ME16B166_ATSA_Project

Sureddy Abhishek

27 November 2019

1. Reading the data

```
Data = read.csv('SHAR_MAY15_JULY7.csv')
# finding the number of columns
n_cols = ncol(Data)
# Removing the first 4 columns, the modified data is :
mod_Data = Data[,5:n_cols]
# Resulting Data is as follows :
head(mod_Data,3)
```

```
##   TIME.GMT.  DATE.GMT. TIME.IST.  DATE.IST. AIR_TEMP WIND_SPEED.m.s.
## 1         0 05/15/2009      5:30 05/15/2009    27.49            0.05
## 2         1 05/15/2009      6:30 05/15/2009    26.96            1.03
## 3         2 05/15/2009      7:30 05/15/2009    27.79            0.05
##   WIND_DIRECTION.deg. ATMO_PRESSURE.hpa. HUMIDITY... RAIN_FALL.mm.
## 1              358.99            1003.15       87.93           894
## 2              239.74            1003.64       83.92           894
## 3              358.99            1004.33       77.96           894
##   SUN_SHINE.hh.mm. BATTERY_VOLTAGE.V.
## 1            10:29              12.49
## 2              0:0              12.49
## 3              0:8              12.78
```

2. Missing value location

Approach :

Using a unique encoding for a particular hour for each day, and creating a sequence of such values,from starting row to ending row, I'm going to locate the missing values by outer_join.

```
# Encoding column, formed by combining date and time
mod_Data$Enc = paste(mod_Data$DATE.GMT., mod_Data$TIME.GMT., sep =" ")
# using as.Date() function, doesn't work here, because
# the Date format is not in standard format, in all rows
# so, we get this error :
# Error in charToDate(x) :
#  character string is not in a standard unambiguous format
# Hence we use alternate way

mod_Data$Enc = as.POSIXct(mod_Data$Enc, '%m/%d/%Y %H', tz = "GMT")
# checking how the created column looks like
head(mod_Data$Enc, 3)
```

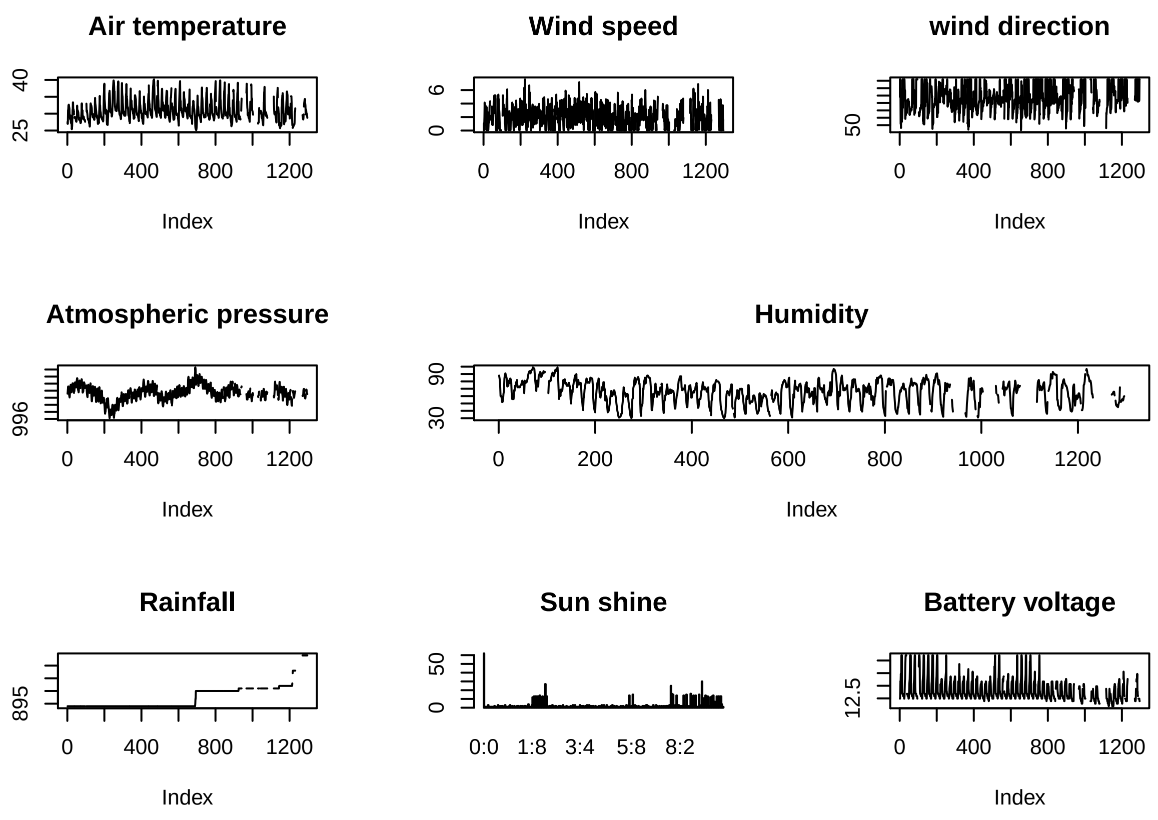
```
## [1] "2009-05-15 00:00:00 GMT" "2009-05-15 01:00:00 GMT"
## [3] "2009-05-15 02:00:00 GMT"
```

```r
start_index = 1
end_index = nrow(mod_Data)
Enc = seq(mod_Data$Enc[start_index], mod_Data$Enc[end_index], by = 'hour')
Enc = as.POSIXct(Enc, '%m/%d/%Y %H', tz = "GMT")
# making Enc column to a data frame
df = data.frame(Enc = Enc)
# outer-joining the df and mod_Data on Enc column
fin_Data = merge(df, mod_Data, by.x = 'Enc', by.y = 'Enc', all.x = TRUE)
# Dropping the TIME.GMT., DATE.GMT, TIME.IST,
# DATE.IST columns, (as we don't use them further)
fin_Data_1 = subset(fin_Data, select = -c(TIME.GMT.,DATE.GMT.,TIME.IST.,DATE.IST.))
# looking few rows of the data frame created
head(fin_Data_1,3)
```

```
##                   Enc AIR_TEMP WIND_SPEED.m.s. WIND_DIRECTION.deg.
## 1 2009-05-15 00:00:00    27.49            0.05              358.99
## 2 2009-05-15 01:00:00    26.96            1.03              239.74
## 3 2009-05-15 02:00:00    27.79            0.05              358.99
##   ATMO_PRESSURE.hpa. HUMIDITY... RAIN_FALL.mm. SUN_SHINE.hh.mm.
## 1            1003.15       87.93           894            10:29
## 2            1003.64       83.92           894              0:0
## 3            1004.33       77.96           894              0:8
##   BATTERY_VOLTAGE.V.
## 1              12.49
## 2              12.49
## 3              12.78
```

In the final Data frame created `fin_Data_1`, the missing values are represented by `NA`, by default.

```r
# suppressing warnings
options(warn = -1)
# ploting the variables with missing data
layout(matrix(c(1,2,3,4,5,5,6,7,8), nrow = 3, ncol = 3, byrow = TRUE))
plot(fin_Data_1[,2], type = 'l', main = "Air temperature", ylab = "")
plot(fin_Data_1$WIND_SPEED.m.s., type = 'l', main = "Wind speed", ylab = "")
plot(fin_Data_1$WIND_DIRECTION.deg., type = 'l', main = "wind direction", ylab = "")
plot(fin_Data_1$ATMO_PRESSURE.hpa., type = 'l', main = "Atmospheric pressure", ylab = "")
plot(fin_Data_1$HUMIDITY..., type = 'l', main = "Humidity", ylab = "")
plot(fin_Data_1$RAIN_FALL.mm., type = 'l', main = "Rainfall", ylab = "")
plot(fin_Data_1$SUN_SHINE.hh.mm., type = 'l', main = "Sun shine", ylab = "")
plot(fin_Data_1$BATTERY_VOLTAGE.V., type = 'l', main = "Battery voltage", ylab = "")
```



3. Imputation

3.1. using mice

```r
library(mice, warn.conflicts=F, quietly=T)
# excluding sunshine (since it is a time variable, mice can't handle it)
Imp_op = mice(data = fin_Data_1[,c(2,3,4,5,6,7,9)], method = 'norm.predict', m = 2, verbose = FALSE)
data_imputed = complete(Imp_op)
# creating a mode function for categorical variables
Mode <- function (x, na.rm) {
    xtab <- table(x)
    xmode <- names(which(xtab == max(xtab)))
    if (length(xmode) > 1)
        xmode <- ">1 mode"
    return(xmode)
}
# imputing the sunshine variable
data_imputed$SunShine = fin_Data_1$SUN_SHINE.hh.mm.
data_imputed$SunShine[is.na(data_imputed$SunShine)] = Mode(data_imputed$SunShine, na.rm = TRUE)
# attaching the encoding column
Imputed_data = cbind(Enc, data_imputed)
```

splitting into train and test data

```
# 1200 train data & 96 test data
# train data
train_data = Imputed_data[1:1200,]
# test data
test_data = Imputed_data[1201:1296,]
```

4. Creating ts objects of RH and temperature variables

RH and temperature are converted to time-series objects using `ts` attribute

```
# train data
Air_temp = ts(train_data[,2])
RH = ts(train_data$HUMIDITY...)
# test data
Air_temp_test = ts(test_data[,2])
RH_test = ts(test_data$HUMIDITY...)
```

