Forecasting Final Project (MATH1307)

Rakshit Chandna s3956924

2023-10-22

Table of Contents

Necessary Libraries

```
library(TSA)
library(car)
library(carData)
library(lmtest)
library(dplyr)
library(AER)
library(dynlm)
library(corrr)
library(Hmisc)
library(forecast)
library(dplyr)
library(xts)
library(x12)
library(ggplot2)
library(x13binary)
library(nardl)
library(dLagM)
library(readr)
library(tseries)
library(urca)
library(expsmooth)
```

Introduction

The report for the assignment is segmented into three distinct tasks, each focusing on the analysis of specific time series data. In the initial task, we scrutinize five weekly series from 2010-2020, which include mortality, temperature, particle size of pollutants, and two chemical emissions named chem1 and chem2. The objective is to forecast mortality four weeks into the future. The subsequent task revolves around predicting FFD for the next four years by leveraging various climate indicators. The third task is bifurcated into sections (a) and (b).

The former involves a univariate analysis of RBO, employing individual climate predictors to project its values for the coming three years. Meanwhile, section (b) necessitates another three-year forecast for RBO, this time factoring in the effects of the Australian drought that spanned from 1996 to 2009.

Task-1 Time series analysis & forecast of mortality series

Data Description

From 2010 to 2020, a group of scientists studied the average weekly mortality related to specific diseases in Paris, France. They also examined the city's ambient temperature (measured in degrees Fahrenheit), the size of pollutant particles, and the concentration of harmful chemicals released from vehicles and industrial processes. The 'mort' dataset comprises weekly records for these five categories: mortality rate, temperature, size of pollutant particles, and emissions of two distinct chemicals (chem1 and chem2), spanning over 508 instances during this decade.

Importing the Dataset "Mort.csv"

```
mort_original <- read_csv("C:/Users/Rakshit Chandna/OneDrive/Desktop/DataMain/Forecasting/mor
t.csv")
head(mort_original)
```

```
## # A tibble: 6 × 6
##    ...1 mortality  temp chem1 chem2 `particle size`
##   <dbl>     <dbl> <dbl> <dbl> <dbl>           <dbl>
## 1     1      184.  72.4 11.5   45.8            72.7
## 2     2      191.  67.2  8.92  43.9            49.6
## 3     3      180.  62.9  9.48  32.2            55.7
## 4     4      185.  72.5 10.3   40.4            55.2
## 5     5      174.  74.2 10.6   48.5            66.0
## 6     6      184.  67.9  7.99  48.6            44.0
```

```
class(mort_original)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

From the above output, we can see that the original dataset variables is either in the form of table or in the data-frame. So we will convert it into Time series.

Converting data into Time series

```
mort <- ts(mort_original[,2:6], start = c(2010,7), frequency = 52)
head(mort)
```

```
## Time Series:
## Start = c(2010, 7)
## End = c(2010, 12)
## Frequency = 52
##          mortality  temp chem1 chem2 particle size
## 2010.115    183.63 72.38 11.51 45.79         72.72
## 2010.135    191.05 67.19  8.92 43.90         49.60
## 2010.154    180.09 62.94  9.48 32.18         55.68
## 2010.173    184.67 72.49 10.28 40.43         55.16
## 2010.192    173.60 74.25 10.57 48.53         66.02
## 2010.212    183.73 67.88  7.99 48.61         44.01
```

Converting each variable of data set into Time series on weekly basis from the year 2010

```
Mort <- ts(mort_original$mortality,start = c(2010,1),frequency = 52)

Temp <- ts(mort_original$temp,start = c(2010,1),frequency = 52)

Chem1 <- ts(mort_original$chem1,start = c(2010,1),frequency = 52)

Chem2 <- ts(mort_original$chem2,start = c(2010,1),frequency = 52)

Size <- ts(mort_original$`particle size`,start = c(2010,1),frequency = 52)
```

Checking Class

```
cbind(Variable= c("Mortality","Temp","Chem1","Chem2","Particle size"), Class = c(class(Mort),
class(Temp),class(Chem1),class(Chem2),class(Size)))
```

```
##      Variable        Class
## [1,] "Mortality"     "ts"
## [2,] "Temp"          "ts"
## [3,] "Chem1"         "ts"
## [4,] "Chem2"         "ts"
## [5,] "Particle size" "ts"
```

As we can now observe that we have successfully converted all the variable in the data into Time series.We will now explore and visualize the data.

Data Exploration and Visualisation

We will now plot the converted Time Series Data for the Mortality time series variables and interpret their important characteristics using the **5 Bullet Points**:

- **Trend**
- **Seasonality**
- **Changing Variance**
- **Behavior**

- **Intervention Point**

Plotting the time series graph of Mortality variable

```
plot(Mort,ylab="Weekely Mortality",xlab="Years",main="Figure-1: Time series plot of Mortality series",type='o',col="red2")
```

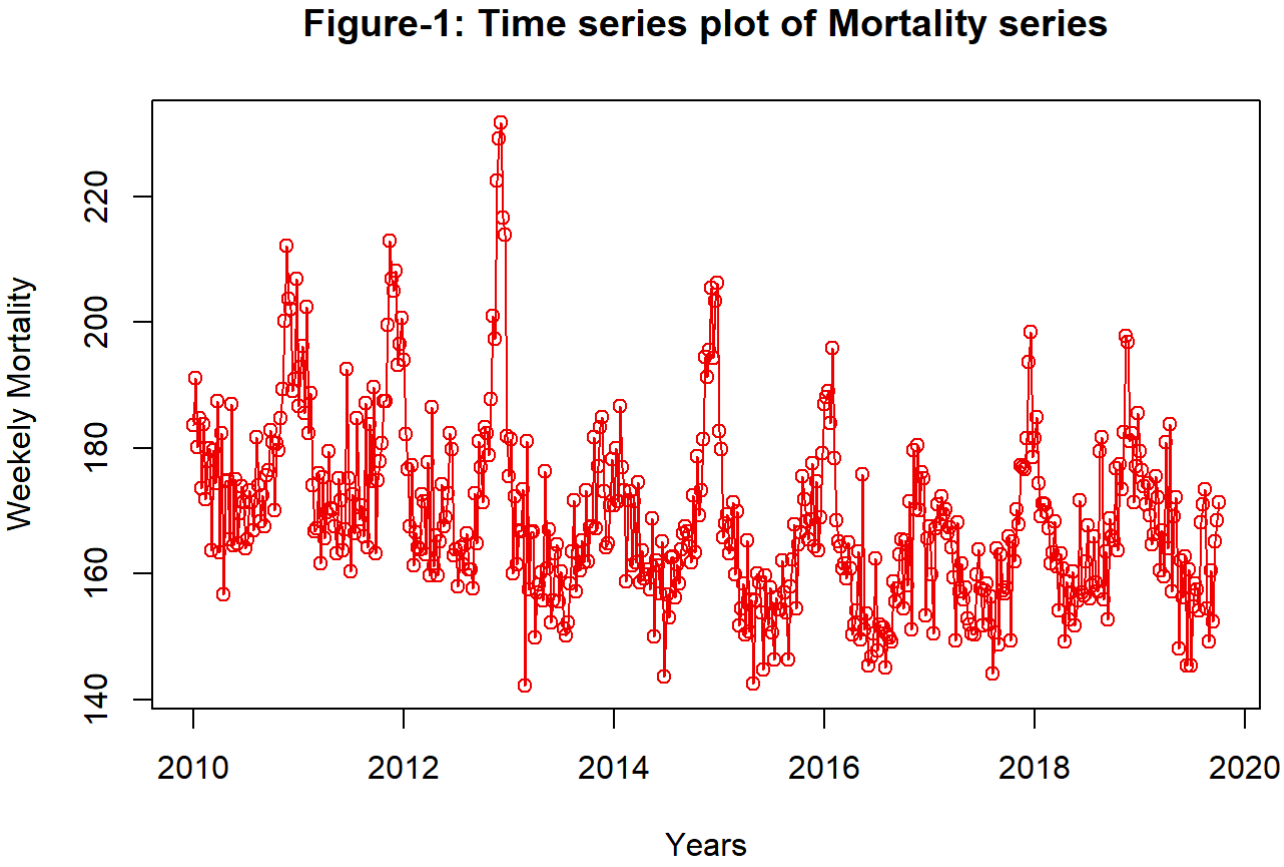


Figure 1

From the above plot(Figure 1), the Time Series plot shows high level of **Seasonality** with successive Auto regressive points and moving average overall from the year 2010 to 2020 respectively.

Checking **5 bullet points**:

- **Trend** - There is nearly slight Downward Trend present in the series.
- **Seasonality** - There is very high level of repeating patterns (seasonality) can be seen in the graph from year 2010-2020
- **Changing Variance** - There is a significant level of changing variance which is also in repeating patterns, can be seen if we look at the years 2012-2014 and 2018-2020.
- **Behavior** - A mixed Auto regressive and Moving average behavior can be seen in the graph.
- **Intervention point** - There is a sudden change point (peaks/drops) in the year 2013 observed in the series.

Plotting the time series graph of Temperature

variable

```
plot(Temp,ylab="Weekely Temperature",xlab="Years",main="Figure-2: Time series plot of Tempera
ture series",type='o',col="darkmagenta")
```

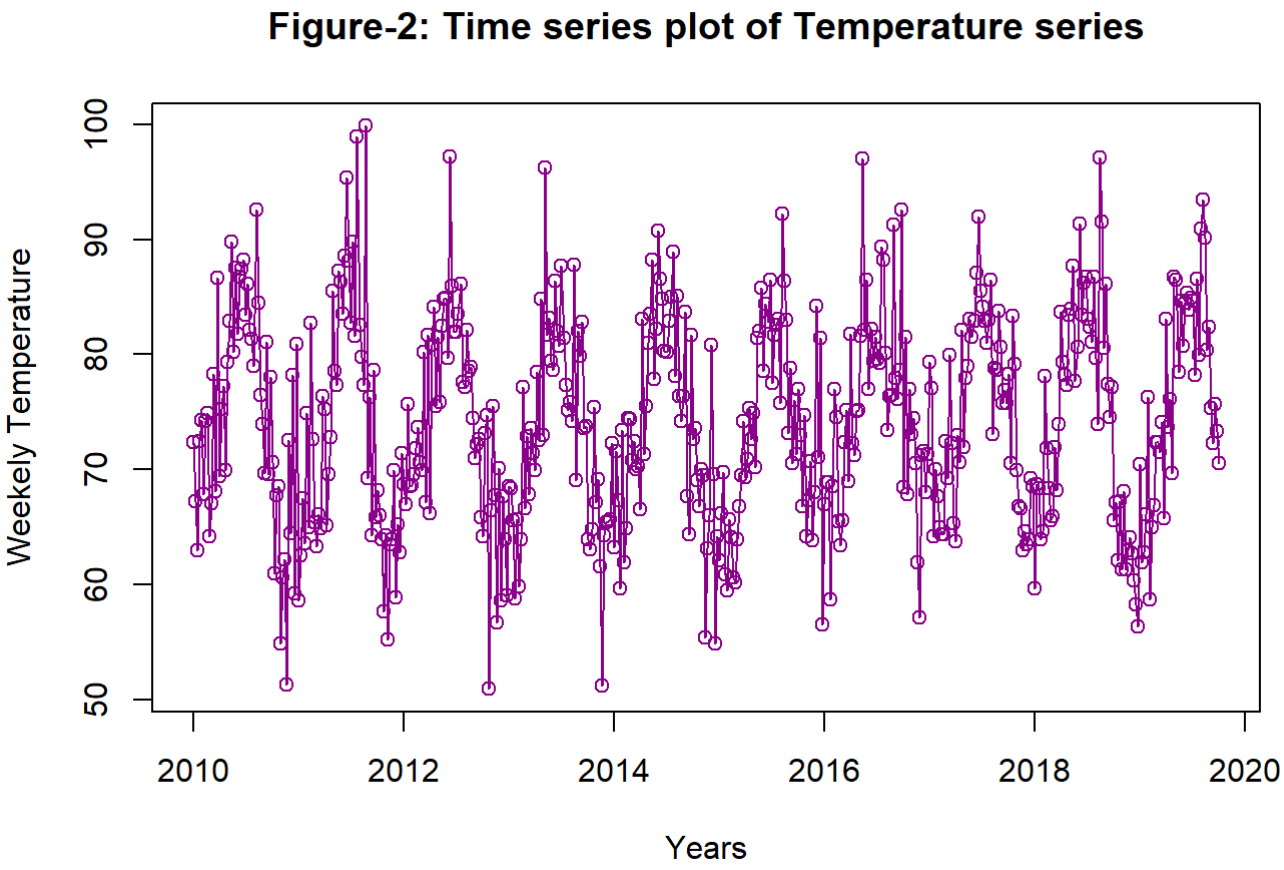


Figure 2

From the above plot(Figure 2), the Time Series plot shows high level of **Seasonality** with successive Auto regressive points and moving average overall from the year 2010 to 2020 respectively.

Checking **5 bullet points**:

- **Trend** - There is nearly slight Upward Trend present in the series.
- **Seasonality** - There is very high level of repeating patterns (seasonality) can be seen in the graph from year 2010-2020
- **Changing Variance** - There is not much changing variance present in the series .
- **Behavior** - A mixed Auto regressive and Moving average behavior can be seen in the graph.
- **Intervention point** - There is not sudden change point observed in the series.

Plotting the time series graph of Chemical-1 variable

