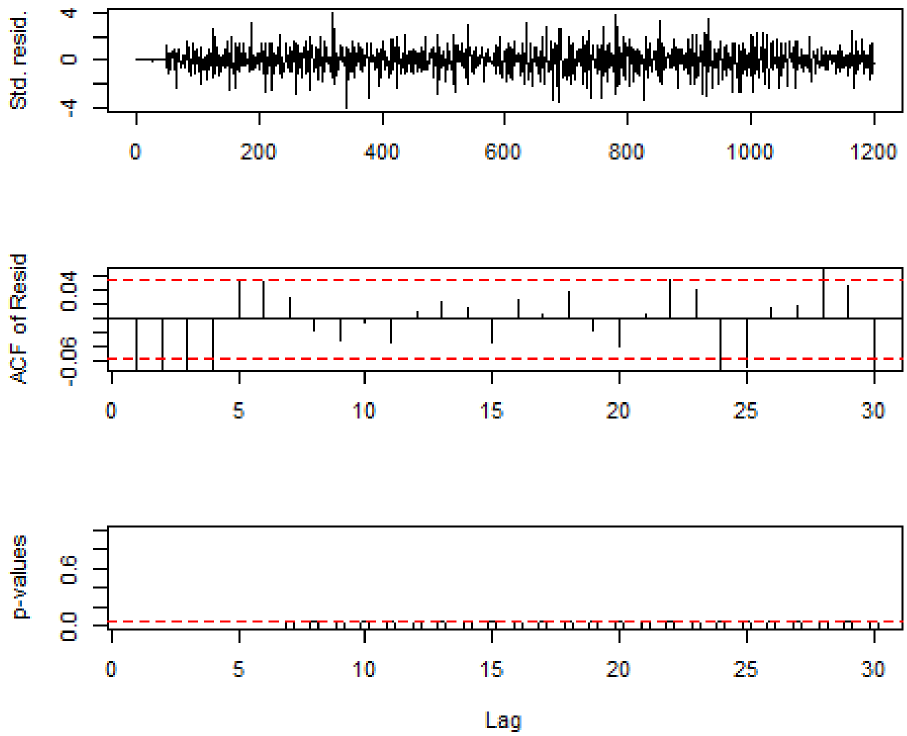
Trying out several ARIMA models to test which one fits best

```
AICVal <- function(modarima,Lmax=30) {
  summary(modarima)
  return(modarima$aic)
}
mytsdiag <- function(modarima,Lmax=30) {
  summary(modarima)
  err_mod <- modarima$residuals
  N = length(err_mod)
  par(mfrow=c(3,1), mai = c(0.6,0.7,0.2,0.2))
  plot(scale(err_mod),type='l',ylab="Std. resid.",xlab="")
  acf_err <- acf(err_mod,lag.max=Lmax,main="",plot=F)
  lowsig = -1.96/sqrt(N); upsig = 1.96/sqrt(N)
  plot(acf_err$lag*tsp(err_mod)[3],acf_err$acf,type='h',main="",ylab="ACF of 
Resid",xlab="",ylim=c(1.2*lowsig,1.2*upsig))
  abline(h=upsig,col="red",lty="dashed")
  abline(h=lowsig,col="red",lty="dashed")
  abline(h=0,col="black")
  blpval <- NULL
  npar <- sum(modarima$arma[1:4])
  Lval <- (npar+1):Lmax
  for (L in Lval) {
```