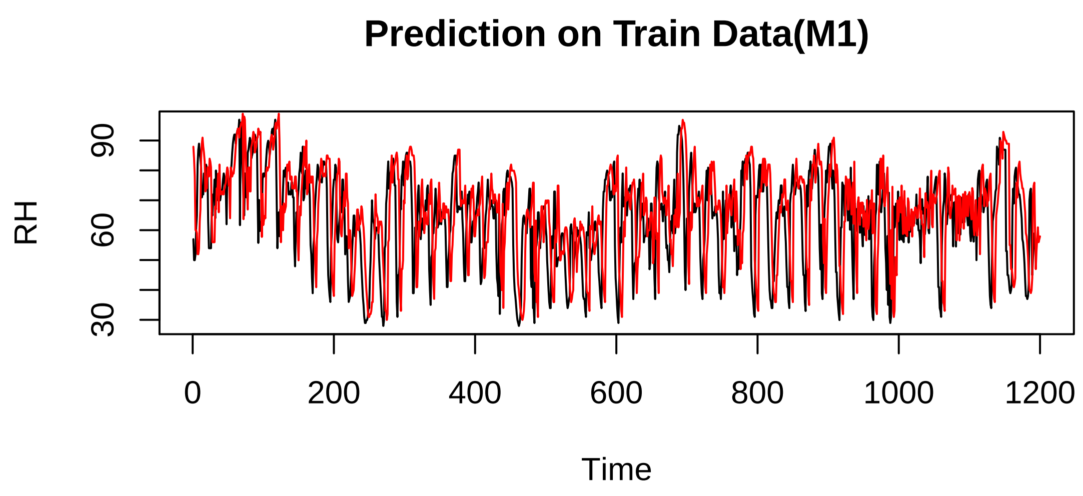
5. Fuzzy time series model (M_1)

Here we build a FTS model using AnalyzeTS package and to model it, we use `fuzzy.ts1` function. We use `Chen` method.

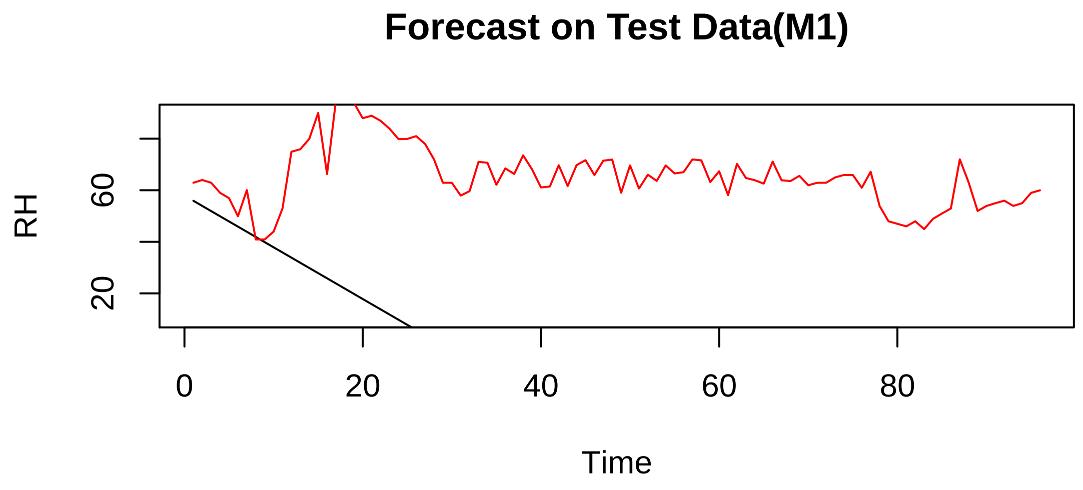
```
library(AnalyzeTS, warn.conflicts=F, quietly=T)
```

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

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

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

```
options(warn = -1)
#fuzzy1 = fuzzy.ts1(ts(Imputed_data$HUMIDITY...),n = 10,D1 = 10,D2 = 10,type = "Chen",trace = 1)
fuzzy1 = fuzzy.ts2(RH,n=5,w=5,C=0,forecast=96,type="Abbasov-Mamedova",trace = TRUE)

pred_train = fuzzy1$table4$interpolate[1:1200]
plot(ts(pred_train),type = 'l', main = 'Prediction on Train Data(M1)',ylab = 'RH')
lines(RH, col = 'red')
```



```
forecast_M1 = fuzzy1$table5$forecast
plot(ts(forecast_M1),type = 'l', ylim = c(10, 90), main = 'Forecast on Test Data(M1)', ylab = 'RH')
lines(RH_test, col = 'red')
```



```
# reporting accuracies
print("The train accuracy metrics for M1 are :")
```

```
## [1] "The train accuracy metrics for M1 are :"
```

```
print(fuzzy1$accuracy)
```

```
##                      ME   MAE   MPE  MAPE    MSE  RMSE        U
## Abbasov.Mamedova 2.004 5.728 2.292 9.201 66.166 8.134 1.031806
```

```r
library(DMwR, warn.conflicts=F, quietly=T)
test_eval = regr.eval(RH_test, forecast_M1)
print("The test accuracy metrics for M1 are :")
```

```
## [1] "The test accuracy metrics for M1 are :"
```

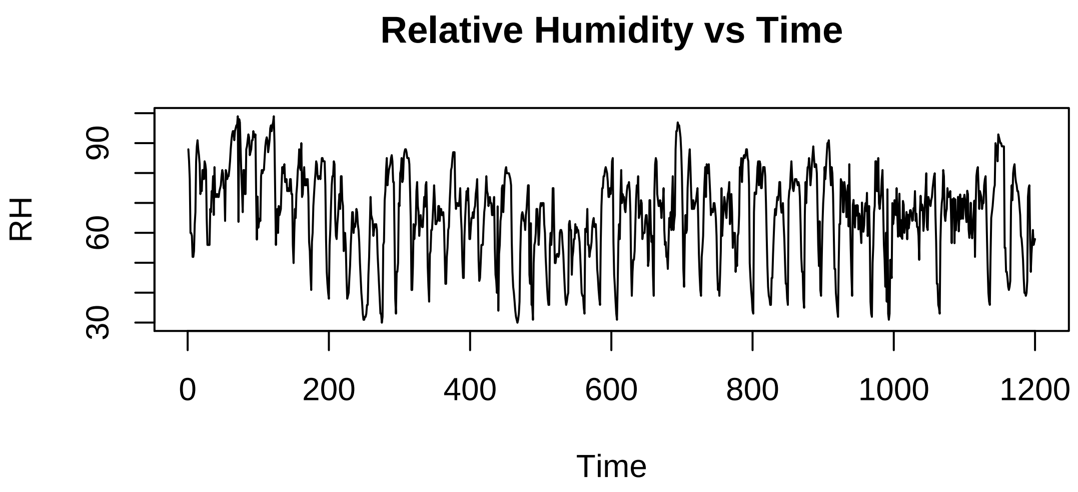
```r
print(test_eval)
```

```
##          mae          mse         rmse         mape
##   104.41099 13623.60730   116.72021      1.67389
```

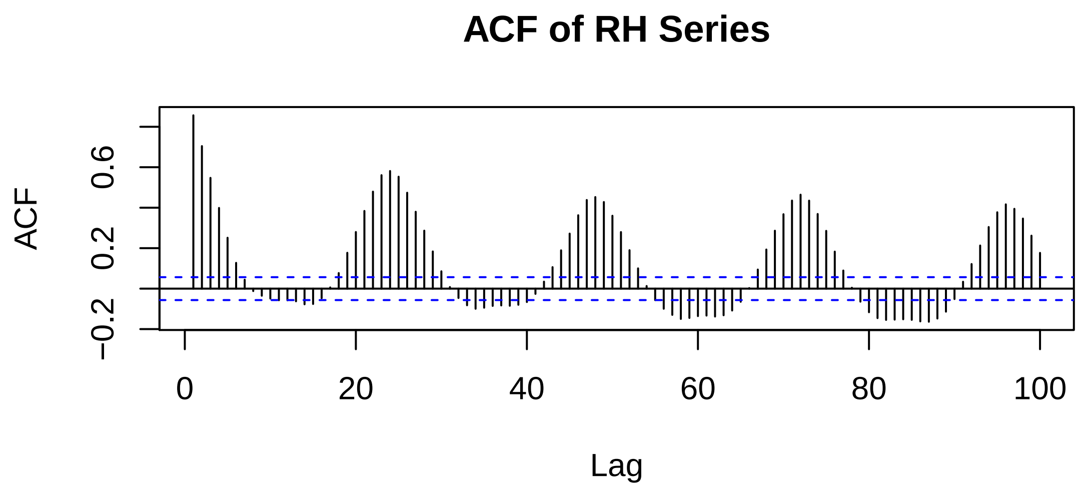
From the above plots, we infer that, the fuzzy time series is non-linear modelling process, it didn't perform well on test data. The black lines are predicted values, the red lines are actual values after imputing.