```
plot(Chem1,ylab="Weekely Chemical1",xlab="Years",main="Figure-3: Time series plot of Chemical
-1 series",type='o',col="orange")
```

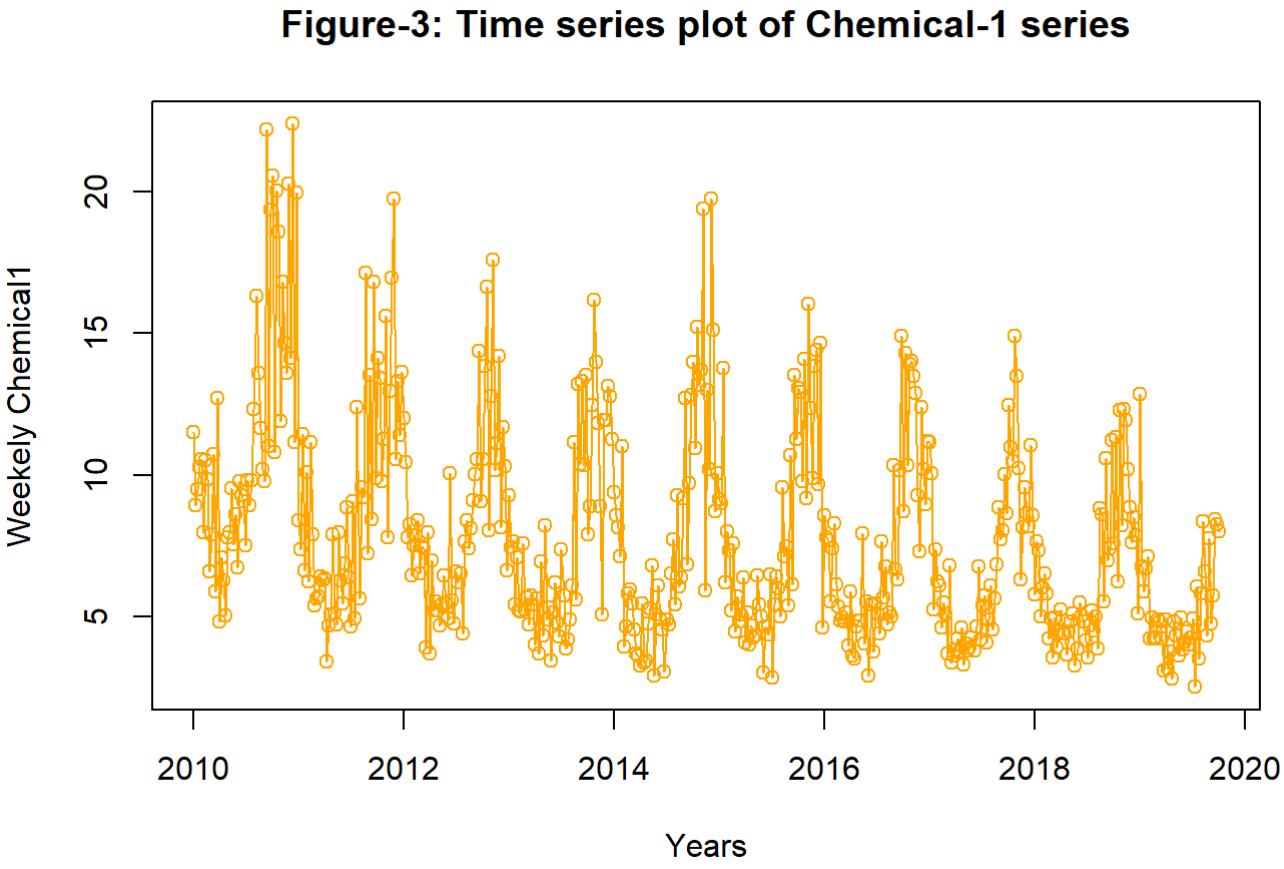


Figure 3

From the above plot(Figure 3), the Time Series plot shows high level of **Seasonality** with successive Auto regressive points and moving average overall from the year 2010 to 2020 respectively.

Checking **5 bullet points**:

- **Trend** - There is a slight Downward Trend present in the series overall.
- **Seasonality** - There is very high level of repeating patterns (seasonality) can be seen in the graph from year 2010-2020
- **Changing Variance** - There is a significant level of changing variance which is also in repeating patterns, can be seen if we look at the years 2010-2011 and 2018-2019.
- **Behavior** - A mixed Auto regressive and Moving average behavior can be seen in the graph.
- **Intervention point** - There is a sudden change point (peaks) in the year 2011 observed in the series.

Plotting the time series graph of Chemical-2 variable

```
plot(Chem2,ylab="Weekely Chemical2",xlab="Years",main="Figure-4: Time series plot of Chemical
-2 series",type='o',col="blue")
```

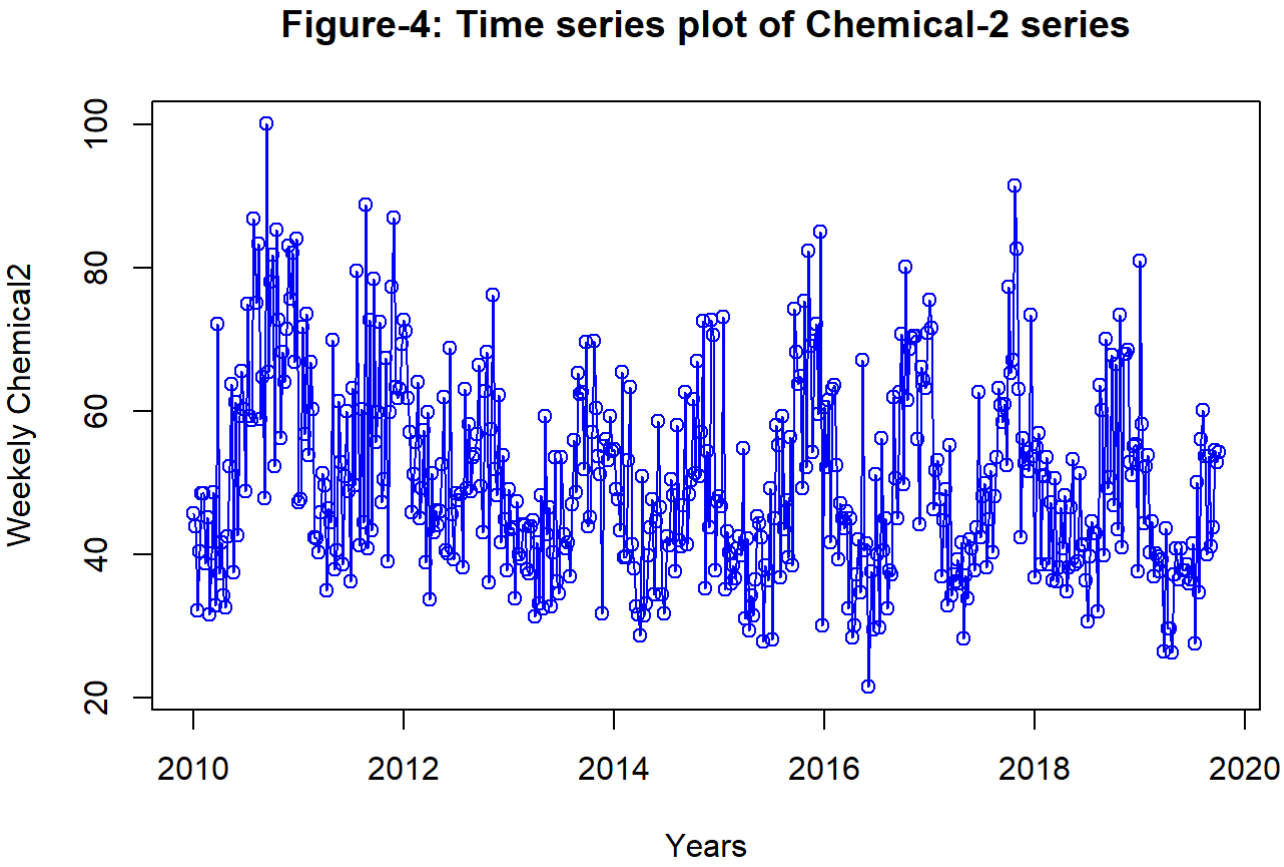


Figure 4

From the above plot(Figure 4), the Time Series plot shows high level of **Seasonality** with successive Auto regressive points and moving average overall from the year 2010 to 2020 respectively.

Checking **5 bullet points**:

- **Trend** - There is nearly No Trend present in the series.
- **Seasonality** - There is very high level of repeating patterns (seasonality) can be seen in the graph from year 2010-2020
- **Changing Variance** - There is a changing variance present in the series if looked the year 2011 and 2019 .
- **Behavior** - A mixed Auto regressive and Moving average behavior can be seen in the graph.
- **Intervention point** - There is not sudden change point observed in the series.

Plotting the time series graph of Particle Size variable

```
plot(Size,ylab="Weekely Particle Size",xlab="Years",main="Figure-5: Time series plot of Parti
cle Size series",type='o',col="darkgreen")
```

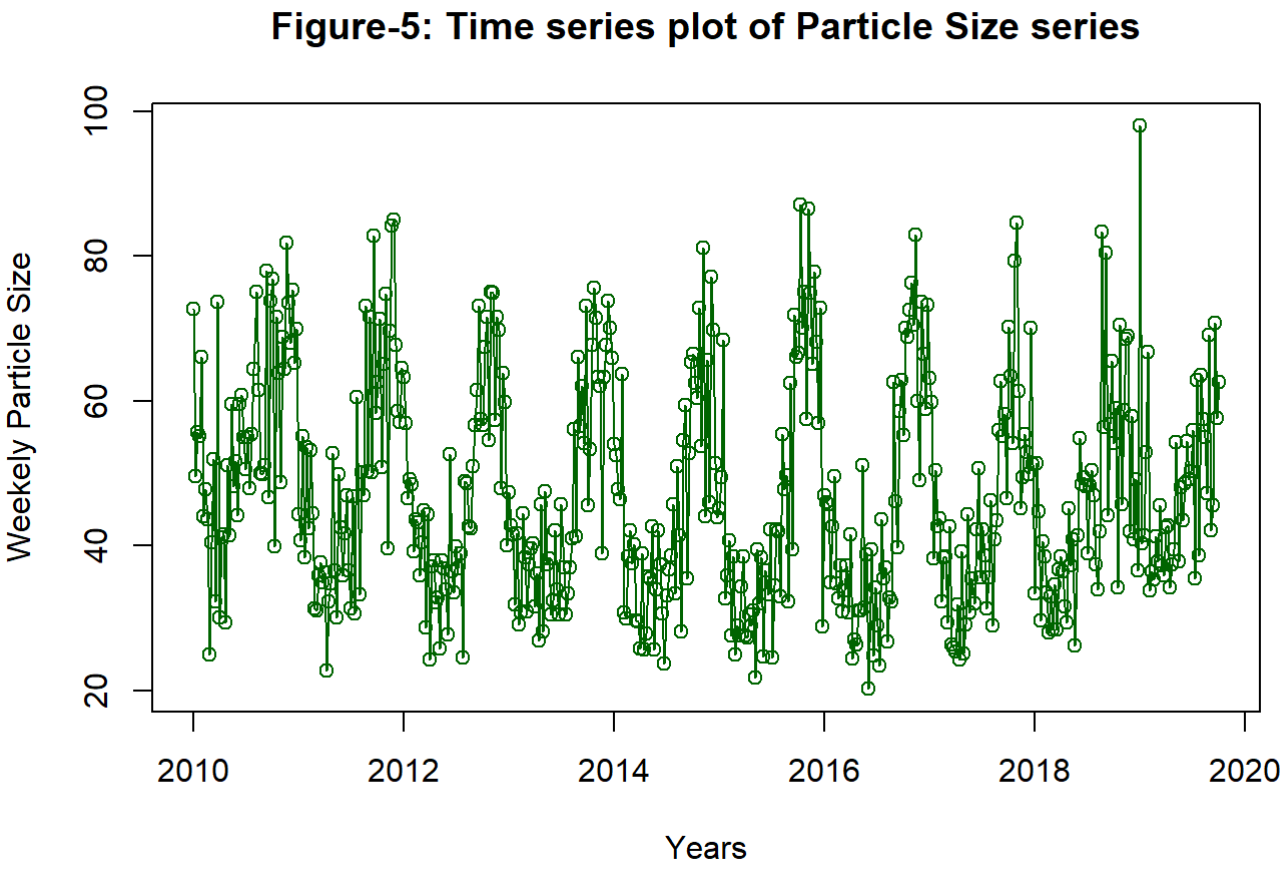


Figure 5

From the above plot(Figure 5), the Time Series plot shows high level of **Seasonality** with successive Auto regressive points and moving average overall from the year 2010 to 2020 respectively.

Checking **5 bullet points**:

- **Trend** - There is nearly slight Upward Trend present in the series overall.
- **Seasonality** - There is very high level of repeating patterns (seasonality) can be seen in the graph from year 2010-2020
- **Changing Variance** - There is a changing variance present in the series .
- **Behavior** - A mixed Auto regressive and Moving average behavior can be seen in the graph.
- **Intervention point** - There is a sudden change point (peak) observed in the series at year 2019.

We will now further explore the relationship between all of the variables by plotting their time series within the same plot. In order to do that,we will perform scaling for the series.