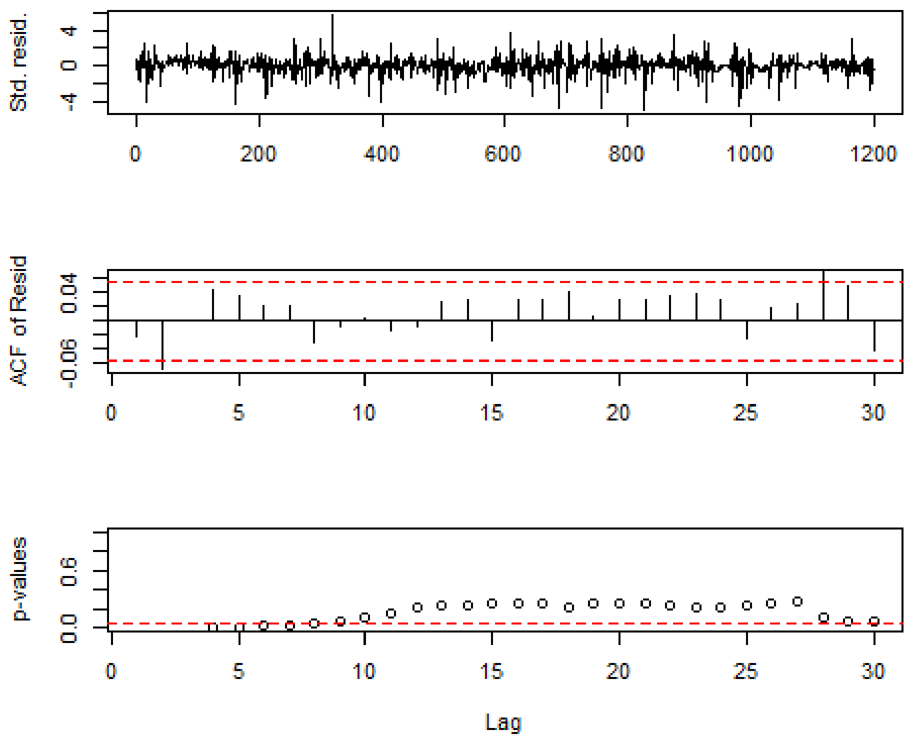
```
    blpval <- c(blpval,Box.test(modarima$residuals,lag=L,fitdf=npar)$p.value)
  }
  # Plot BLP statistic
  plot(1:Lmax,c(rep(NA,npar),blpval),ylab="p-values",xlab="Lag",ylim=c(0,1))
  abline(h=0.05,col='red',lty="dashed")
}
arima_model_1 <-
forecast::Arima(humid_train_ts,order=c(2,2,0),seasonal=list(order=c(2,2,0),
period=24), xreg = temp_train_ts)
AIC1=AICVal(arima_model_1)
arima_model_2 <-
forecast::Arima(humid_train_ts,order=c(3,2,0),seasonal=list(order=c(3,2,0),
period=24),xreg = temp_train_ts)
AIC2=AICVal(arima_model_2)
arima_model_3 <-
forecast::Arima(humid_train_ts,order=c(1,0,0),seasonal=list(order=c(1,0,1),
period=24),xreg = temp_train_ts)
AIC3=AICVal(arima_model_3)
arima_model_4 <-
forecast::Arima(humid_train_ts,order=c(1,0,1),seasonal=list(order=c(1,0,0),
period=24),xreg = temp_train_ts)
AIC4=AICVal(arima_model_4)

print('AIC value and diagnostics for different order SARIMA model')

## [1] "AIC value and diagnostics for different order SARIMA model"

print(AIC1)

## [1] 7720.981

mytsdiag(arima_model_1)
```