6. Linear SARIMA model (model M_2)

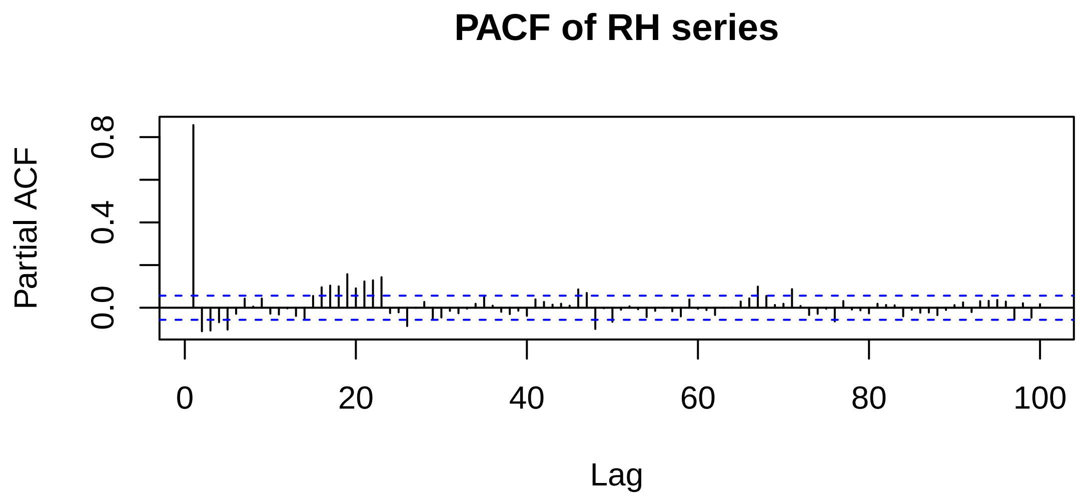
```r
library(TSA, warn.conflicts=F, quietly=T)
# ploting the series

plot(RH, main = 'Relative Humidity vs Time')
```



```r
# ACF of the Relative humidity series
acf(RH, lag.max = 100, main = 'ACF of RH Series')
```



```r
# PACf of the Relative humidity series
pacf(RH, lag.max = 100, main = 'PACF of RH series')
```



```r
library(tseries, warn.conflicts=F, quietly=T)
kpss.test(RH)
```

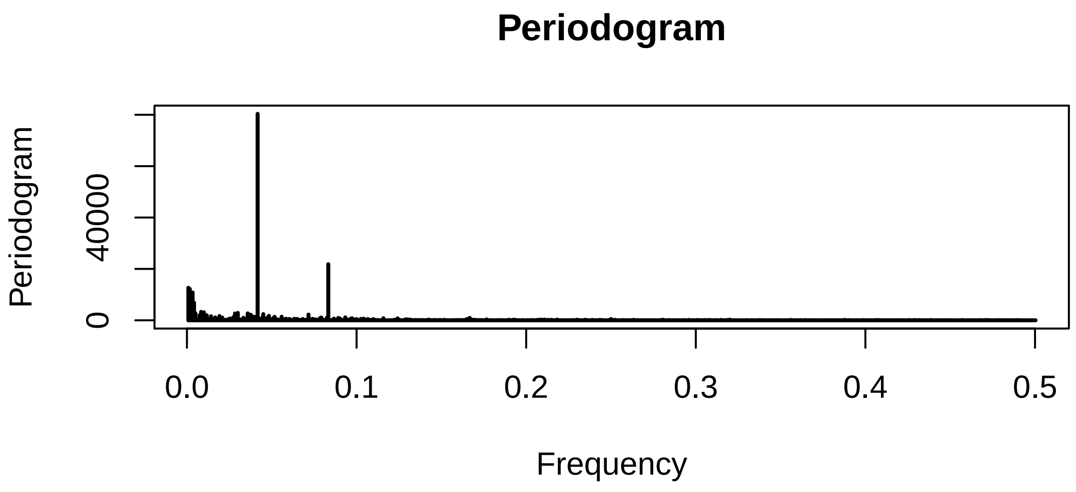
```
##
##  KPSS Test for Level Stationarity
##
## data:  RH
## KPSS Level = 0.7487, Truncation lag parameter = 7, p-value = 0.01
```

```r
adf.test(RH)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  RH
## Dickey-Fuller = -9.2248, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

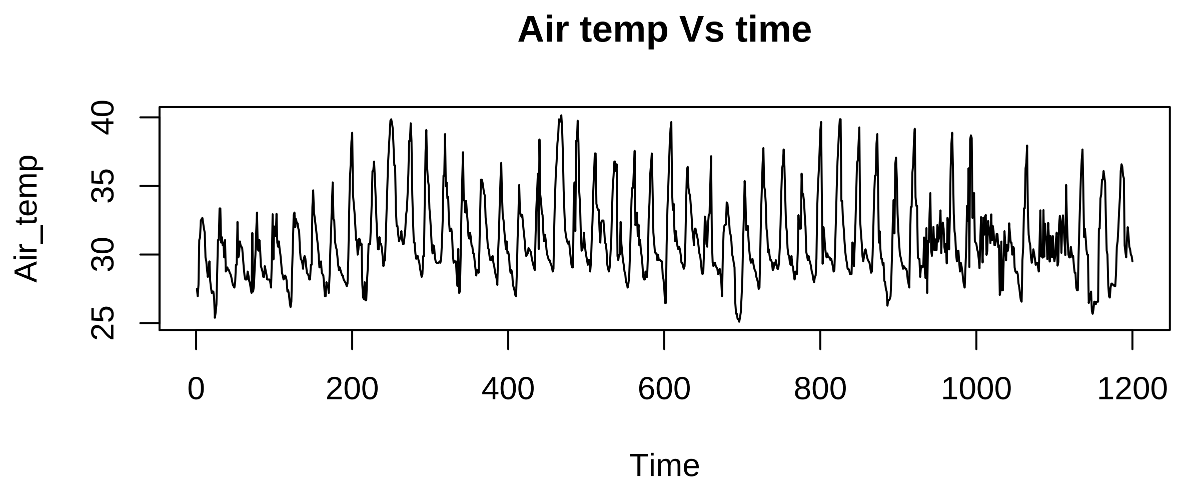
From the ACF of the data, we see that the data has periodic component. From the KPSS test and ADF test, we conclude that the `RH` series doesn't have trend.(i.e. determininstic periodic component is not present). So we confirm this by it's periodogram to check if it is a deterministic component or seasonality. It is a non-zero mean process.

```r
# Periodogram of RH series
periodogram(RH, main = "Periodogram")
```

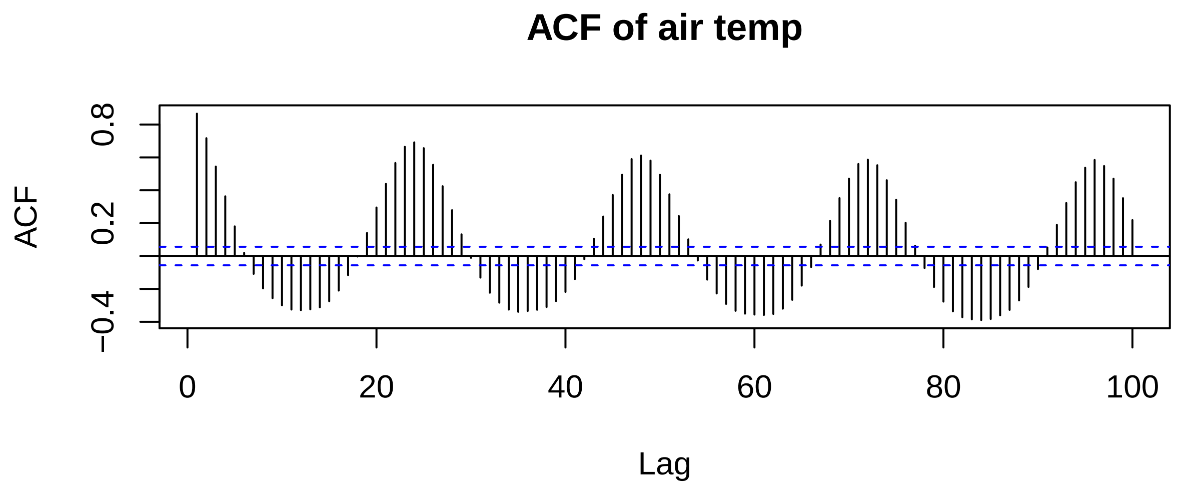


From the periodogram, we conclude that, the data has a seasonal component with period 24, we should build a multiplicative model.

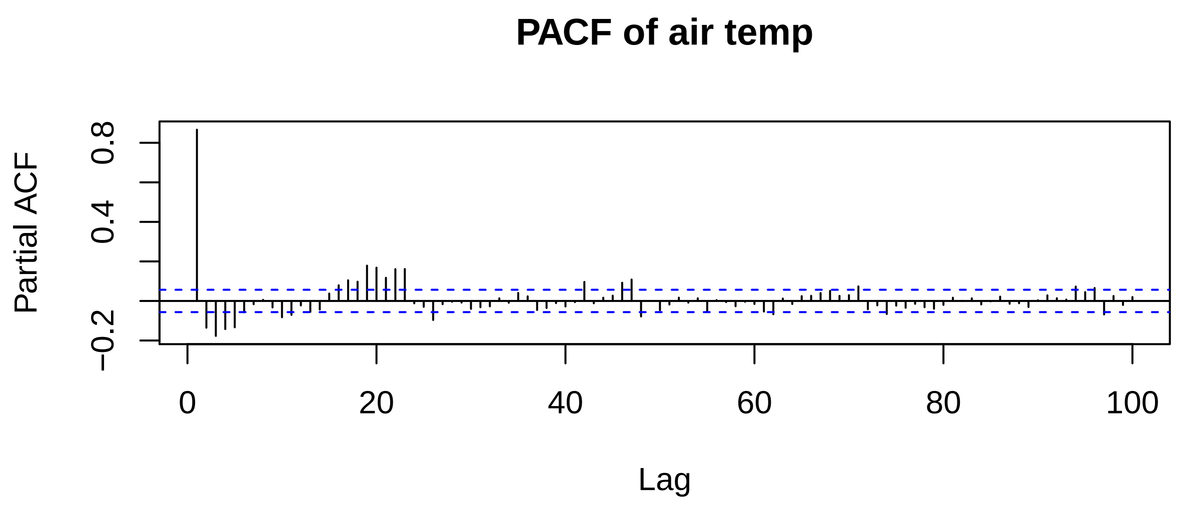
```r
# plotting the air temperature plots
plot(Air_temp, main = "Air temp Vs time")
```



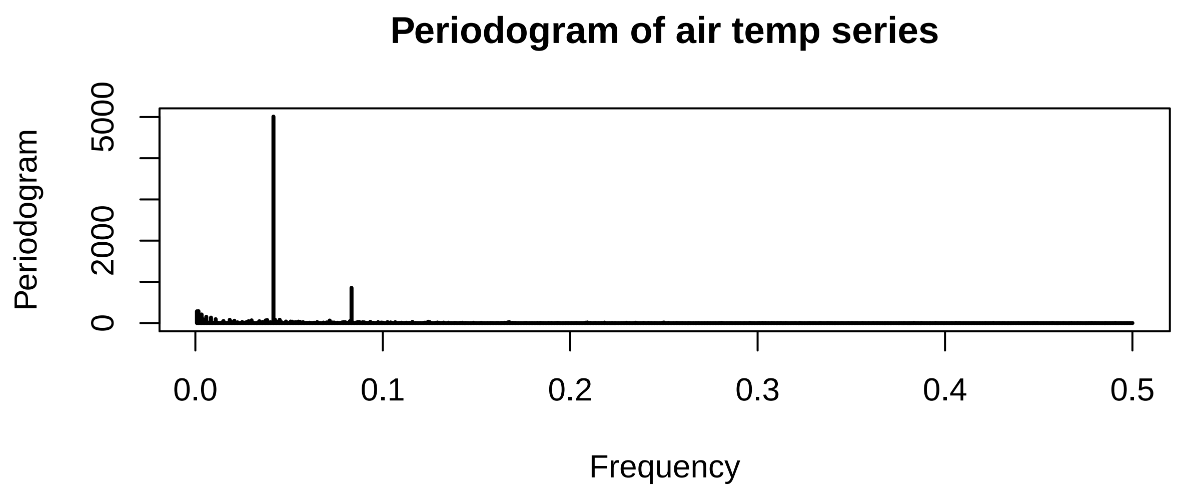
```r
# ACF
acf(Air_temp, main = "ACF of air temp", lag.max = 100)
```