```
Mort_scale = scale(mort)
plot(Mort_scale, plot.type = "s" ,xlab="Years",ylab="Mortality scaled",col = c("red","darkmag
enta", "orange", "blue","darkgreen"),main="Figure-6: Time series plot of Scaled Mortality dat
a-set")
legend("topright",lty=1,cex=0.65, text.width = 2, col=c("red","darkmagenta", "orange", "blu
e","darkgreen"), c("Mortality", "Temperature", "Chemical-1", "Chemical-2","Particle size"))
```

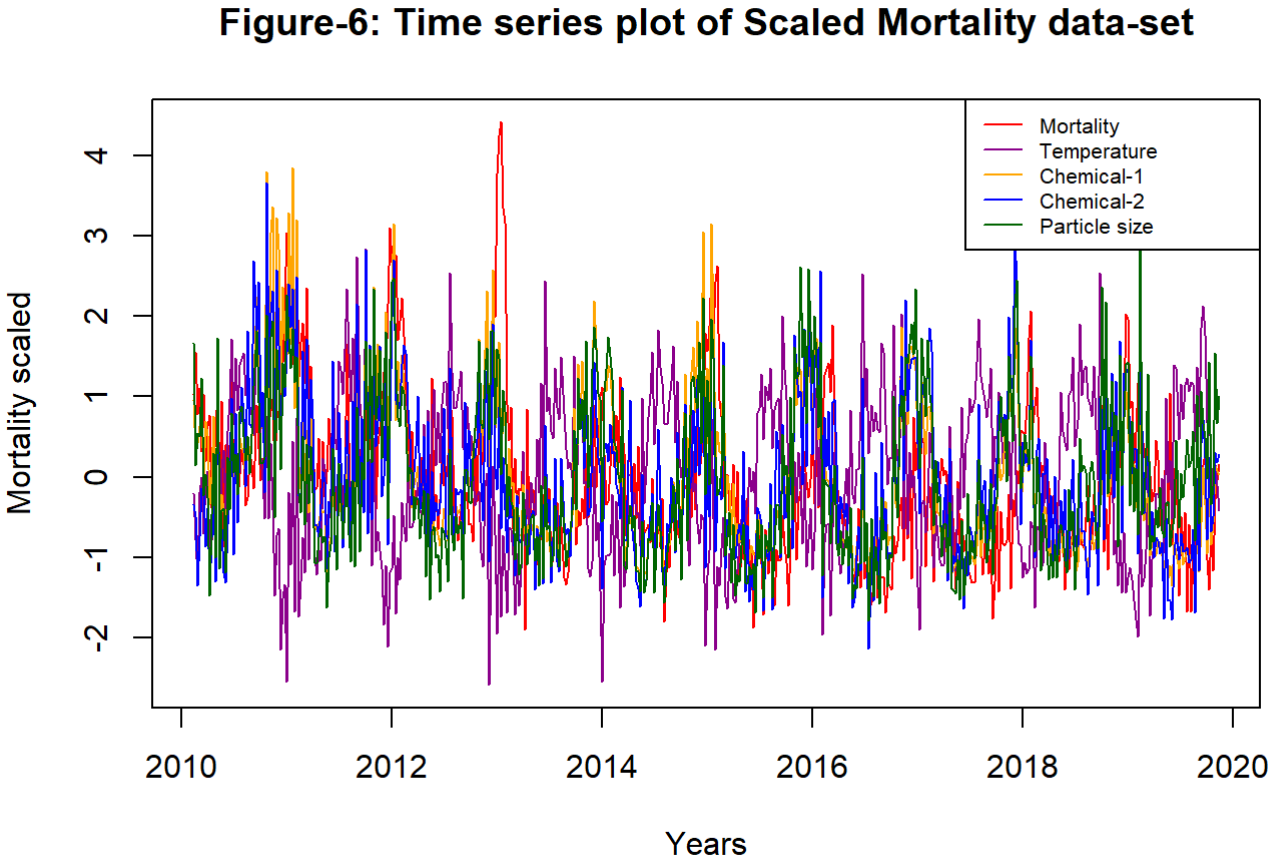


Figure 6

From the above scaled plot, we can observe that all the time series are likely to be very highly correlated with each other due to their seasonality component.To confirm that correlation, we will calculate and identify the correlation between them.

Correlation among the series

```
cor(mort)
```

```
##                mortality        temp       chem1     chem2 particle size
## mortality      1.0000000 -0.36018067  0.55707393 0.4407115    0.47410247
## temp          -0.3601807  1.00000000 -0.09785582 0.1141372   -0.01723095
## chem1          0.5570739 -0.09785582  1.00000000 0.8510448    0.86611747
## chem2          0.4407115  0.11413717  0.85104485 1.0000000    0.80750391
## particle size  0.4741025 -0.01723095  0.86611747 0.8075039    1.00000000
```

From the above matrix of correlation we can observe that, Mortality has a moderate negative correlation with temperature and positive correlations with chem1, chem2, and particle size. Temperature has weak correlations with the other variables. Chem1 has very strong positive correlations with chem2 and particle size. Chem2 and particle size also share a strong positive correlation.

Check for Stationarity

We will now proceed with by checking the stationarity of data with the help of ACF, PACF and ADF test.

ACF and PACF of Mortality series

```
par(mfrow=c(1,2))
acf(Mort,main="ACF of Mortality series")
pacf(Mort,main="PACF of Mortality series")
```

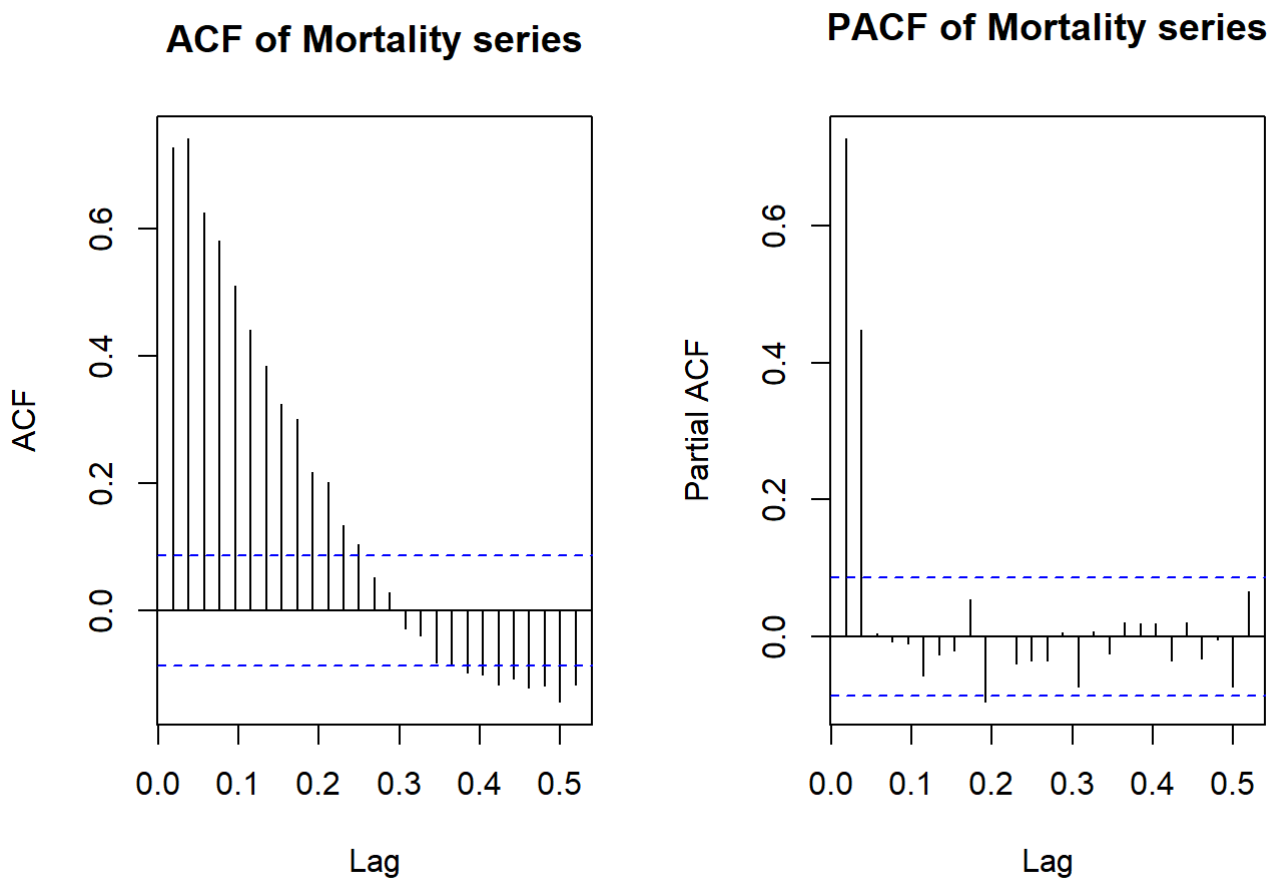


Figure 7

From the above ACF and PACF test, it can be interpreted that there is some trend in the series due to decaying pattern of ACF and there is also some autocorrelation present in the series shown by first significant lag in PACF.

ADF test of Mortality series

$H0$ - series is non-stationary $H1$ - series is stationary

```
adf.test(Mort)
```

```
## Warning in adf.test(Mort): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  Mort
## Dickey-Fuller = -5.4301, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

From the above ADF test we can see that the p-value is less than 5% level of significance , hence we reject the null hypothesis $H0$ and confirm our series stationary.

ACF and PACF of Temperature series

```
par(mfrow=c(1,2))
acf(Temp,main="ACF of Temperature series")
pacf(Temp,main="PACF of Temperature series")
```

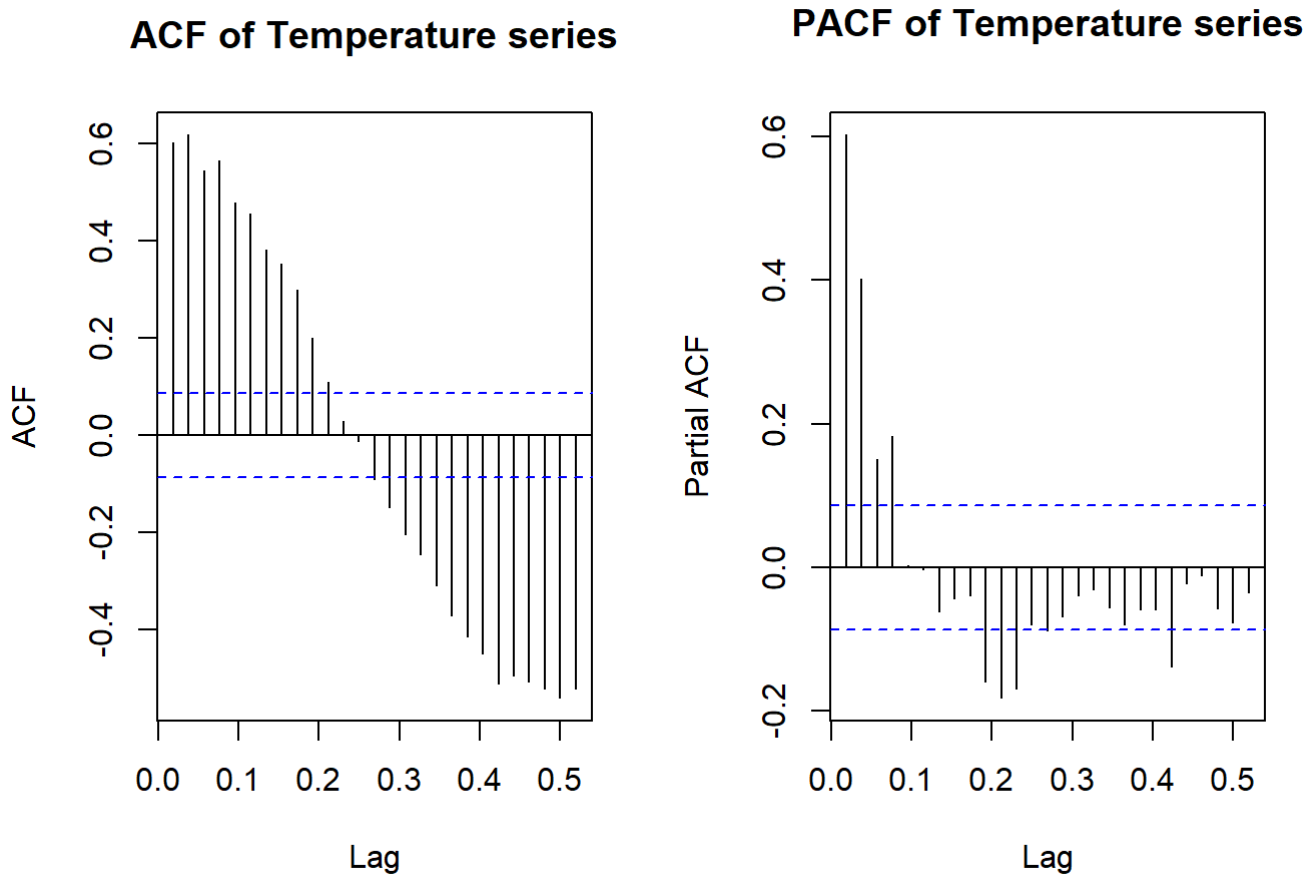


Figure 8

From the above ACF and PACF test, it can be interpreted that there is some trend in the series due to decaying pattern of ACF and there is also some autocorrelation present in the series shown by first significant lag in PACF.

ADF test of Temperature series

$H0$ - series is non-stationary $H1$ - series is stationary

```
adf.test(Temp)
```

```
## Warning in adf.test(Temp): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  Temp
## Dickey-Fuller = -4.4572, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

From the above ADF test we can see that the p-value is less than 5% level of significance , hence we reject the null hypothesis $H0$ and confirm our series stationary.

ACF and PACF of Chemical-1 series

```
par(mfrow=c(1,2))
acf(Chem1,main="ACF of Chemical-1 series")
pacf(Chem1,main="PACF of Chemical-1 series")
```

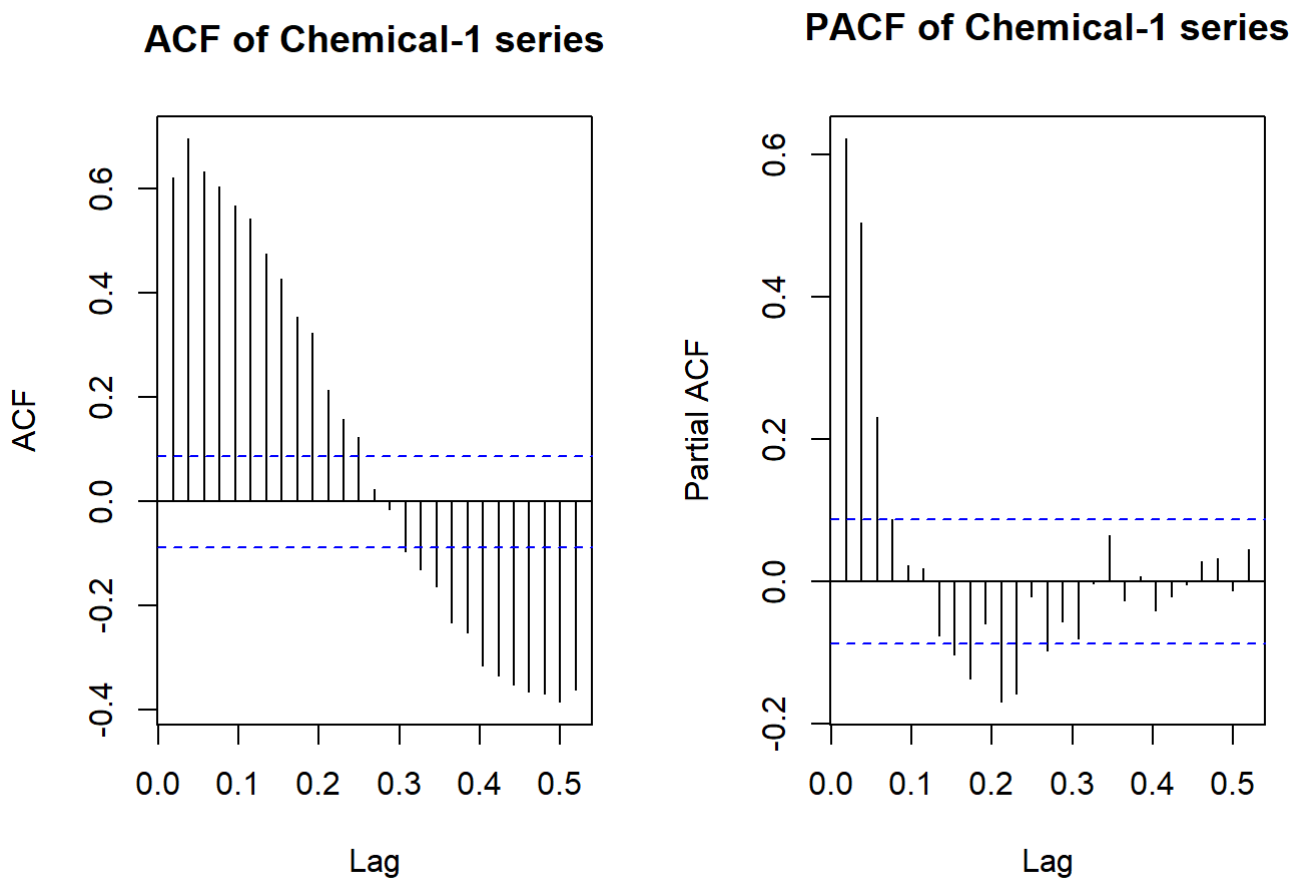


Figure 9

From the above ACF and PACF test, it can be interpreted that there is some trend in the series due to decaying pattern of ACF and there is also some autocorrelation present in the series shown by first significant lag in PACF.

ADF test of Chemical-1 series

$H0$ - series is non-stationary $H1$ - series is stationary

```
adf.test(Chem1)
```

```
## Warning in adf.test(Chem1): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  Chem1
## Dickey-Fuller = -4.4926, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

From the above ADF test we can see that the p-value is less than 5% level of significance , hence we reject the null hypothesis $H0$ and confirm our series stationary.