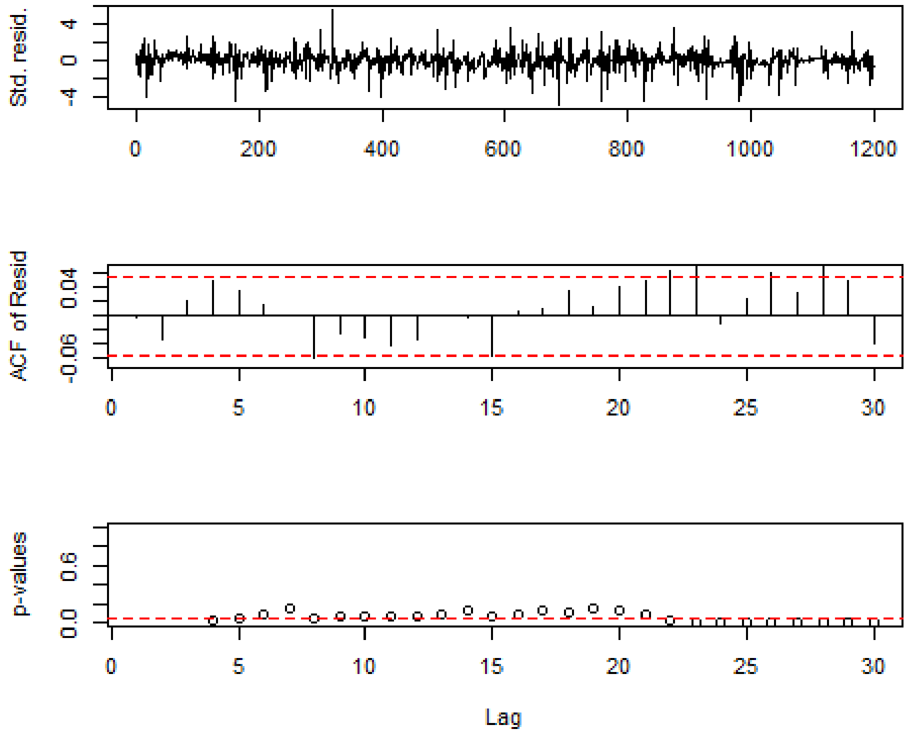

```
print(AIC2)

## [1] 7457.249

mytsdiag(arima_model_2)
```




```
print(AIC3)

## [1] 6495.601

mytsdiag(arima_model_3)
```




```
print(AIC4)

## [1] 6535.874

mytsdiag(arima_model_4)
```



We will find confidence intervals and look for significant coefficients for the two models that are successful above, based on p-value.

```
print('SARIMA(1,0,0)(1,0,1)')

## [1] "SARIMA(1,0,0)(1,0,1)"

arima_model_3$coef

##         ar1        sar1        sma1   intercept        xreg
##   0.8686866   0.9780800  -0.9271273 209.2898271  -4.6162893

confint(arima_model_3)

##                  2.5 %      97.5 %
## ar1          0.8404230   0.8969501
## sar1         0.9498487   1.0063113
## sma1        -0.9824903  -0.8717643
## intercept 202.1899700 216.3896843
## xreg        -4.8170084  -4.4155702

print('SARIMA(1,0,1)(1,0,0)')

## [1] "SARIMA(1,0,1)(1,0,0)"

arima_model_4$coef

##          ar1          ma1         sar1    intercept         xreg
##   0.86704897   0.03365516   0.12773872 206.57206340   -4.52939925
```

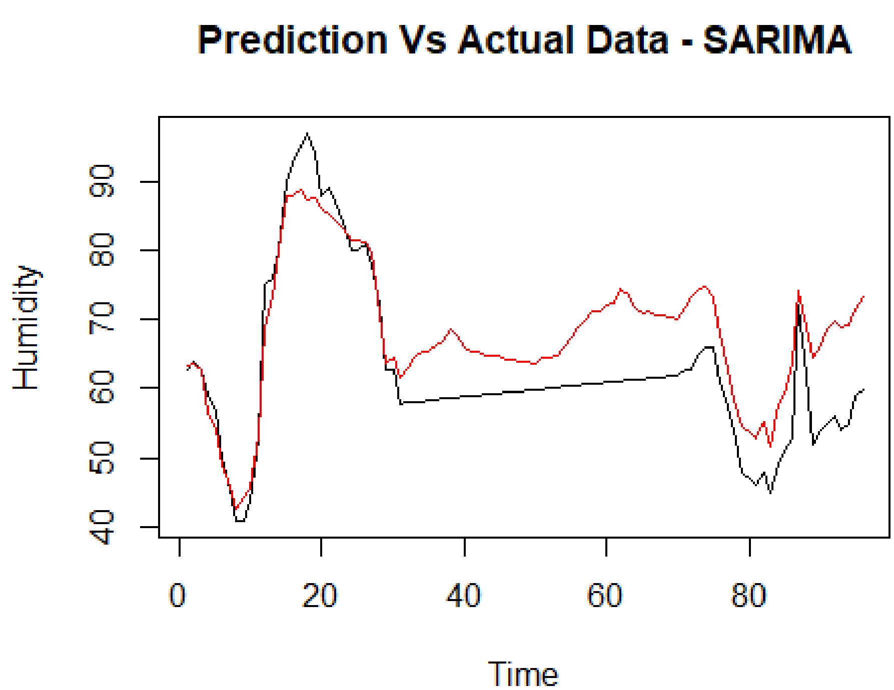
```
confint(arima_model_4)

##                  2.5 %      97.5 %
## ar1          0.83401749   0.9000804
## ma1         -0.03456597   0.1018763
## sar1         0.06945620   0.1860213
## intercept 200.69104485 212.4530820
## xreg        -4.70902057  -4.3497779
```