```r
# PACF
pacf(Air_temp, main = "PACF of air temp", lag.max = 100)
```



```r
# periodogram
periodogram(Air_temp, main = "Periodogram of air temp series")
```



```r
library(tseries, warn.conflicts=F, quietly=T)
kpss.test(Air_temp)
```

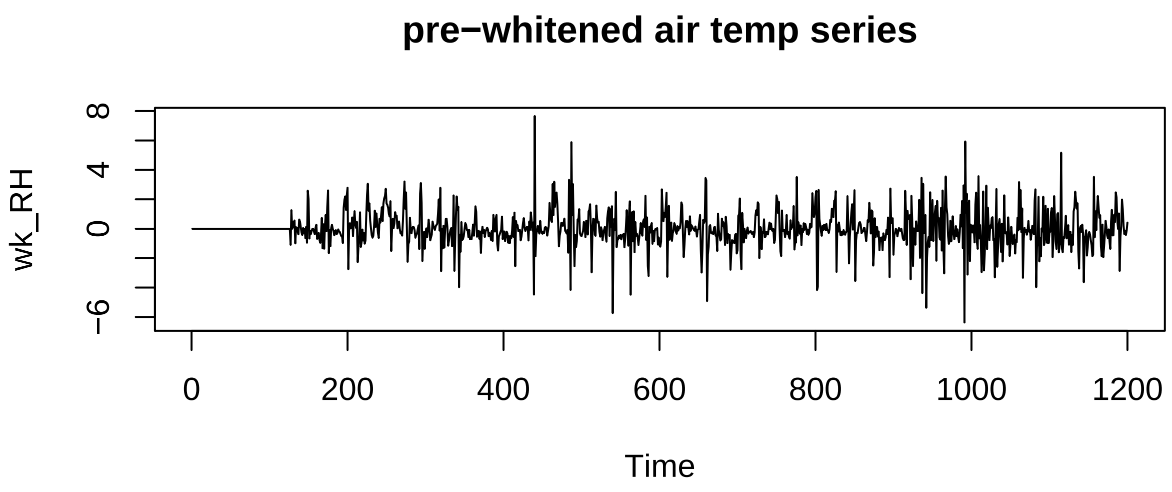
```
##
##  KPSS Test for Level Stationarity
##
## data:  Air_temp
## KPSS Level = 0.28005, Truncation lag parameter = 7, p-value = 0.1
```

```r
adf.test(Air_temp)
```

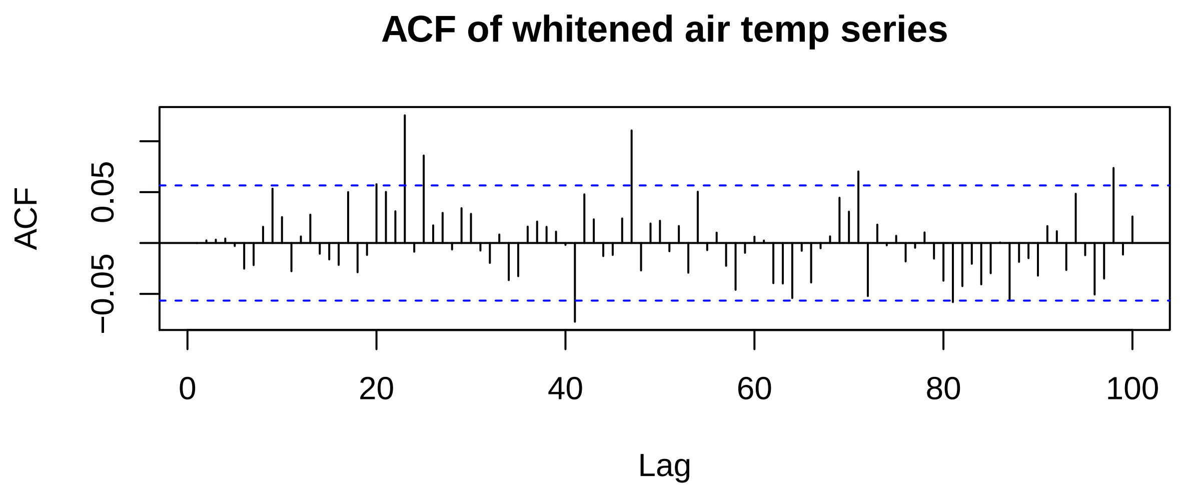
```
##
##  Augmented Dickey-Fuller Test
##
## data:  Air_temp
## Dickey-Fuller = -12.105, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

From the ACF of the data, we see that the data has periodic component. From the KPSS test and ADF test, we conclude that the `Air_temp` series doesn't have trend.(i.e. determininstic periodic component is not present) at 10 % significance. It was confirmed by it's periodogram. From the periodogram, we conclude that, the data has a seasonal component with period `24`, we should build a multiplicative model. Both the series (air temperature and Relative humidity) are not white, so pre-whitening of atleast one of the series should be done.(Here, I'm whitening both series)

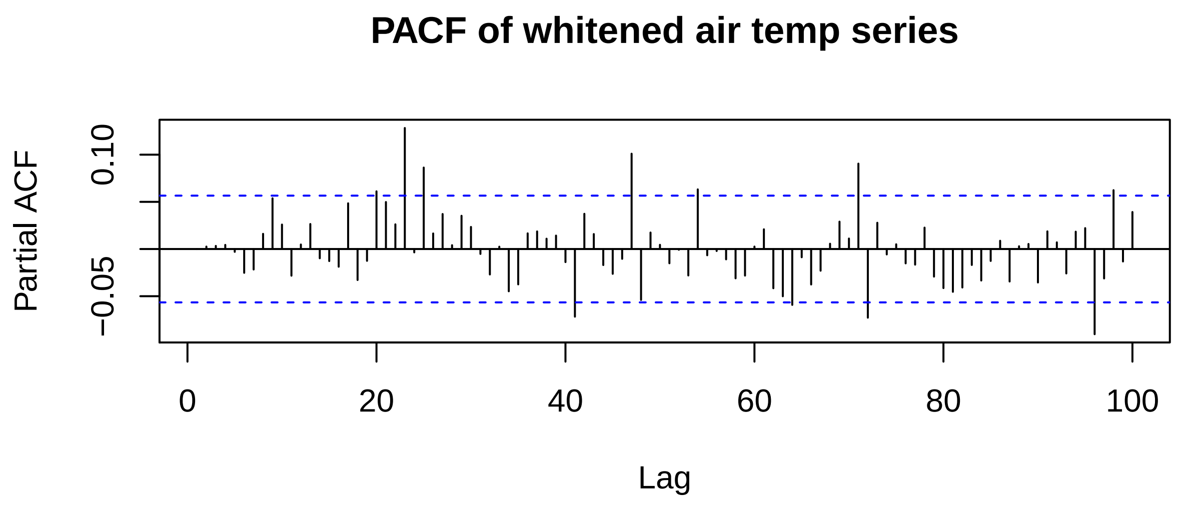
```r
# whitenning using high order AR and seasonal AR model
library(forecast, warn.conflicts=F, quietly=T)
library(stats, warn.conflicts=F, quietly=T)
high_ar = Arima(Air_temp, order = c(5,0,0), seasonal =
                  list(order = c(5,0,0), period = 24), method = "CSS")
# residual analysis
wk_RH = high_ar$residuals
plot(wk_RH,main = "pre-whitened air temp series")
```