ACF and PACF of Chemical-2 series

```
par(mfrow=c(1,2))
acf(Chem2,main="ACF of Chemical-2 series")
pacf(Chem2,main="PACF of Chemical-2 series")
```

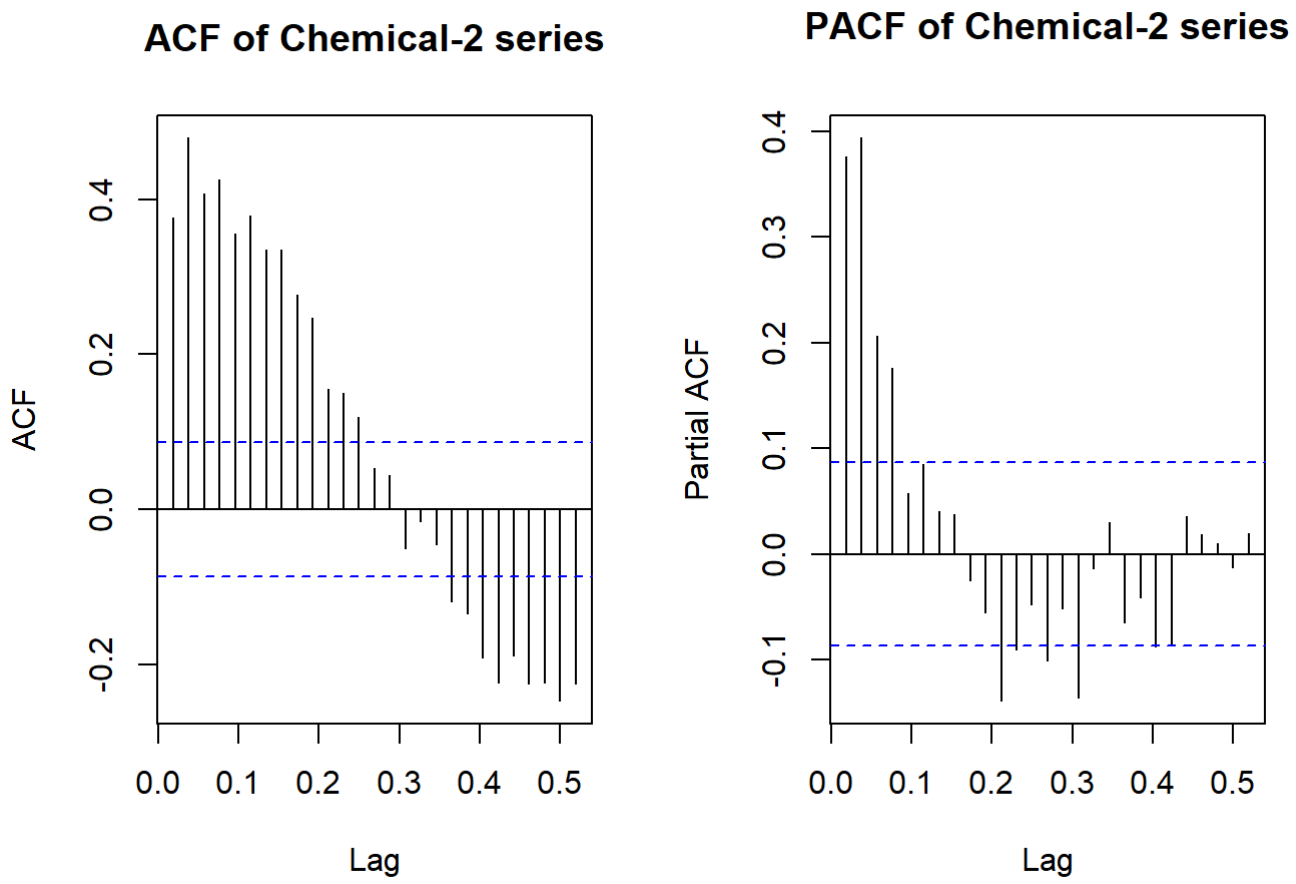


Figure 10

From the above ACF and PACF test, it can be interpreted that there is some trend in the series due to decaying pattern of ACF and there is also some autocorrelation present in the series shown by first significant lag in PACF.

ADF test of Chemical-2 series

$H0$ - series is non-stationary $H1$ - series is stationary

```
adf.test(Chem2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  Chem2
## Dickey-Fuller = -3.9606, Lag order = 7, p-value = 0.01097
## alternative hypothesis: stationary
```

From the above ADF test we can see that the p-value is less than 5% level of significance , hence we reject the null hypothesis $H0$ and confirm our series stationary.

ACF and PACF of Particle Size series

```
par(mfrow=c(1,2))
acf(Size,main="ACF of Particle Size  series")
pacf(Size,main="PACF of Particle Size series")
```

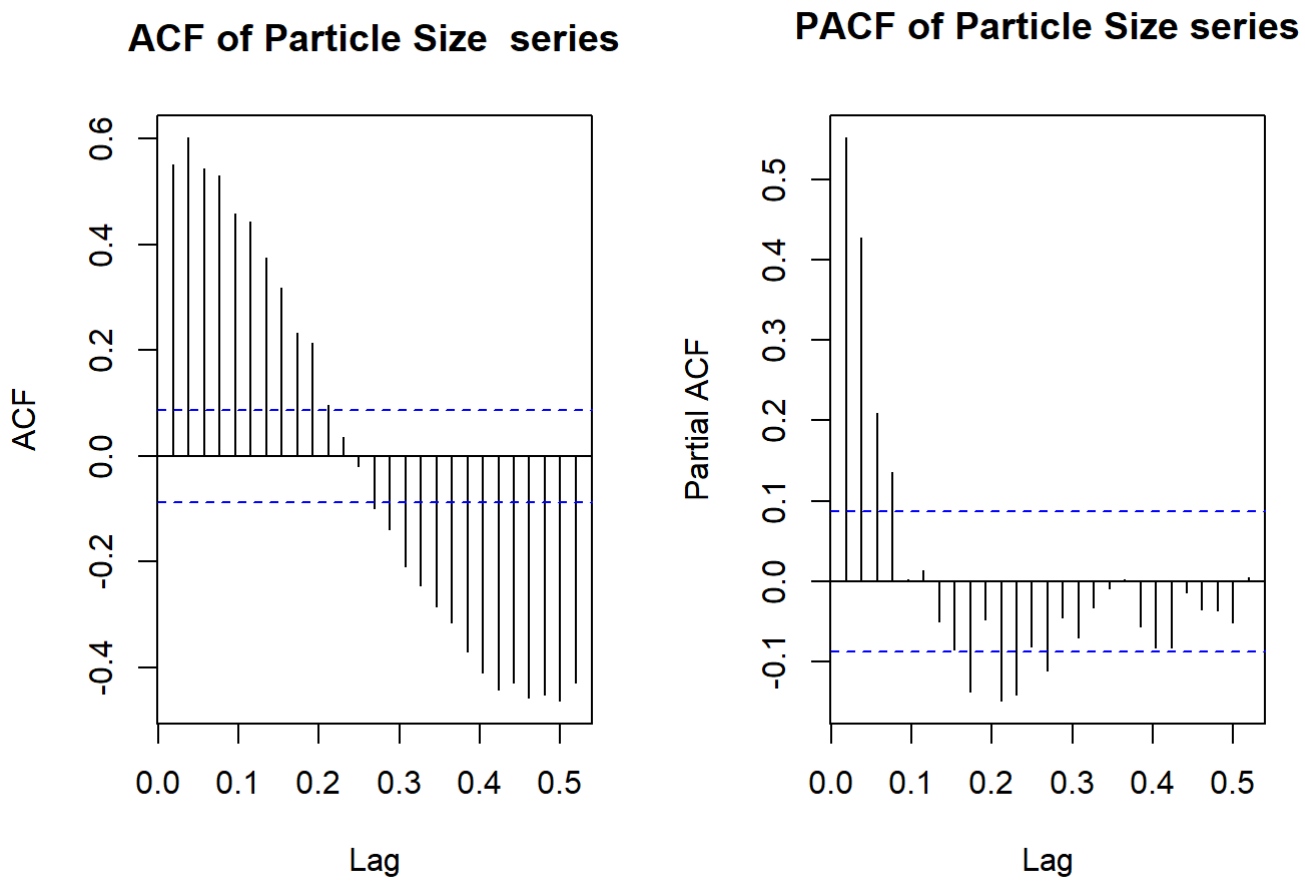


Figure 11

From the above ACF and PACF test, it can be interpreted that there is some trend in the series due to decaying pattern of ACF and there is also some autocorrelation present in the series shown by first significant lag in PACF.

ADF test of Particle Size series

$H0$ - series is non-stationary $H1$ - series is stationary

```
adf.test(Size)
```

```
## Warning in adf.test(Size): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  Size
## Dickey-Fuller = -4.493, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

From the above ADF test we can see that the p-value is less than 5% level of significance , hence we reject the null hypothesis $H0$ and confirm our series stationary.

As we encountered that all the 5 series above are stationary in nature. So we will now proceed with the Decomposition analysis of all the series.

Decomposition of Series

Analyzing the impact of the components of a time series data on the given data set.The use of **STL** decomposition allows for a more thorough investigation of the time series' **seasonal**, **trend**, and **remainder**.

STL Decomposition of Mortality series

```
#Decomposition of Mort series
Mort_stl = stl(Mort, t.window = 15, s.window = "periodic", robust = TRUE)
plot(Mort_stl, main = "Figure 12: STL decomposition of Mortality series", col="red")
```

Figure 12: STL decomposition of Mortality series

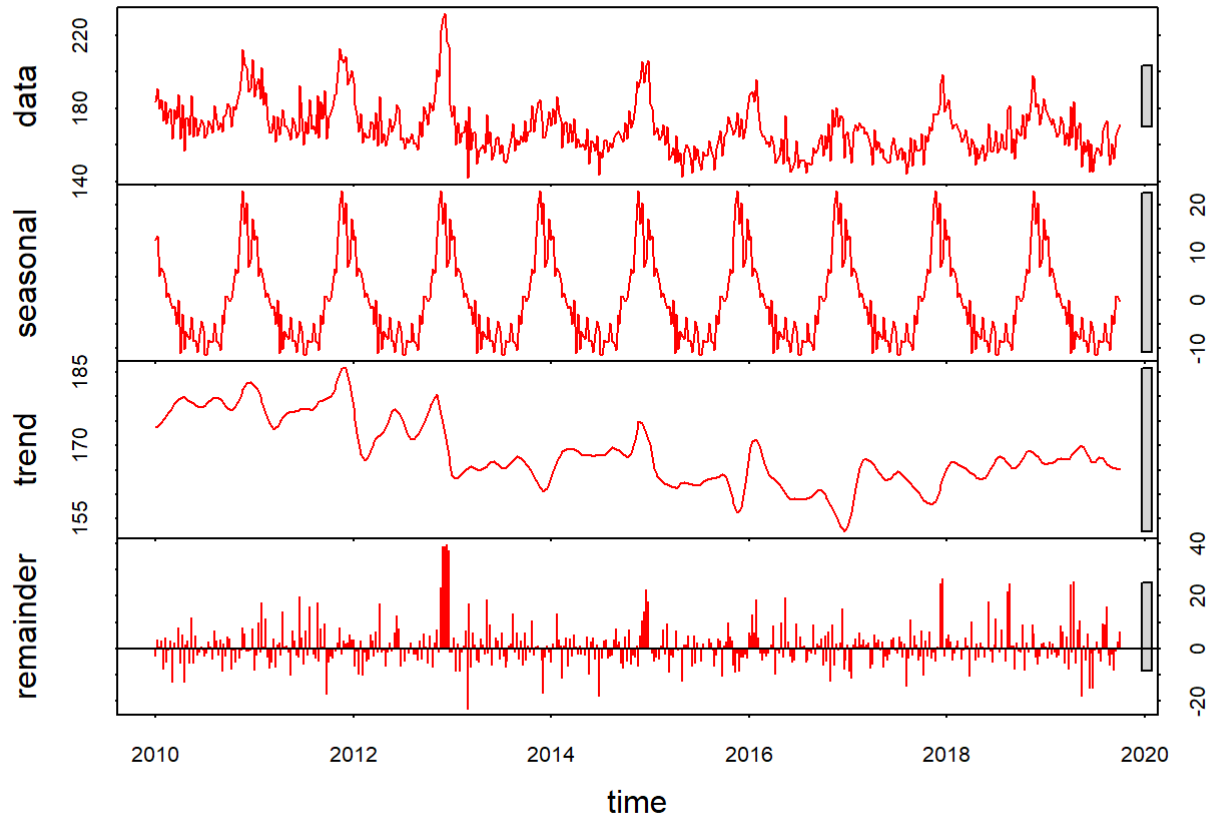


Figure 12

The STL decomposition of the mortality series revealed a trend pattern that mirrors the original series, indicating its absence. The ACF and its corresponding plot indicated seasonality in the series, meaning the observed seasonal pattern is significant. Notable interventions appear in the decomposition's remainder between 2012 & 2014 and during some weeks in 2016.

STL Decomposition of Temperature series