As per the p-value, confidence interval, and significance of coefficients, SARIMA (1,0,0)(1,0,1) is a better model. Hence we finalise M2 as below.

```
m2=forecast::Arima(humid_train_ts,order=c(1,0,0),seasonal=list(order=c(1,0,1)
, period=24),xreg = temp_train_ts)
```

7. Compare m1 and m2 to conclude which model is better - SARIMA or analyzeTS.*

```
humid_forecast_m2 <- predict(m2,newxreg = temp_test_ts, n.ahead = 96)
Metrics::rmse(humid_test_ts[1:95], humid_forecast_m2$pred[1:95])

## [1] 7.353004

plot(humid_test_ts,ylab="Humidity",main="Prediction Vs Actual Data - SARIMA")
lines(ts(humid_forecast_m2$pred),col='red')
```



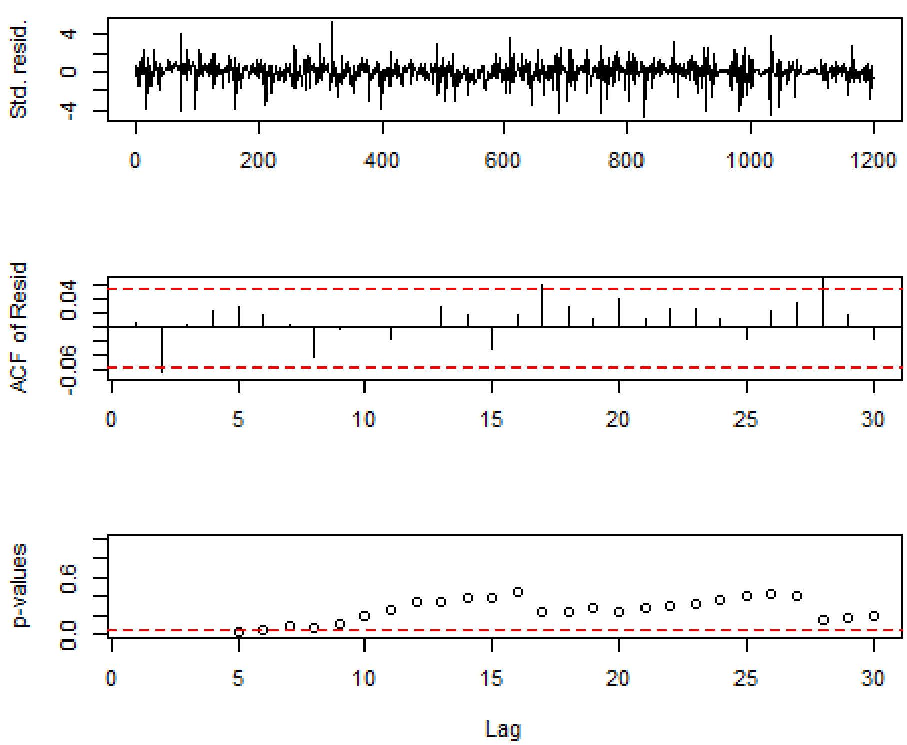
We receive an RMSE value of 7.353004 using SARIMA model. ##### To summarize - Best RMSE value using AnalyzeTS (NFTS) on training data is `0.608`, but the same model gives `RMSE value > 31` on the test dataset. In comparison, `RMSE 7.353004` is much more tempting for SARIMA (1,0,0)(1,0,1).