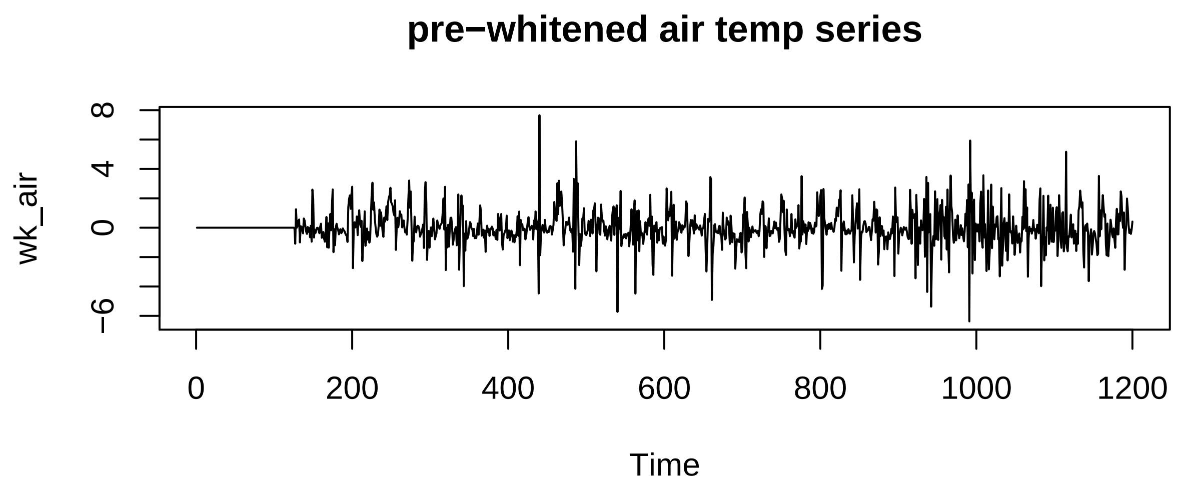
```r
# ACF
acf(wk_RH, main = "ACF of whitened air temp series", lag.max = 100)
```



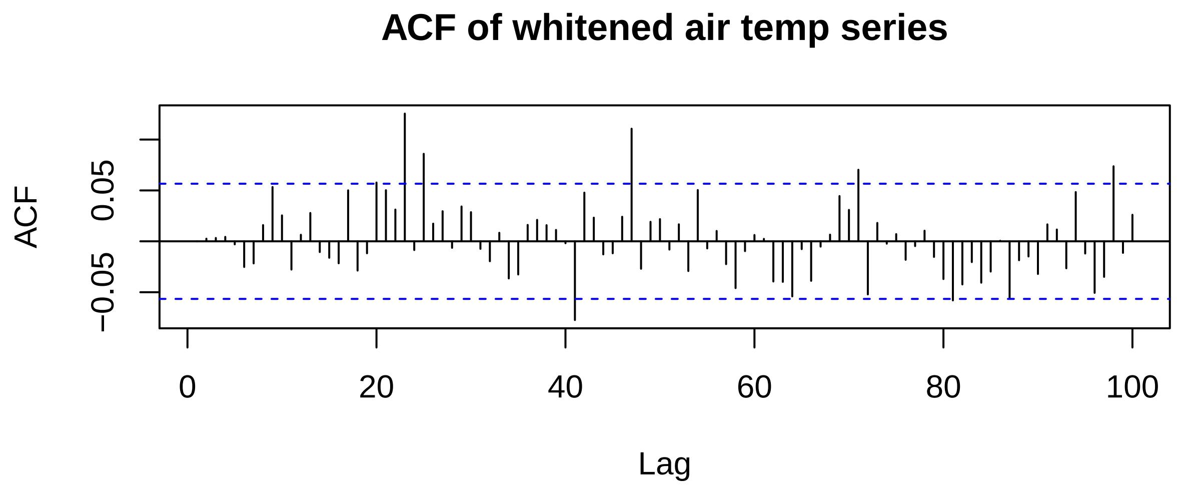
```
# PACF
pacf(wk_RH, main = "PACF of whitened air temp series", lag.max = 100)
```



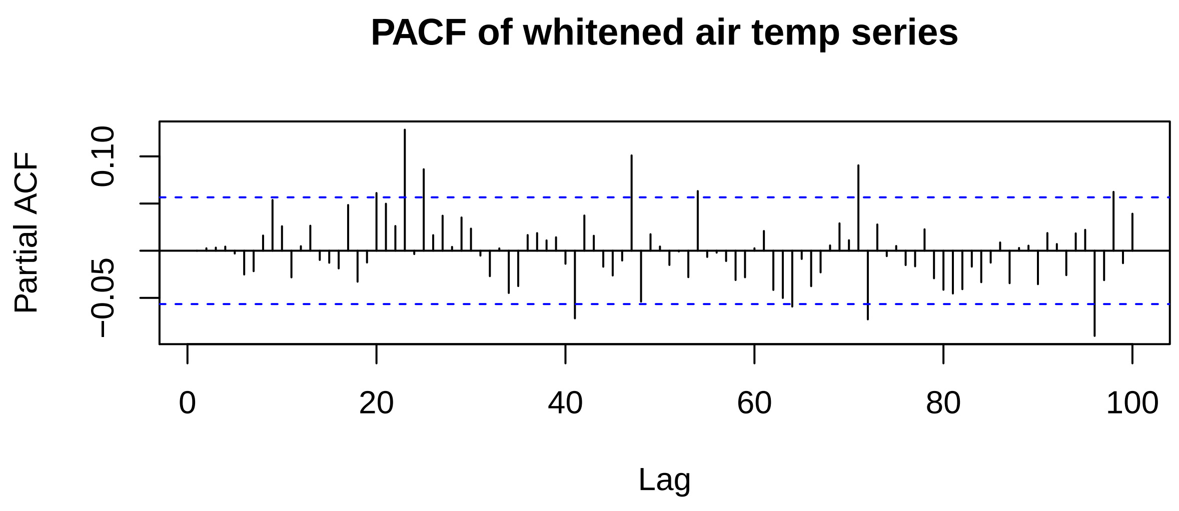
```
high_ar2 = Arima(Air_temp, order = c(5,0,0), seasonal =
                   list(order = c(5,0,0), period = 24), method = "CSS")
# residual analysis
wk_air = high_ar$residuals
plot(wk_air,main = "pre-whitened air temp series")
```



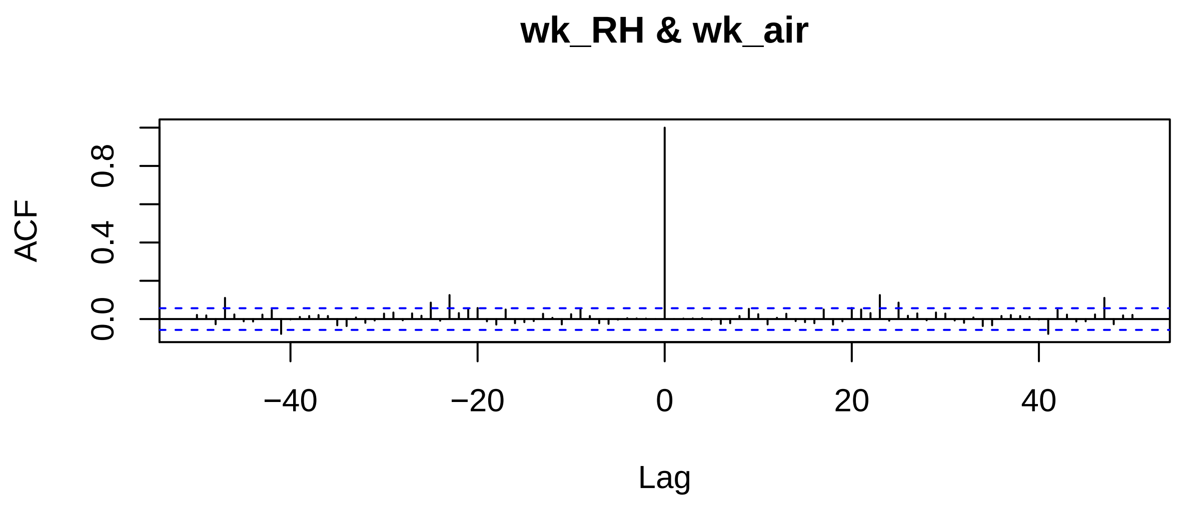
```r
# ACF
acf(wk_air, main = "ACF of whitened air temp series", lag.max = 100)
```



```r
# PACF
pacf(wk_air, main = "PACF of whitened air temp series", lag.max = 100)
```



```r
ccf(wk_RH, wk_air, lag.max = 50, type = 'correlation')
```