```
#Decomposition of Temp series
Temp_stl = stl(Temp, t.window = 15, s.window = "periodic", robust = TRUE)
plot(Temp_stl, main = "Figure 13: STL decomposition of Temperature series", col="darkmagent
a")
```

Figure 13: STL decomposition of Temperature series

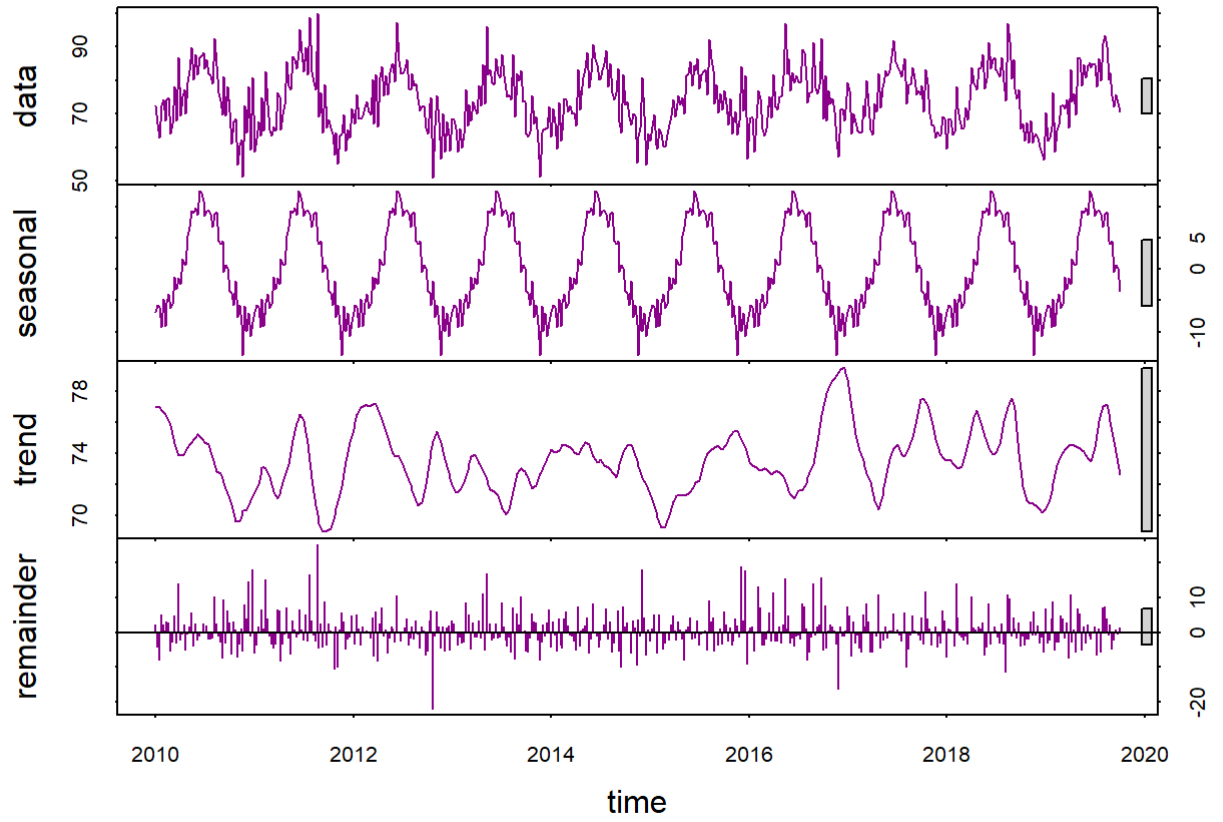


Figure 13

The STL decomposition of the Temperature series indicates that its trend closely resembles the original series, suggesting no distinct trend. The ACF plot revealed clear seasonality in the series, making it essential to consider the seasonal pattern identified. Notable interventions appear in the decomposition's remainder between 2012-2013 and 2016-2017.

STL Decomposition of Chemical-1 series

```
#Decomposition of Temp series
Chem1_stl = stl(Chem1, t.window = 15, s.window = "periodic", robust = TRUE)
plot(Chem1_stl, main = "Figure 14: STL decomposition of Chemical-1 series", col="orange")
```

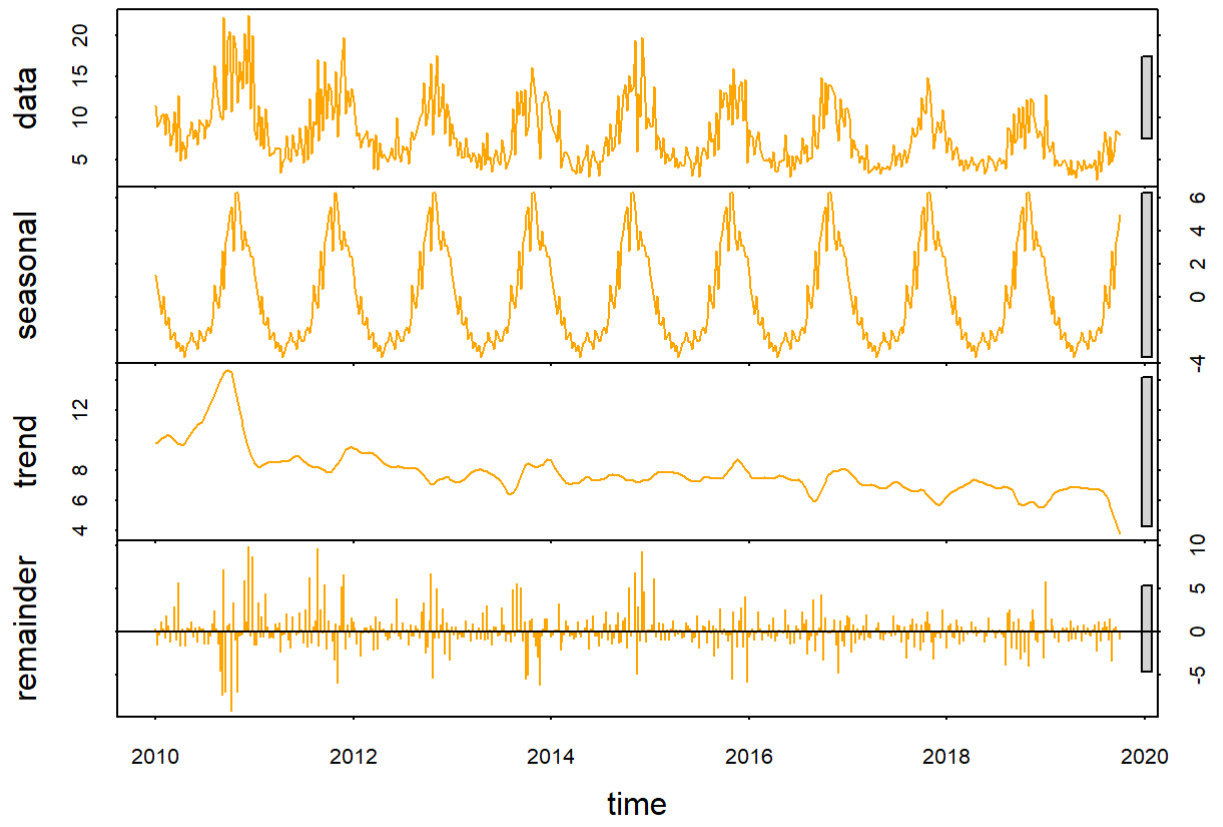


Figure 14

The STL decomposition of the Chemical-1 series reveals a trend pattern that mirrors the original series, with a noticeable decline. The ACF plot indicates clear seasonality, which is reflected in the decomposition as well. The remainder component of the decomposition displays multiple significant interventions.

STL Decomposition of Chemical-2 series

```
#Decomposition of Temp series
Chem2_stl = stl(Chem2, t.window = 15, s.window = "periodic", robust = TRUE)
plot(Chem2_stl, main = "Figure 15: STL decomposition of Chemical-2 series", col="blue")
```

Figure 15: STL decomposition of Chemical-2 series

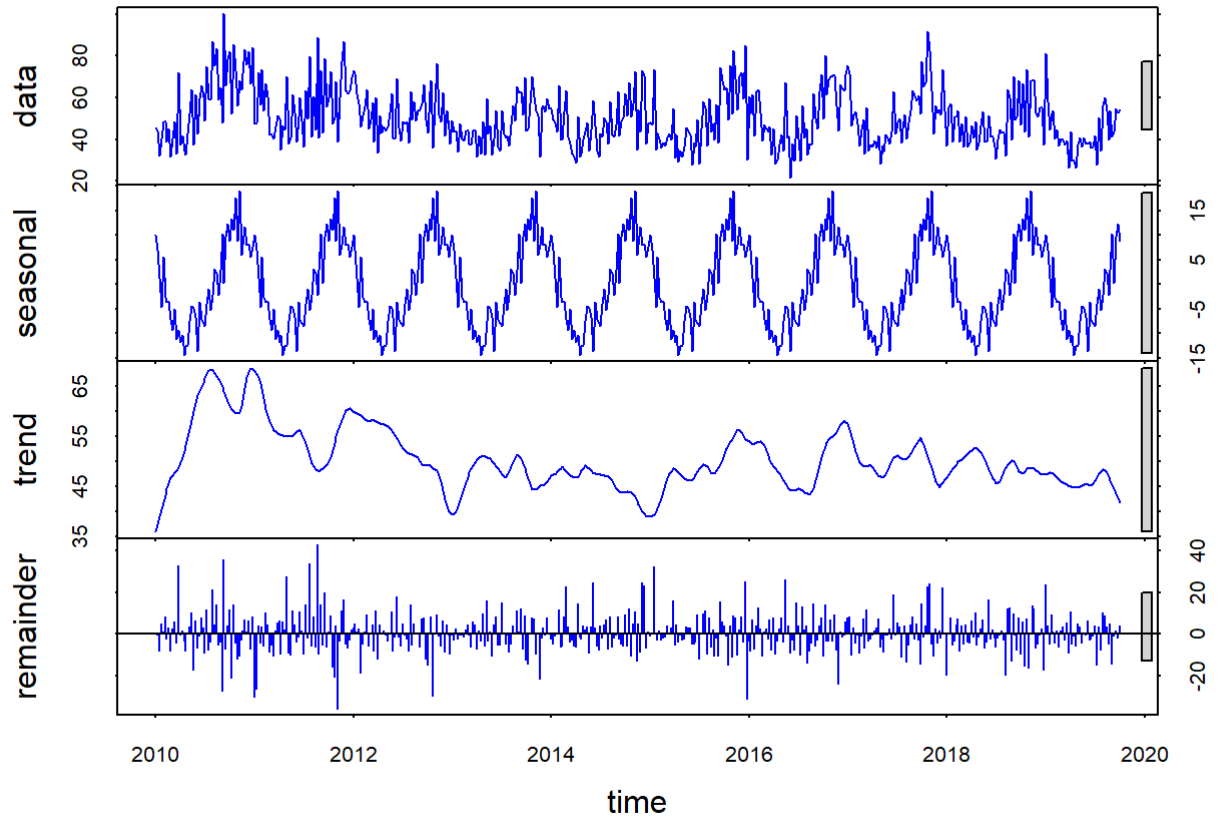


Figure 15

The STL decomposition of the chem2 series reveals a trend that mirrors the initial series, with a noticeable downward progression. The ACF plot for this series indicates clear seasonality, which is also evident in the decomposition. The residual component of the decomposition displays multiple significant interventions.

STL Decomposition of Particle size series

```
#Decomposition of Temp series
Size_stl = stl(Size, t.window = 15, s.window = "periodic", robust = TRUE)
plot(Size_stl, main = "Figure 16: STL decomposition of Particle size series", col="darkgree
n")
```

Figure 16: STL decomposition of Particle size series

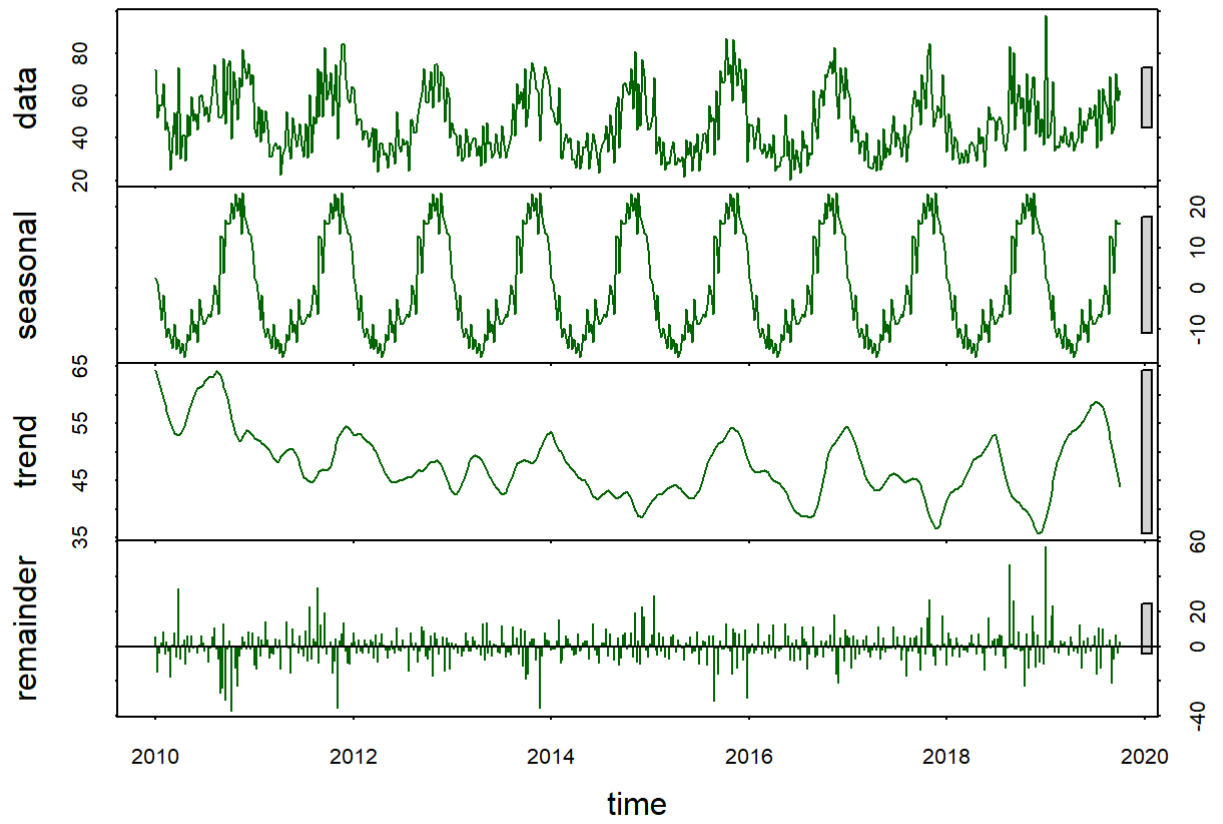


Figure 16

Based on the STL decomposition of the Particle size series, its trend is consistent with the original series, with no distinct trend evident. The ACF plot for the series indicates clear seasonality, which is mirrored in the decomposition. After 2018, notable interventions are visible in the residual component of the decomposition.

Time Series Regression Methods

We will try fitting distributed lag models, which incorporate an independent explanatory series and its lags to assist explain the general variance and correlation structure in our dependent series, in an effort to identify a good model for predicting Mortality data-set overall.

To determine the model's finite lag length, we create a procedure that computes measurements of accuracy like AIC/BIC and MASE for models with various lag lengths, then select the model with the lowest values.

Distributed Lag Model

The Distributed Lag Model from the Regression models describes that the effect of an independent variable on the dependent variables occurs over the time. Therefore we require to build distributed lag models to reduce the multi-collinearity and dependency for each variable. Some of the most important used methods are:

- **Finite Distributed Lag Model**
- **Polynomial Distributed Lag Model**
- **Koyck Distributed Lag Model**
- **Autoregressive Distributed Lag Model**