8. Replace imputed values from step 3 by the values that we predict from SARIMA model.

```
humid_with_NA=as.ts(weather_readings_NA$HUMIDITY)
humid_all_pred<- predict(m2,newxreg = interpolated_data$AIR_TEMP..C. ,
n.ahead = 1296)
my_range <- 1:1296
for(i in my_range) {
  if(is.na( humid_with_NA[i] ))
  {
    humid_with_NA[i]<-humid_all_pred$pred[i]
  }
}
sarima_predicted_humid_values <- humid_with_NA
```

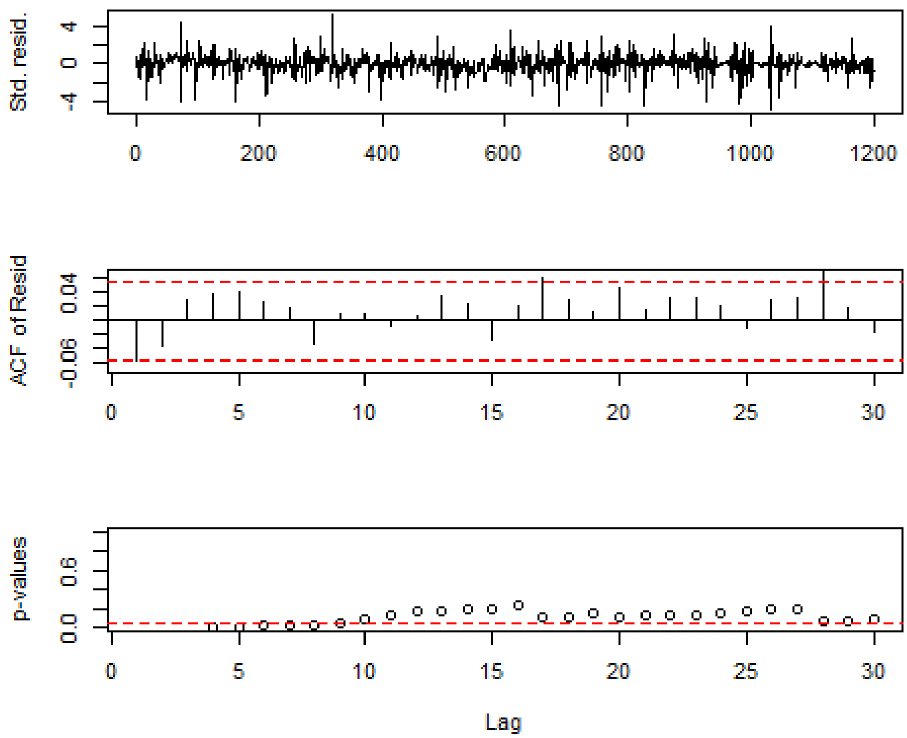
Now we will rebuild the SARIMA model to see if we get better results.

```
humid_train_data_retrain = sarima_predicted_humid_values[1:1200]
humid_test_data_retrain = sarima_predicted_humid_values[1201:1296]

arima_rebuild_model_1 <-
forecast::Arima(humid_train_data_retrain,order=c(1,0,1),seasonal=list(order=c
(1,0,1), period=24),xreg = temp_train_ts)
AIC1=AICVal(arima_rebuild_model_1)
mytsdiag(arima_rebuild_model_1)
```




```
arima_rebuild_model_2 <-
forecast::Arima(humid_train_data_retrain,order=c(1,0,0),seasonal=list(order=c
(1,0,1), period=24),xreg = temp_train_ts)
```

```
AIC2=AICVal(arima_rebuild_model_2)
mytsdiag(arima_rebuild_model_2)
```




```
print('SARIMA(1,0,0)(1,0,1)')

## [1] "SARIMA(1,0,0)(1,0,1)"

arima_rebuild_model_2$coef

##         ar1          sar1         sma1   intercept        xreg
##   0.8328743    0.9811286   -0.9333077 210.3000787   -4.6390266

confint(arima_rebuild_model_2)

##                  2.5 %      97.5 %
## ar1          0.8011530   0.8645957
## sar1         0.9536302   1.0086270
## sma1        -0.9900568  -0.8765586
## intercept  203.0780659 217.5220915
## xreg        -4.8522593  -4.4257938

print('SARIMA(1,0,1)(1,0,0)')

## [1] "SARIMA(1,0,1)(1,0,0)"

arima_rebuild_model_1$coef

##         ar1          ma1         sar1         sma1   intercept         xreg
##   0.8629911   -0.1020440    0.9850149   -0.9380148 210.9935990   -4.6619828
```

```
confint(arima_rebuild_model_1)

##                   2.5 %       97.5 %
## ar1           0.8274833   0.89849902
## ma1          -0.1777579  -0.02633005
## sar1          0.9632614   1.00676845
## sma1         -0.9882491  -0.88778054
## intercept 203.6194091 218.36778889
## xreg         -4.8732852  -4.45068043
```