From the CCF, the highest peak is found at $lag = 0$ and other values seems to be insignificant.

```r
# Assuming there is no delay between RH and Air_temp
# converge to SARIMA(1,0,0)(2,0,0)
sarima_mod = Arima(RH, order = c(1,0,0), seasonal = list(
  order = c(2,0,0), period = 24), xreg = Air_temp, method = 'CSS' )
# model coefficients are
sarima_mod
```

```
## Series: RH
## Regression with ARIMA(1,0,0)(2,0,0)[24] errors
##
```

```
## Coefficients:
##          ar1    sar1    sar2  intercept     xreg
##       0.8450  0.0958  0.0908   206.4568  -4.5141
## s.e.  0.0158  0.0293  0.0287     2.6586   0.0803
##
## sigma^2 estimated as 14.63:  part log likelihood=-3335.16
```

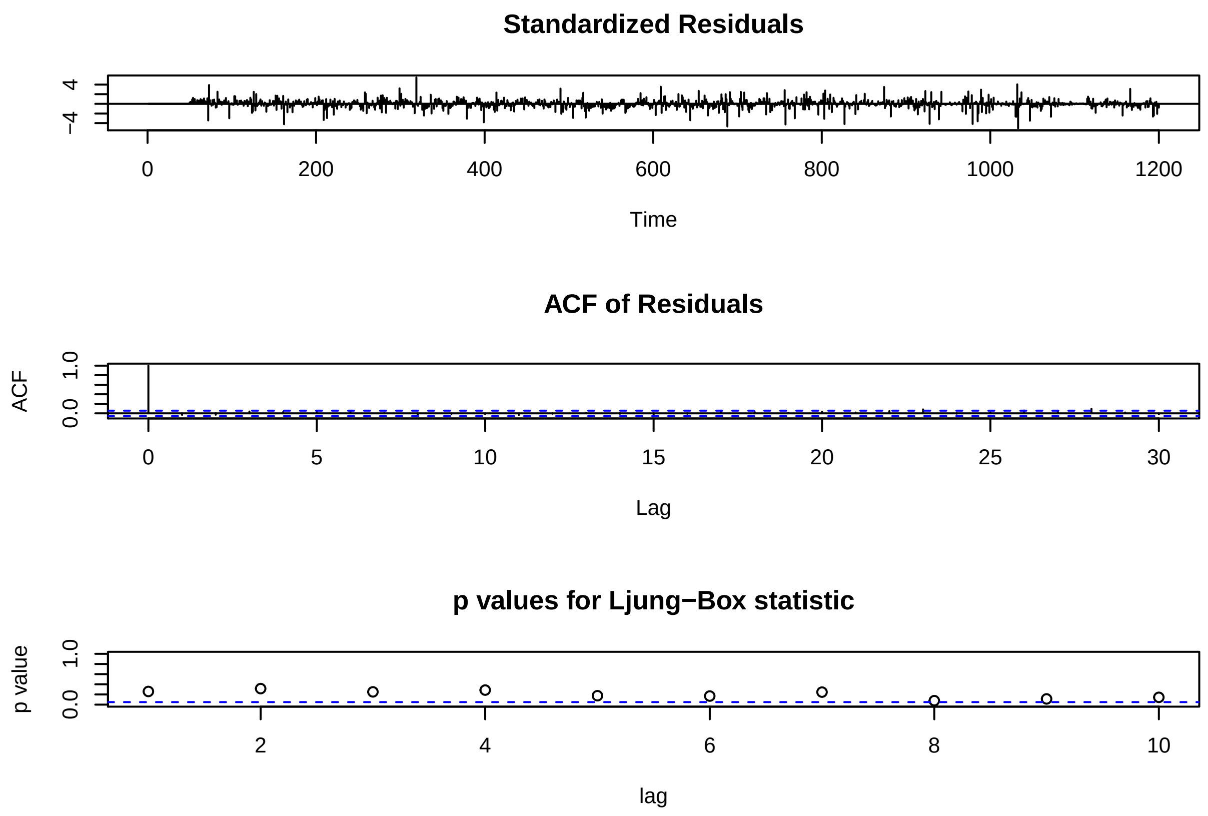
```
print("confidence intervals are :")
```

```
## [1] "confidence intervals are :"
```

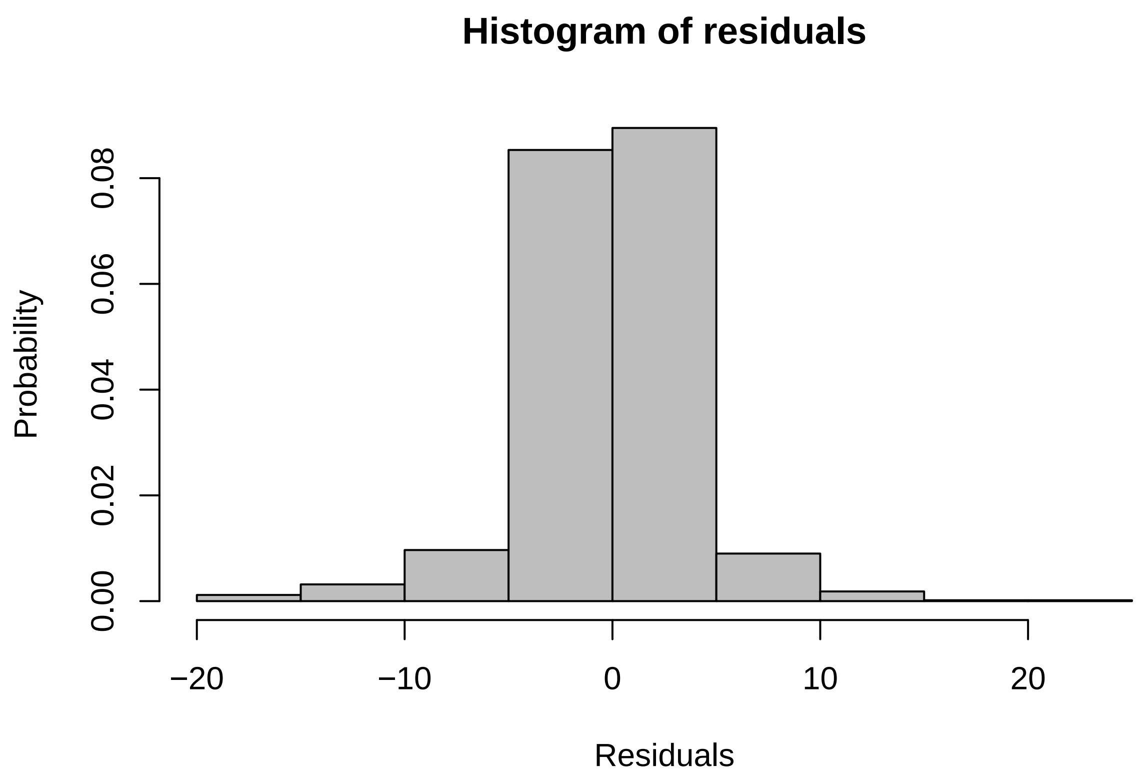
```
confint(sarima_mod)
```

```
##                  2.5 %      97.5 %
## ar1           0.81399727   0.8760824
## sar1          0.03845049   0.1532286
## sar2          0.03454941   0.1471172
## intercept 201.24612812 211.6675492
## xreg        -4.67156004  -4.3566817
```

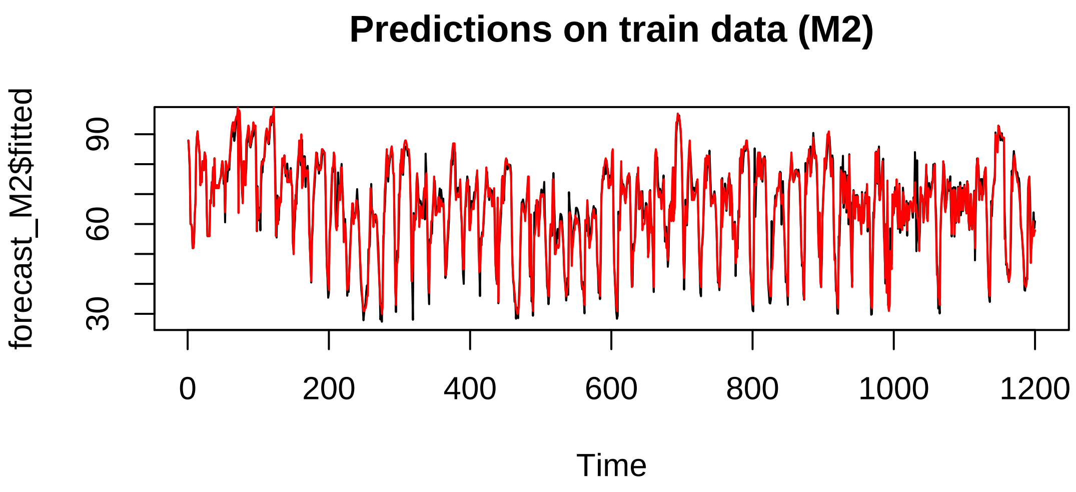
```
# diagnostics
tsdiag(sarima_mod)
```