```
# Pre-defined function for sort.score

sort.score <- function(x, score = c("bic", "aic")){
  if (score == "aic"){
    x[with(x, order(AIC)),]
  } else if (score == "bic") {
    x[with(x, order(BIC)),]
  }
   else if (score == "mase") {
    x[with(x, order(MASE)),]
   }
  else {
    warning('score = "x" only accepts valid arguments ("aic","bic","mase")')
  }
}
```

Finite Distributed Lag model

In order to find out the best model with the help of Finite distributed lag model, we will consider all the 5 variable series and compute them together in all possible combinations to find the best model out of it.

We will also find the accurate lag-length of the model with the help of **AIC**, **BIC** and **MASE**.

Models based on AIC

```
finiteDLMauto(x=as.vector(Temp+Chem1+Chem2+Size), y= as.vector(Mort), q.min = 1, q.max = 10,
model.type="dlm", error.type = "AIC",trace=TRUE)
```

```
##    q - k    MASE      AIC      BIC   GMRAE   MBRAE R.Adj.Sq Ljung-Box
## 10    10 0.95618 3808.640 3863.378 0.91034 0.08080  0.40930         0
## 9      9 0.96472 3824.031 3874.582 0.91224 0.48915  0.39781         0
## 8      8 0.98748 3852.760 3899.121 0.89770 0.49823  0.37015         0
## 7      7 1.04097 3884.534 3926.700 1.01979 0.50372  0.33704         0
## 6      6 1.08280 3928.067 3966.035 1.06045 0.74233  0.28527         0
## 5      5 1.11104 3953.877 3987.641 1.10525 0.26136  0.25792         0
## 4      4 1.12516 3974.644 4004.202 1.13114 1.21069  0.23579         0
## 3      3 1.17269 4014.211 4039.558 1.14260 1.04393  0.18507         0
## 2      2 1.18186 4028.943 4050.076 1.13933 0.17544  0.17181         0
## 1      1 1.20116 4053.877 4070.791 1.12165 0.63659  0.14432         0
```

Models based on BIC

```
finiteDLMauto(x=as.vector(Temp+Chem1+Chem2+Size), y= as.vector(Mort), q.min = 1, q.max = 10,
model.type="dlm", error.type = "BIC",trace=TRUE)
```

```
##    q - k    MASE      AIC      BIC   GMRAE   MBRAE R.Adj.Sq Ljung-Box
## 10    10 0.95618 3808.640 3863.378 0.91034 0.08080  0.40930         0
## 9      9 0.96472 3824.031 3874.582 0.91224 0.48915  0.39781         0
## 8      8 0.98748 3852.760 3899.121 0.89770 0.49823  0.37015         0
## 7      7 1.04097 3884.534 3926.700 1.01979 0.50372  0.33704         0
## 6      6 1.08280 3928.067 3966.035 1.06045 0.74233  0.28527         0
## 5      5 1.11104 3953.877 3987.641 1.10525 0.26136  0.25792         0
## 4      4 1.12516 3974.644 4004.202 1.13114 1.21069  0.23579         0
## 3      3 1.17269 4014.211 4039.558 1.14260 1.04393  0.18507         0
## 2      2 1.18186 4028.943 4050.076 1.13933 0.17544  0.17181         0
## 1      1 1.20116 4053.877 4070.791 1.12165 0.63659  0.14432         0
```

Models based on MASE

```
finiteDLMauto(x=as.vector(Temp+Chem1+Chem2+Size), y= as.vector(Mort), q.min = 1, q.max = 10,
model.type="dlm", error.type = "MASE",trace=TRUE)
```

```
##    q - k    MASE      AIC      BIC   GMRAE   MBRAE R.Adj.Sq Ljung-Box
## 10    10 0.95618 3808.640 3863.378 0.91034 0.08080  0.40930         0
## 9      9 0.96472 3824.031 3874.582 0.91224 0.48915  0.39781         0
## 8      8 0.98748 3852.760 3899.121 0.89770 0.49823  0.37015         0
## 7      7 1.04097 3884.534 3926.700 1.01979 0.50372  0.33704         0
## 6      6 1.08280 3928.067 3966.035 1.06045 0.74233  0.28527         0
## 5      5 1.11104 3953.877 3987.641 1.10525 0.26136  0.25792         0
## 4      4 1.12516 3974.644 4004.202 1.13114 1.21069  0.23579         0
## 3      3 1.17269 4014.211 4039.558 1.14260 1.04393  0.18507         0
## 2      2 1.18186 4028.943 4050.076 1.13933 0.17544  0.17181         0
## 1      1 1.20116 4053.877 4070.791 1.12165 0.63659  0.14432         0
```

From the above results based on AIC, BIC and MASE we can consider the best lag-length to be 10.We will now fit different possible combinations of the model by keeping **“Mort”** as y and rest as predictor.

Distributed Model Fitting with all possible combinations

```
dlm_model_temp = dlm(x=as.vector(Temp), y=as.vector(Mort), q=10)
dlm_model_chem1 = dlm(x=as.vector(Chem1), y=as.vector(Mort), q=10)
dlm_model_chem2 = dlm(x=as.vector(Chem2), y=as.vector(Mort), q=10)
dlm_model_size = dlm(x=as.vector(Size), y=as.vector(Mort), q=10)
dlm_model_temp.chem1 = dlm(x=as.vector(Temp+Chem1), y=as.vector(Mort), q=10)
dlm_model_temp.chem2 = dlm(x=as.vector(Temp+Chem2), y=as.vector(Mort), q=10)
dlm_model_temp.size = dlm(x=as.vector(Temp+Size), y=as.vector(Mort), q=10)
dlm_model_chem1.chem2 = dlm(x=as.vector(Chem1+Chem2), y=as.vector(Mort), q=10)
dlm_model_chem1.size = dlm(x=as.vector(Chem1+Size), y=as.vector(Mort), q=10)
dlm_model_chem2.size = dlm(x=as.vector(Chem2+Size), y=as.vector(Mort), q=10)
dlm_model_temp.chem1.size = dlm(x=as.vector(Temp+Chem1+Size), y=as.vector(Mort), q=10)
dlm_model_temp.chem2.size = dlm(x=as.vector(Temp+Chem2+Size), y=as.vector(Mort), q=10)
dlm_model_chem1.chem2.size = dlm(x=as.vector(Temp+Chem1+Size), y=as.vector(Mort), q=10)
dlm_model_temp.chem1.chem2.size = dlm(x=as.vector(Temp+Chem1+Chem2+Size), y=as.vector(Mort),
q=10)
```

Comparing Sort Score based on AIC

```
sort.score(AIC(dlm_model_temp$model,dlm_model_chem1$model,dlm_model_chem2$model,dlm_model_siz
e$model,dlm_model_temp.chem1$model,dlm_model_temp.chem2$model,dlm_model_temp.size$model,dlm_m
odel_chem1.chem2$model,dlm_model_chem1.size$model,dlm_model_chem2.size$model,dlm_model_temp.c
hem1.size$model,dlm_model_temp.chem2.size$model,dlm_model_chem1.chem2.size$model,dlm_model_te
mp.chem1.chem2.size$model), score = "aic")
```

```
##                                       df      AIC
## dlm_model_chem1$model                 13 3694.700
## dlm_model_chem1.size$model            13 3735.550
## dlm_model_chem2.size$model            13 3756.135
## dlm_model_size$model                  13 3768.114
## dlm_model_chem1.chem2$model           13 3776.923
## dlm_model_temp.chem1.chem2.size$model 13 3808.640
## dlm_model_chem2$model                 13 3820.629
## dlm_model_temp.chem2.size$model       13 3838.046
## dlm_model_temp.chem1.size$model       13 3856.550
## dlm_model_chem1.chem2.size$model      13 3856.550
## dlm_model_temp$model                  13 3880.743
## dlm_model_temp.size$model             13 3914.604
## dlm_model_temp.chem2$model            13 3975.103
## dlm_model_temp.chem1$model            13 3977.181
```

From the above out put we can say that the model “dlm_model_chem1” is the best model based on AIC scores.

Comparing Sort Score based on BIC

```
sort.score(BIC(dlm_model_temp$model,dlm_model_chem1$model,dlm_model_chem2$model,dlm_model_siz
e$model,dlm_model_temp.chem1$model,dlm_model_temp.chem2$model,dlm_model_temp.size$model,dlm_m
odel_chem1.chem2$model,dlm_model_chem1.size$model,dlm_model_chem2.size$model,dlm_model_temp.c
hem1.size$model,dlm_model_temp.chem2.size$model,dlm_model_chem1.chem2.size$model,dlm_model_te
mp.chem1.chem2.size$model), score = "bic")
```

```
##                                       df      BIC
## dlm_model_chem1$model                 13 3749.438
## dlm_model_chem1.size$model            13 3790.288
## dlm_model_chem2.size$model            13 3810.873
## dlm_model_size$model                  13 3822.852
## dlm_model_chem1.chem2$model           13 3831.661
## dlm_model_temp.chem1.chem2.size$model 13 3863.378
## dlm_model_chem2$model                 13 3875.367
## dlm_model_temp.chem2.size$model       13 3892.783
## dlm_model_temp.chem1.size$model       13 3911.288
## dlm_model_chem1.chem2.size$model      13 3911.288
## dlm_model_temp$model                  13 3935.481
## dlm_model_temp.size$model             13 3969.342
## dlm_model_temp.chem2$model            13 4029.841
## dlm_model_temp.chem1$model            13 4031.919
```

From the above output, we again got the same model “dlm_model_chem1” as our best model with respect to the BIC scores.

Comparing Sort Score based on BIC

```
sort.score(MASE(dlm_model_temp$model,dlm_model_chem1$model,dlm_model_chem2$model,dlm_model_si
ze$model,dlm_model_temp.chem1$model,dlm_model_temp.chem2$model,dlm_model_temp.size$model,dlm_
model_chem1.chem2$model,dlm_model_chem1.size$model,dlm_model_chem2.size$model,dlm_model_temp.
chem1.size$model,dlm_model_temp.chem2.size$model,dlm_model_chem1.chem2.size$model,dlm_model_t
emp.chem1.chem2.size$model), score = "mase")
```

```
##                                         n      MASE
## dlm_model_chem1$model                 498 0.8570098
## dlm_model_chem1.size$model            498 0.8825999
## dlm_model_chem2.size$model            498 0.8939644
## dlm_model_chem1.chem2$model           498 0.9075805
## dlm_model_size$model                  498 0.9138630
## dlm_model_chem2$model                 498 0.9497703
## dlm_model_temp.chem1.chem2.size$model 498 0.9561827
## dlm_model_temp.chem2.size$model       498 0.9860995
## dlm_model_temp.chem1.size$model       498 1.0132430
## dlm_model_chem1.chem2.size$model      498 1.0132430
## dlm_model_temp.size$model             498 1.0712643
## dlm_model_temp$model                  498 1.0725463
## dlm_model_temp.chem2$model            498 1.1413249
## dlm_model_temp.chem1$model            498 1.1688053
```

From the above outputs, again the model “dlm_model_chem1” is considered to be the best with respect to the MASE scores.

Analysisng the model “dlm_model_chem1”

We will now analyse and summarize the best model from the finite distributed lag model based on the AIC , BIC and MASE scores.