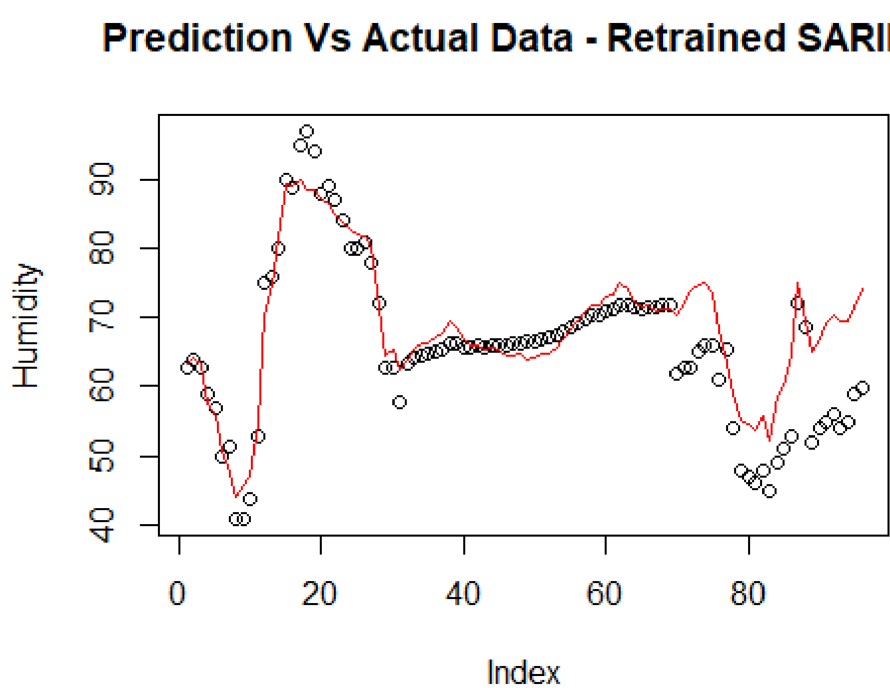
Comparing retrained SARIMA (1,0,0) (1,0,1) with the model selected at the end of step 7, i.e. SARIMA (1,0,0)(1,0,1) , we find that p-values fit better after retrain. We will also compare the RMSE values -

```
humid_retrain_forecast <- predict(arima_rebuild_model_2,newxreg =
temp_test_ts, n.ahead = 96)
Metrics::rmse(humid_test_ts[1:95], humid_retrain_forecast$pred[1:95])

## [1] 7.795987
```

RMSE after retrain is `7.795987` and the RMSE after step 7 is `7.353004`.

We notice that even though we have better p-value fit, the RMSE does not significantly get better.

```
plot(humid_test_data_retrain,ylab="Humidity",main="Prediction Vs Actual Data
- Retrained SARIMA")
lines(ts(humid_retrain_forecast$pred),col='red')
```



The resulting retrained model and the forecasts appear any significantly different from its predecessor. This can be inferred from the retrained graph showing prediction vs actual data plotted. Also, the SARIMA (1,0,0)(1,0,1) model from retrained data gives significantly better p-values. With this, we conclude that retraining a model is extremely useful device especially when there are many missing values involved.