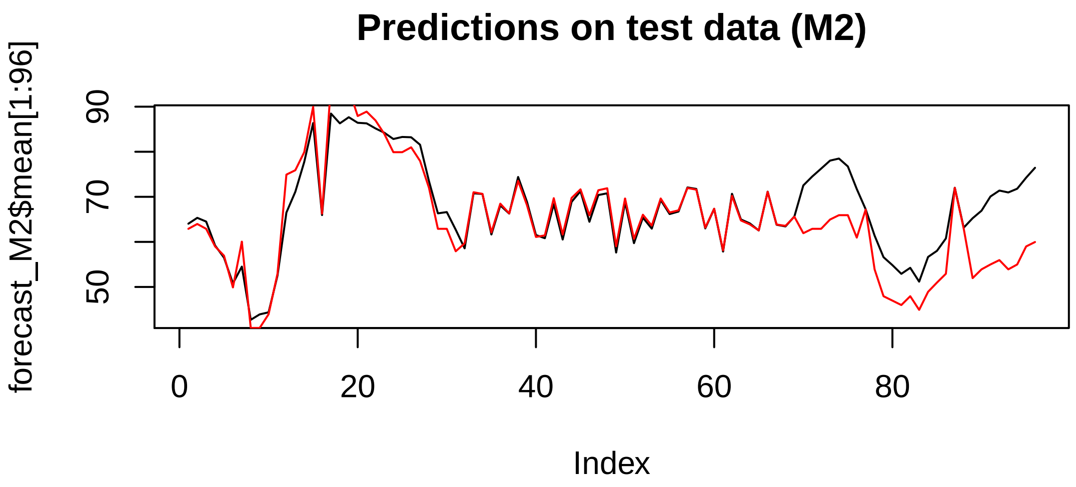
```r
# histogram of residuals for normality check
hist(sarima_mod$residuals, probability = T, xlab = "Residuals",ylab = "Probability",
     main = "Histogram of residuals", col = "gray")
```



```r
# Forecast For model M2
forecast_M2 = forecast(sarima_mod, h = 96, xreg = Air_temp_test)
# On train data
plot(forecast_M2$fitted, type = 'l', main = "Predictions on train data (M2)")
lines(RH, col = 'red')
```



```r
# On test data
plot(forecast_M2$mean[1:96], type = 'l', main = "Predictions on test data (M2)")
lines(RH_test, col = 'red')
```



```r
# accuracy metrics
train_eval = regr.eval(RH, forecast_M2$fitted)
print("The train accuracy metrics for M2 are :")
```

```
## [1] "The train accuracy metrics for M2 are :"
```

```r
print(train_eval)
```

```
##         mae         mse        rmse        mape
##  2.54217008 14.57095909  3.81719257  0.04093752
```

```r
test_eval = regr.eval(RH_test, forecast_M2$mean[1:96])
print("The test accuracy metrics for M2 are :")
```

```
## [1] "The test accuracy metrics for M2 are :"
```

```r
print(test_eval)
```

```
##         mae         mse        rmse        mape
##  3.98526684 40.02514782  6.32654312  0.06680753
```

7. Comparing forecasts of models M1 and M2

On the basis of accuracy metrics on test data, we select M2 (linear SARIMA with exogenous variable) as the better model, because it could generalize the test data better.

8. Replacing missing values and rebuilding the model

8.1 Replacing the missing values of RH

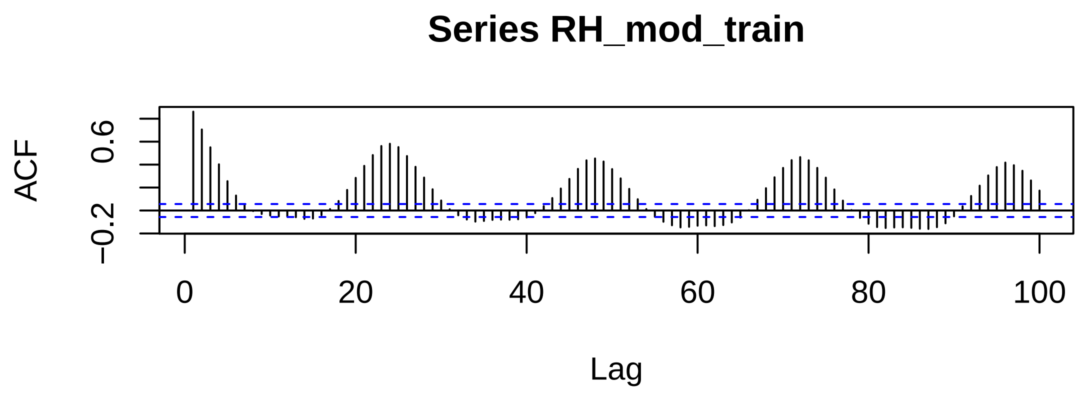
Here we replace the missing values of RH by the values predicted by the model M2 (best model obtained in step 7) and rebuild the model.

```r
# Replacing the missing values of RH
RH_replaced = fin_Data_1$HUMIDITY...
# missing indices
miss_indices = which(is.na(RH_replaced))
# updating with predicted values
for (i in miss_indices){
  # checking if the index is in train part or forecast part
  if (i < 1200){
    RH_replaced[i] = forecast_M2$fitted[i]
  }
  else{
    RH_replaced[i] = forecast_M2$mean[i - 1200]
  }
}
```

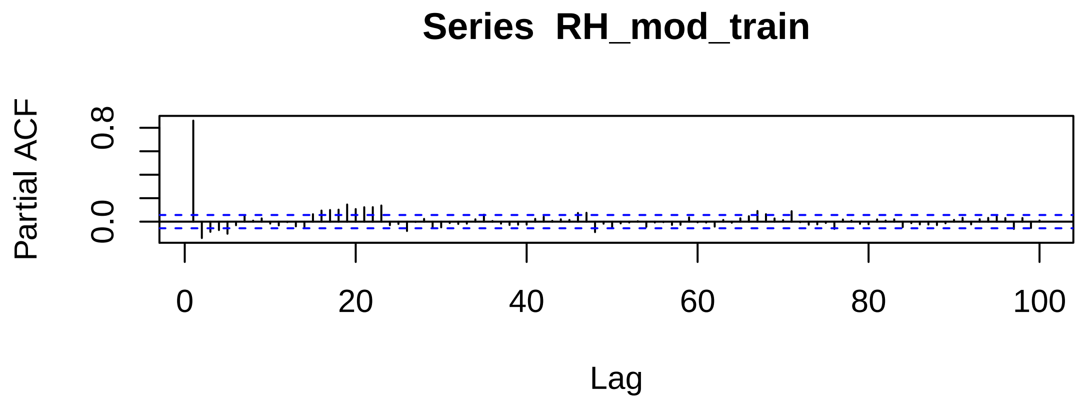
8.2 Rebuilding the model

```r
# train and test
RH_mod_train = RH_replaced[1:1200]
RH_mod_test = RH_replaced[1201:1296]
```

```r
# acf
acf(RH_mod_train, lag.max = 100)
```



```r
# pacf
pacf(RH_mod_train, lag.max = 100)
```



```r
# Assuming there is no delay between RH and Air_temp
# converge to SARIMA(1,0,0)(2,0,0)
sarima_mod1 = Arima(RH_mod_train, order = c(1,0,0), seasonal =
                      list(order = c(2,0,0), period = 24), xreg = Air_temp, method = 'CSS' )
# model coefficients are
sarima_mod1
```

```
## Series: RH_mod_train
## Regression with ARIMA(1,0,0)(2,0,0)[24] errors
##
## Coefficients:
##          ar1    sar1    sar2  intercept     xreg
##       0.8618  0.1130  0.0927   206.0180  -4.5021
## s.e.  0.0150  0.0292  0.0286     2.5734   0.0764
##
## sigma^2 estimated as 13.23:  part log likelihood=-3274.88
```

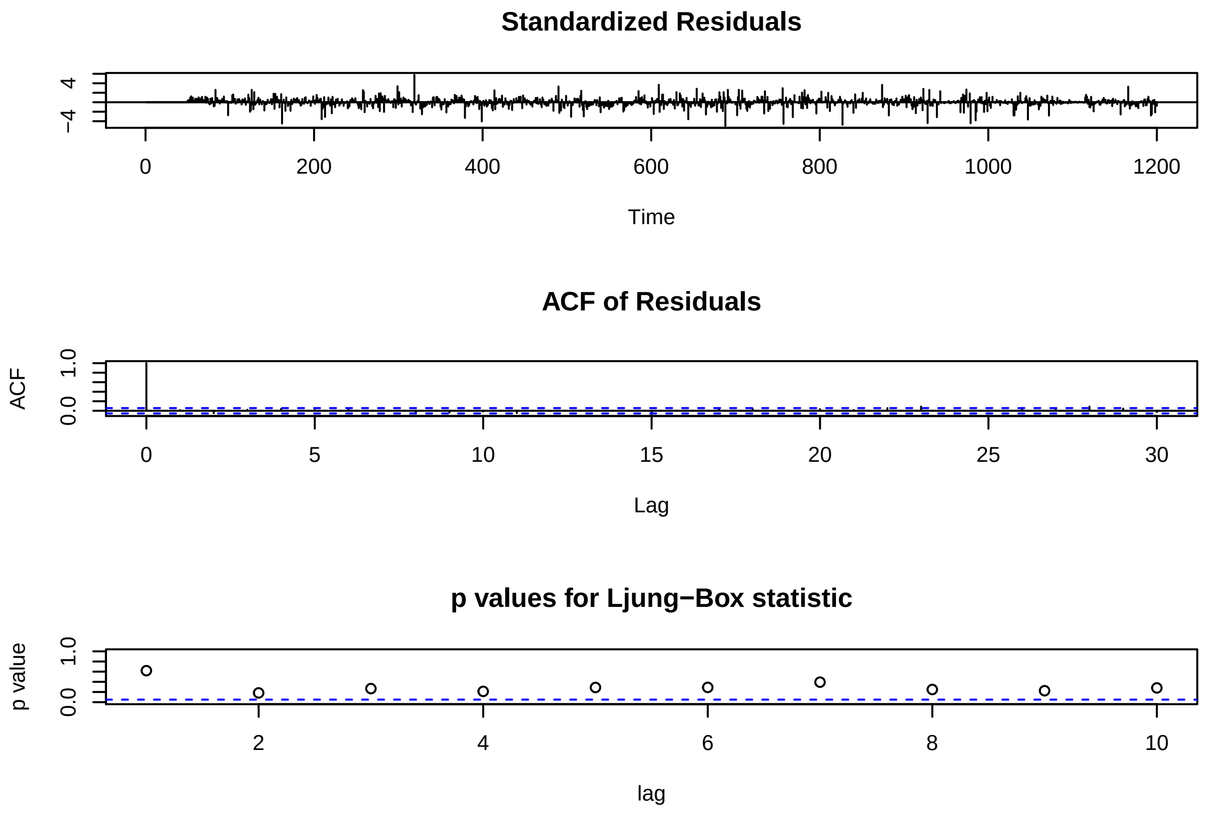
```r
#AIC(sarima_mod)
print("confidence intervals are :")
```

```
## [1] "confidence intervals are :"
```