```
finite_best = dlm(x=as.vector(Chem1), y=as.vector(Mort), q=10)
summary(finite_best)
```

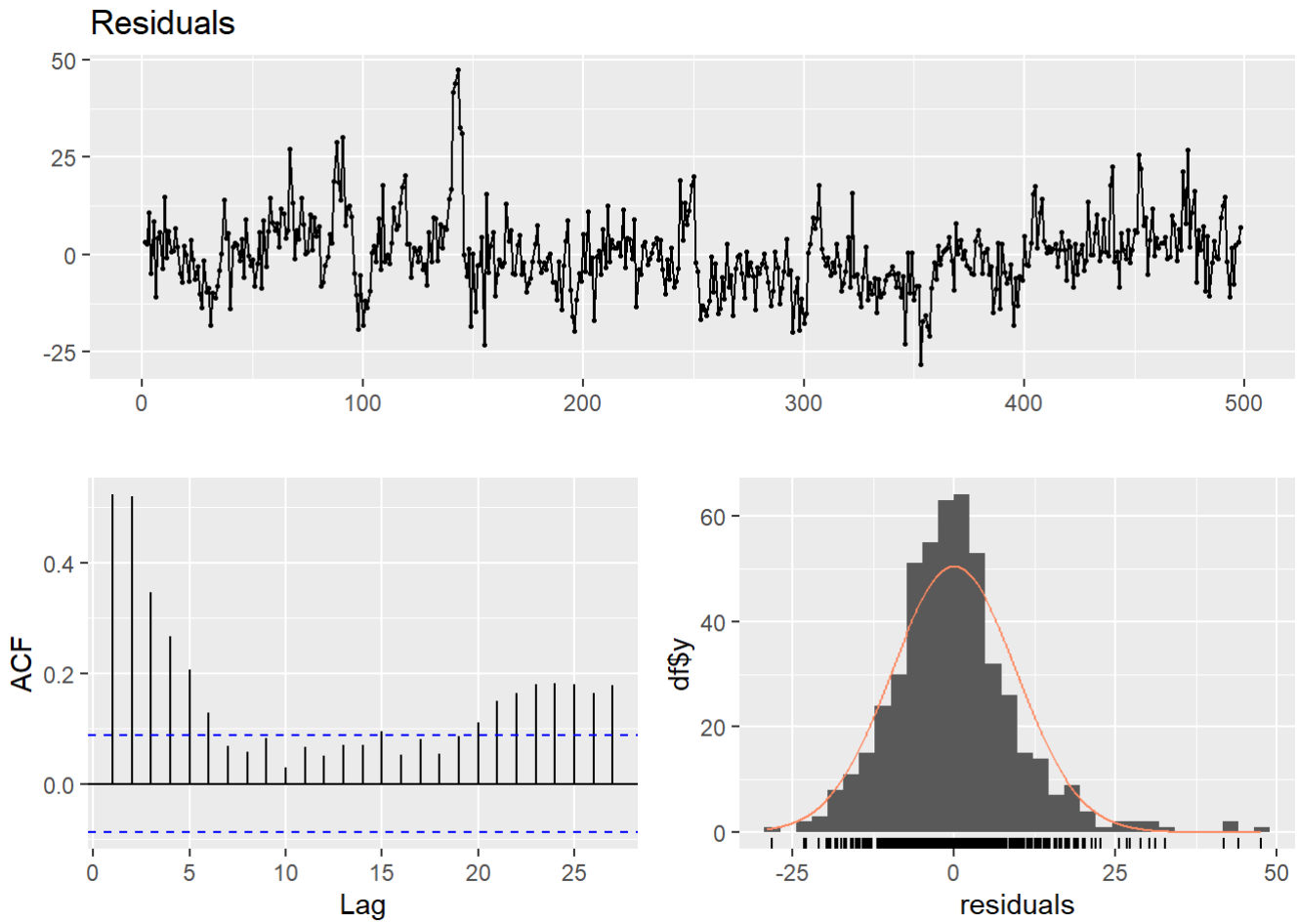
```
##
## Call:
## lm(formula = model.formula, data = design)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.258  -5.462  -0.547   4.347  47.536
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 141.94242    1.27083 111.693  < 2e-16 ***
## x.t           0.86498    0.17997   4.806 2.05e-06 ***
## x.1          -0.26184    0.18135  -1.444 0.149430
## x.2           0.15725    0.19242   0.817 0.414221
## x.3          -0.01975    0.19617  -0.101 0.919859
## x.4           0.56747    0.19730   2.876 0.004203 **
## x.5           0.11239    0.19725   0.570 0.569093
## x.6           0.30151    0.19747   1.527 0.127448
## x.7           0.70386    0.19642   3.583 0.000373 ***
## x.8           0.47447    0.19262   2.463 0.014112 *
## x.9           0.45753    0.18159   2.520 0.012067 *
## x.10          0.04697    0.18013   0.261 0.794365
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.744 on 486 degrees of freedom
## Multiple R-squared:  0.5405, Adjusted R-squared:  0.5301
## F-statistic: 51.97 on 11 and 486 DF,  p-value: < 2.2e-16
##
## AIC and BIC values for the model:
##      AIC      BIC
## 1 3694.7 3749.438
```

From the outputs of **finite_best** model and its summary :

- The **AIC** and **BIC** scores of the model have been reported 3694.7 and 3749.43.
- The adjusted R-squared value reported was 0.53 which is not bad.
- Few of the coefficients are significant.
- The F-statistic is reported at 51.97 on 11 and 486 Degrees of Freedom.

Residual Analysis of the model “dlm_model_chem1”

```
checkresiduals(dlm_model_chem1$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 15
##
## data:  Residuals
## LM test = 185.97, df = 15, p-value < 2.2e-16
```

From the outputs of Model “dlm_model_chem1” residuals :

- The p-value from Breusch-Godfrey test was less than significance level of 0.05.
- The ACF plot reveals that there is seasonality and correlation exist in residual due to significant lags present in the ACF.
- The time-series plot of residual seems to follow a trend overall.

So, we can conclude that the model “dlm_model_chem1” is a decent model with a appropriate composition of residuals.

Polynomial Distributed Lag Model

We will now proceed with fitting a polynomial model with the help of “finiteDLMauto” function and select the best lag length before choosing the best model with respect to AIC, BIC and MASE.

Models based on AIC

```
finiteDLMauto(x = as.vector(Temp+Chem1+Chem2+Size), y = as.vector(Mort), q.min = 1, q.max = 1
0, k.order = 2,
              model.type = "poly", error.type ="AIC", trace = TRUE)
```

```
##     q - k    MASE      AIC      BIC   GMRAE    MBRAE R.Adj.Sq Ljung-Box
## 10 10 - 2 1.00370 3824.015 3845.068 0.94826 -0.32411  0.38108         0
## 9   9 - 2 1.00965 3837.622 3858.685 0.95312 -0.18358  0.37257         0
## 8   8 - 2 1.01991 3862.536 3883.609 0.94784  0.32880  0.35007         0
## 7   7 - 2 1.06717 3893.555 3914.638 1.02471 -2.48798  0.31832         0
## 6   6 - 2 1.10132 3935.758 3956.851 1.05592  0.23084  0.26850         0
## 5   5 - 2 1.13173 3964.708 3985.811 1.10060  0.44803  0.23728         0
## 4   4 - 2 1.12724 3975.765 3996.878 1.06786  0.61708  0.23108         0
## 3   3 - 2 1.17975 4018.930 4040.052 1.14521  5.48844  0.17581         0
## 2   2 - 2 1.18186 4028.943 4050.076 1.13933  0.17544  0.17181         0
## 1   1 - 2 1.20116 4053.877 4070.791 1.12165  0.63659  0.14432         0
```

Models based on BIC

```
finiteDLMauto(x = as.vector(Temp+Chem1+Chem2+Size), y = as.vector(Mort), q.min = 1, q.max = 1
0, k.order = 2,
              model.type = "poly", error.type ="BIC", trace = TRUE)
```

```
##     q - k    MASE      AIC      BIC   GMRAE    MBRAE R.Adj.Sq Ljung-Box
## 10 10 - 2 1.00370 3824.015 3845.068 0.94826 -0.32411  0.38108         0
## 9   9 - 2 1.00965 3837.622 3858.685 0.95312 -0.18358  0.37257         0
## 8   8 - 2 1.01991 3862.536 3883.609 0.94784  0.32880  0.35007         0
## 7   7 - 2 1.06717 3893.555 3914.638 1.02471 -2.48798  0.31832         0
## 6   6 - 2 1.10132 3935.758 3956.851 1.05592  0.23084  0.26850         0
## 5   5 - 2 1.13173 3964.708 3985.811 1.10060  0.44803  0.23728         0
## 4   4 - 2 1.12724 3975.765 3996.878 1.06786  0.61708  0.23108         0
## 3   3 - 2 1.17975 4018.930 4040.052 1.14521  5.48844  0.17581         0
## 2   2 - 2 1.18186 4028.943 4050.076 1.13933  0.17544  0.17181         0
## 1   1 - 2 1.20116 4053.877 4070.791 1.12165  0.63659  0.14432         0
```

Models based on MASE

```
finiteDLMauto(x = as.vector(Temp+Chem1+Chem2+Size), y = as.vector(Mort), q.min = 1, q.max = 1
0, k.order = 2,
              model.type = "poly", error.type ="MASE", trace = TRUE)
```

```
##     q - k    MASE      AIC      BIC   GMRAE    MBRAE R.Adj.Sq Ljung-Box
## 10 10 - 2 1.00370 3824.015 3845.068 0.94826 -0.32411  0.38108         0
## 9   9 - 2 1.00965 3837.622 3858.685 0.95312 -0.18358  0.37257         0
## 8   8 - 2 1.01991 3862.536 3883.609 0.94784  0.32880  0.35007         0
## 7   7 - 2 1.06717 3893.555 3914.638 1.02471 -2.48798  0.31832         0
## 6   6 - 2 1.10132 3935.758 3956.851 1.05592  0.23084  0.26850         0
## 4   4 - 2 1.12724 3975.765 3996.878 1.06786  0.61708  0.23108         0
## 5   5 - 2 1.13173 3964.708 3985.811 1.10060  0.44803  0.23728         0
## 3   3 - 2 1.17975 4018.930 4040.052 1.14521  5.48844  0.17581         0
## 2   2 - 2 1.18186 4028.943 4050.076 1.13933  0.17544  0.17181         0
## 1   1 - 2 1.20116 4053.877 4070.791 1.12165  0.63659  0.14432         0
```

Hence, from the above AIC, BIC and MASE we can consider the lag-length 10 as the best in order to find out the scores and ultimately the best polynomial model.

Polynomial Model Fitting with all possible combinations

Comparing Sort Scores based on AIC

```
sort.score(AIC(poly_model_temp$model,
               poly_model_chem1$model,poly_model_chem2$model,
               poly_model_size$model,poly_model_temp.chem1$model,
               poly_model_temp.chem2$model,poly_model_temp.size$model,
               poly_model_chem1.chem2$model,poly_model_chem1.size$model,
               poly_model_chem2.size$model,poly_model_temp.chem1.size$model,
               poly_model_temp.chem2.size$model,poly_model_chem1.chem2.size$model,
               poly_model_temp.chem1.chem2.size$model), score = "aic")
```

```
##                                       df      AIC
## poly_model_chem1$model                 5 3707.006
## poly_model_chem1.size$model            5 3746.979
## poly_model_chem2.size$model            5 3769.626
## poly_model_size$model                  5 3776.444
## poly_model_chem1.chem2$model           5 3788.854
## poly_model_temp.chem1.chem2.size$model 5 3824.015
## poly_model_chem2$model                 5 3829.875
## poly_model_temp.chem2.size$model       5 3852.111
## poly_model_temp.chem1.size$model       5 3868.946
## poly_model_chem1.chem2.size$model      5 3868.946
## poly_model_temp$model                  5 3895.032
## poly_model_temp.size$model             5 3924.376
## poly_model_temp.chem2$model            5 3984.185
## poly_model_temp.chem1$model            5 3986.968
```

Comparing Sort Scores based on BIC

```
sort.score(BIC(poly_model_temp$model,
               poly_model_chem1$model,poly_model_chem2$model,
               poly_model_size$model,poly_model_temp.chem1$model,
               poly_model_temp.chem2$model,poly_model_temp.size$model,
               poly_model_chem1.chem2$model,poly_model_chem1.size$model,
               poly_model_chem2.size$model,poly_model_temp.chem1.size$model,
               poly_model_temp.chem2.size$model,poly_model_chem1.chem2.size$model,
               poly_model_temp.chem1.chem2.size$model), score = "bic")
```

```
##                                         df      BIC
## poly_model_chem1$model                   5 3728.059
## poly_model_chem1.size$model              5 3768.032
## poly_model_chem2.size$model              5 3790.679
## poly_model_size$model                    5 3797.497
## poly_model_chem1.chem2$model             5 3809.907
## poly_model_temp.chem1.chem2.size$model   5 3845.068
## poly_model_chem2$model                   5 3850.928
## poly_model_temp.chem2.size$model         5 3873.164
## poly_model_temp.chem1.size$model         5 3889.999
## poly_model_chem1.chem2.size$model        5 3889.999
## poly_model_temp$model                    5 3916.085
## poly_model_temp.size$model               5 3945.429
## poly_model_temp.chem2$model              5 4005.238
## poly_model_temp.chem1$model              5 4008.021
```

Comparing Sort Scores based on MASE

```
sort.score(MASE(poly_model_temp$model,
               poly_model_chem1$model,poly_model_chem2$model,
               poly_model_size$model,poly_model_temp.chem1$model,
               poly_model_temp.chem2$model,poly_model_temp.size$model,
               poly_model_chem1.chem2$model,poly_model_chem1.size$model,
               poly_model_chem2.size$model,poly_model_temp.chem1.size$model,
               poly_model_temp.chem2.size$model,poly_model_chem1.chem2.size$model,
               poly_model_temp.chem1.chem2.size$model), score = "mase")
```

```
##                                           n      MASE
## poly_model_chem1$model                  498 0.8907160
## poly_model_chem1.size$model             498 0.9216187
## poly_model_chem2.size$model             498 0.9362575
## poly_model_chem1.chem2$model            498 0.9478129
## poly_model_size$model                   498 0.9526993
## poly_model_chem2$model                  498 0.9860390
## poly_model_temp.chem1.chem2.size$model  498 1.0036987
## poly_model_temp.chem2.size$model        498 1.0343651
## poly_model_temp.chem1.size$model        498 1.0631728
## poly_model_chem1.chem2.size$model       498 1.0631728
## poly_model_temp$model                   498 1.1092022
## poly_model_temp.size$model              498 1.1197624
## poly_model_temp.chem2$model             498 1.1907271
## poly_model_temp.chem1$model             498 1.2028487
```

From the out puts of above sort scores based on AIC, BIC and MASE we can observe that the model “poly_model_chem1” is considered as the best model overall.

Analysisng the model “poly_model_chem1”

We will now analyse and summarize the best model from the Polynomial distributed lag model based on the AIC , BIC and MASE scores.