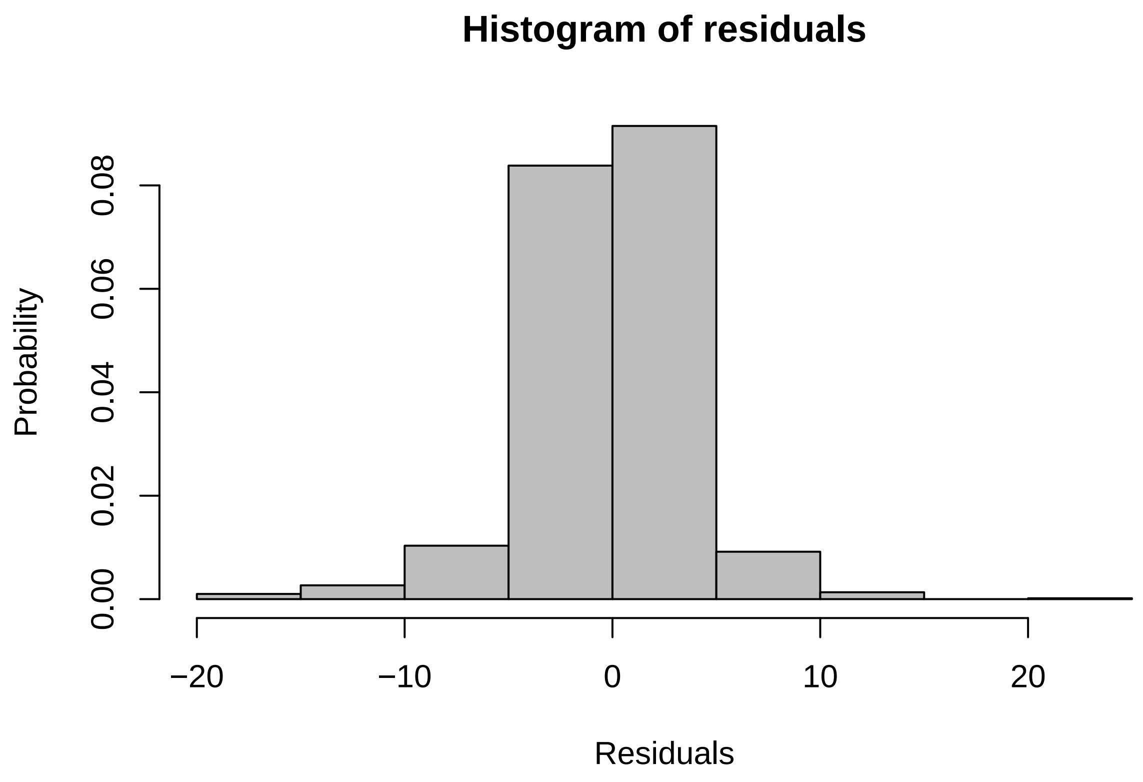
```r
confint(sarima_mod1)
```

```
##                  2.5 %      97.5 %
## ar1          0.83250403   0.8911565
## sar1         0.05579337   0.1701976
## sar2         0.03660907   0.1487511
## intercept 200.97415393 211.0618157
## xreg        -4.65183155  -4.3524022
```

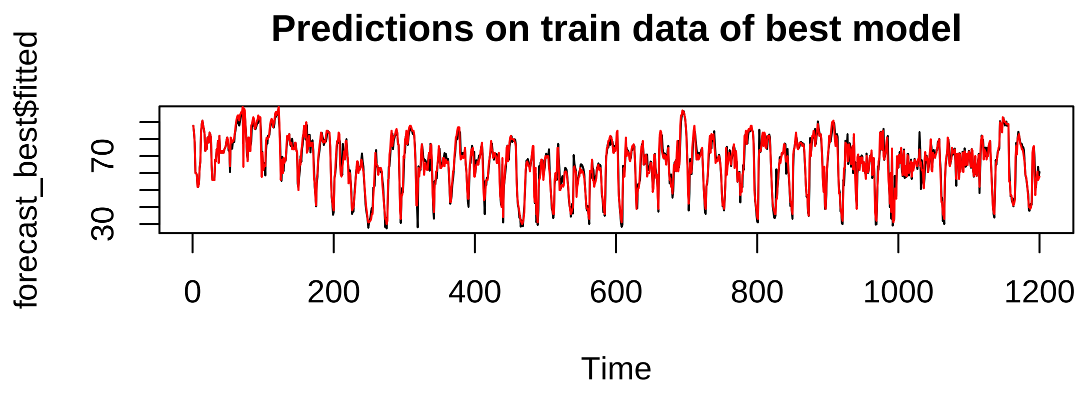
```r
# diagnostics
tsdiag(sarima_mod1)
```



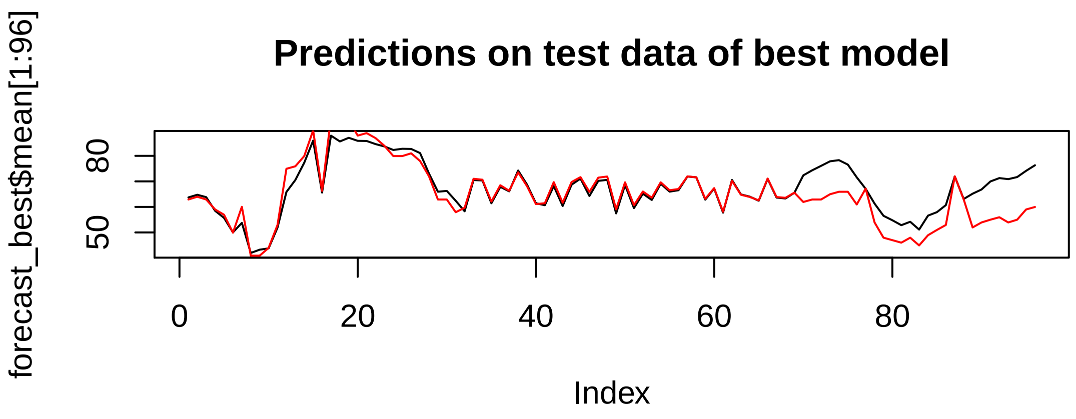
```r
# histogram of residuals for normality check
hist(sarima_mod1$residuals, probability = T, xlab = "Residuals",
     ylab = "Probability",main = "Histogram of residuals", col = "gray")
```



```r
# Forecast For new model
forecast_best = forecast(sarima_mod1, h = 96, xreg = Air_temp_test)
# On train data
plot(forecast_best$fitted, type = 'l', main = "Predictions on train data of best model")
lines(RH, col = 'red')
```



```r
# On test data
plot(forecast_best$mean[1:96], type = 'l', main = "Predictions on test data of best model")
lines(RH_test, col = 'red')
```



```
# accuracy metrics
train_eval = regr.eval(RH, forecast_best$fitted)
print("The train accuracy metrics for best model are :")
```

```
## [1] "The train accuracy metrics for best model are :"
```

```
print(train_eval)
```

```
##         mae         mse        rmse        mape
##  2.54015781 14.23203181  3.77253652  0.04113901
```

```
test_eval = regr.eval(RH_test, forecast_best$mean[1:96])
print("The test accuracy metrics for best model are :")
```

```
## [1] "The test accuracy metrics for best model are :"
```

```
print(test_eval)
```

```
##         mae         mse        rmse        mape
##  4.01267128 39.85616871  6.31317422  0.06685666
```

8.3 Inferences

The model built after replacing the `NA` values with the predicted values of linear SARIMA model obtained in part `6`. It performs better than both the models obtained in part 5 and 6, (inferred from the reduction in the metrics MAE,MSE, . . .). `This is because, the imputation method used in part 3, might not have considered the time-correlation`.where as the linear sarima model takes into account both time correlation and also the temperature exogenous variable.

The model didn't change that much but, it's coefficients changed slightly, and their standard errors decreased. Even the s.e of the error also reduced.