```
polynomial_best = polyDlm(x=as.vector(Chem1), y=as.vector(Mort), q=10, k=2)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value  P(>|t|)
## beta.0     0.323     0.1270    2.53 1.16e-02
## beta.1     0.298     0.0655    4.56 6.62e-06
## beta.2     0.281     0.0347    8.10 4.58e-15
## beta.3     0.272     0.0482    5.63 3.01e-08
## beta.4     0.269     0.0658    4.09 4.95e-05
## beta.5     0.274     0.0722    3.80 1.63e-04
## beta.6     0.287     0.0658    4.36 1.59e-05
## beta.7     0.306     0.0482    6.35 4.83e-10
## beta.8     0.333     0.0347    9.60 4.17e-20
## beta.9     0.368     0.0655    5.62 3.26e-08
## beta.10    0.410     0.1270    3.22 1.38e-03
```

```
summary(polynomial_best)
```

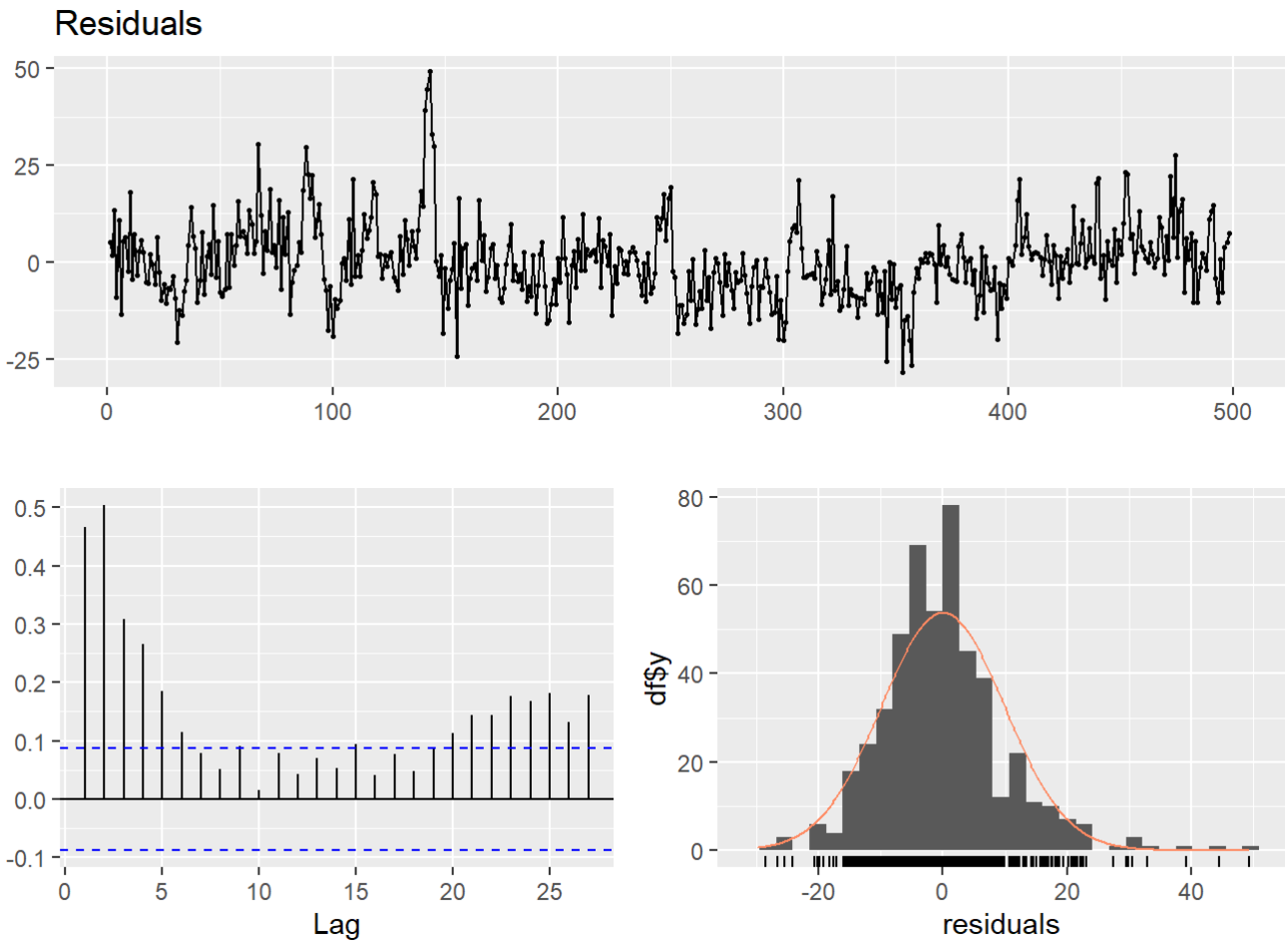
```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.469  -6.146  -0.539   4.763  49.285
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 141.818519   1.295285 109.488   <2e-16 ***
## z.t0          0.322648   0.127338   2.534   0.0116 *
## z.t1         -0.028028   0.075126  -0.373   0.7092
## z.t2          0.003672   0.007438   0.494   0.6218
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.944 on 494 degrees of freedom
## Multiple R-squared:  0.5136, Adjusted R-squared:  0.5107
## F-statistic: 173.9 on 3 and 494 DF,  p-value: < 2.2e-16
```

From the outputs of **polynomial_best** model and its summary :

- The residual standard error was 9.94 on 494 defrees of freedom..
- The adjusted R-squared value reported was 0.51 which is not bad.
- 2 out of 4 of the coefficients are significant.
- The F-statistic is reported at 173.9 on 3 and 494 Degrees of Freedom.

Residual Analysis of the model “poly_model_chem1”

```
checkresiduals(polynomial_best$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 165.47, df = 10, p-value < 2.2e-16
```

From the outputs of Model “poly_model_chem1” residuals :

- The p-value from Breusch-Godfrey test was less than significance level of 0.05.
- The ACF plot reveals that there is seasonality and correlation exist in residual due to significant lags present in the ACF.
- The time-series plot of residual seems to follow a trend overall.
- The histogram doesn’t seems to follow normal distribution and a bit right skewed.

So, we can conclude that the model “poly_model_chem1” is a decent model with an appropriate composition of residuals.

Koyck Distributed Lag Model

We will now proceed with fitting a Koyck model with the help of “koyckDlm” function and select the best lag length before choosing the best model with respect to AIC, BIC and MASE.

Koyck Model Fitting with all possible combinations

```
koyck_model_temp = koyckDlm(x=as.vector(Temp), y=as.vector(Mort))
koyck_model_chem1 = koyckDlm(x=as.vector(Chem1), y=as.vector(Mort))
koyck_model_chem2 = koyckDlm(x=as.vector(Chem2), y=as.vector(Mort))
koyck_model_size = koyckDlm(x=as.vector(Size), y=as.vector(Mort))
koyck_model_temp.chem1 = koyckDlm(x=as.vector(Temp+Chem1), y=as.vector(Mort))
koyck_model_temp.chem2 = koyckDlm(x=as.vector(Temp+Chem2), y=as.vector(Mort))
koyck_model_temp.size = koyckDlm(x=as.vector(Temp+Size), y=as.vector(Mort))
koyck_model_chem1.chem2 = koyckDlm(x=as.vector(Chem1+Chem2), y=as.vector(Mort))
koyck_model_chem1.size = koyckDlm(x=as.vector(Chem1+Size), y=as.vector(Mort))
koyck_model_chem2.size = koyckDlm(x=as.vector(Chem2+Size), y=as.vector(Mort))
koyck_model_temp.chem1.size = koyckDlm(x=as.vector(Temp+Chem1+Size), y=as.vector(Mort))
koyck_model_temp.chem2.size = koyckDlm(x=as.vector(Temp+Chem2+Size), y=as.vector(Mort))
koyck_model_chem1.chem2.size = koyckDlm(x=as.vector(Temp+Chem1+Size), y=as.vector(Mort))
koyck_model_temp.chem1.chem2.size = koyckDlm(x=as.vector(Temp+Chem1+Chem2+Size), y=as.vector
(Mort))
```

Comparing Sort Scores based on MASE

```
koyck_MASE <- MASE(koyck_model_temp,
               koyck_model_chem1,koyck_model_chem2,
               koyck_model_size,koyck_model_temp.chem1,
               koyck_model_temp.chem2,koyck_model_temp.size,
               koyck_model_chem1.chem2,koyck_model_chem1.size,
               koyck_model_chem2.size,koyck_model_temp.chem1.size,
               koyck_model_temp.chem2.size,koyck_model_chem1.chem2.size,
               koyck_model_temp.chem1.chem2.size)

# Sorting Mase scores in ascending order
arrange(koyck_MASE,MASE)
```

```
##                                    n      MASE
## koyck_model_chem1                507 0.8530742
## koyck_model_chem1.size           507 0.8736690
## koyck_model_size                 507 0.8837852
## koyck_model_chem2.size           507 0.9154523
## koyck_model_chem1.chem2          507 0.9405181
## koyck_model_chem2                507 0.9927387
## koyck_model_temp.chem1.chem2.size 507 1.0755800
## koyck_model_temp.chem1.size      507 1.0871750
## koyck_model_chem1.chem2.size     507 1.0871750
## koyck_model_temp.chem2.size      507 1.1327976
## koyck_model_temp                 507 1.1396264
## koyck_model_temp.size            507 1.1559805
## koyck_model_temp.chem1           507 1.3667254
## koyck_model_temp.chem2           507 1.6049166
```

From the above scores, we can say that the model “koyck_model_chem1” is the best one with respect to the MASE scores.We will now further analyse and summarize the model.

Analysisng the model “koyck_model_chem1”

We will now analyse and summarize the best model from the Koyck distributed lag model based on the MASE scores.

```
koyck_best = koyckDlm(x=as.vector(Chem1), y=as.vector(Mort))
summary(koyck_best)
```

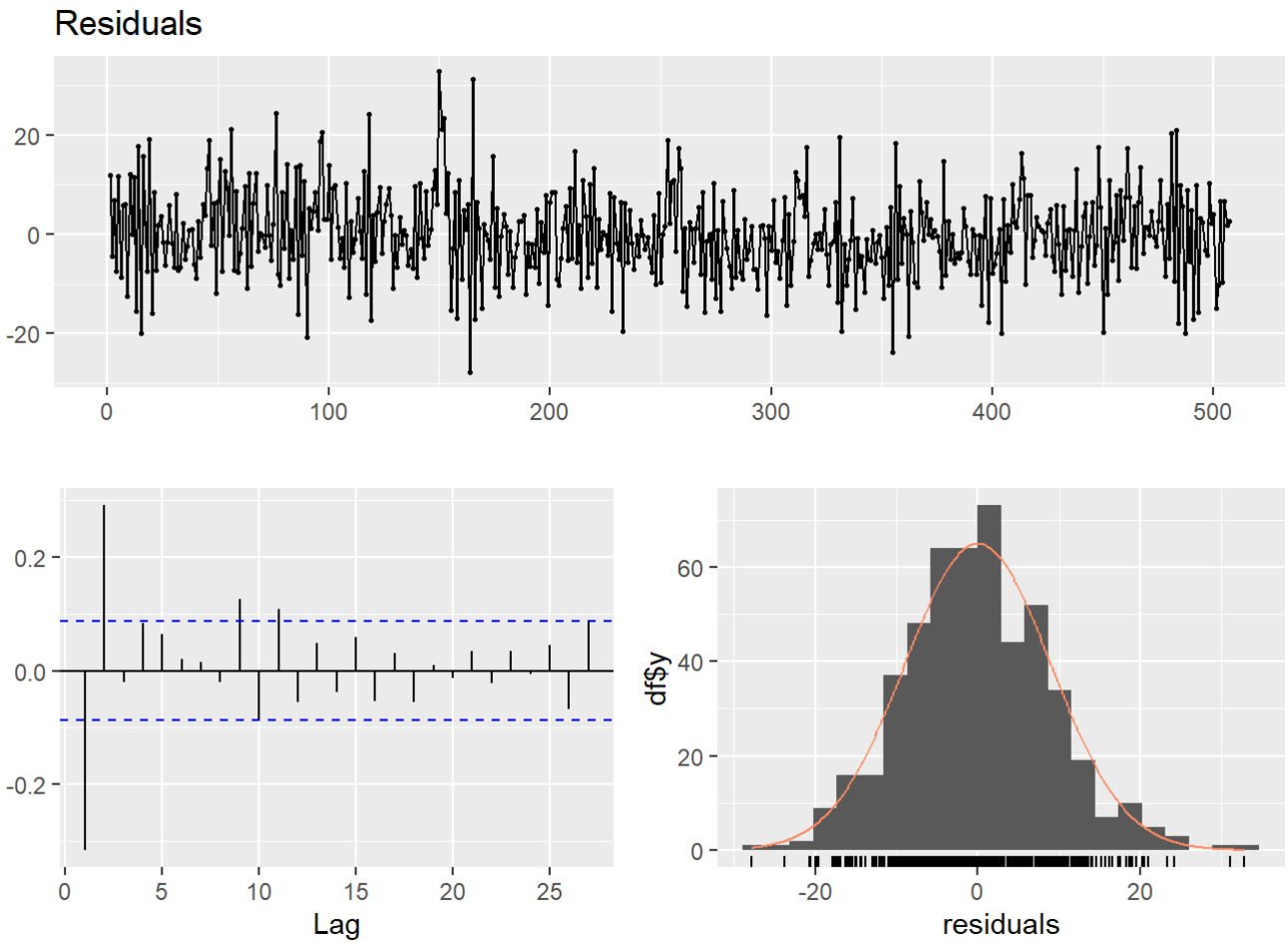
```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -27.82596  -5.89508  -0.06125   6.06967  32.82722
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 53.46578    5.34042  10.012   <2e-16 ***
## Y.1          0.65058    0.03738  17.407   <2e-16 ***
## X.t          0.70588    0.22498   3.138   0.0018 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.017 on 504 degrees of freedom
## Multiple R-Squared: 0.5974,  Adjusted R-squared: 0.5958
## Wald test: 336.8 on 2 and 504 DF,  p-value: < 2.2e-16
##
## Diagnostic tests:
## NULL
##
##                            alpha      beta       phi
## Geometric coefficients:  153.011 0.7058756 0.6505756
```

From the outputs of **koyck_best** model and its summary :

- The residual standard error was 9.01 on 504 degrees of freedom..
- The adjusted R-squared value reported was 0.59 which is better than previous ones.
- All the coefficients of this model are significant.
- The Wald test is reported at 336.8 on 2 and 504 Degrees of Freedom.
- The values of Alpha , Beta and phi are 153.011, 0.70587 and 0.65057.

Residual Analysis of the model “koyck_model_chem1”

```
checkresiduals(koyck_best$model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 112.71, df = 10, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 10
```

From the outputs of Model “koyck_model_chem1” residuals :

- The p-value from Breusch-Godfrey test was less than significance level of 0.05.
- The ACF plot reveals that there correlation exist in residual due to significant lags present in the ACF.
- The time-series plot of residual doesnot seems to follow a trend overall.
- The histogram doesn’t seems to follow normal distribution and a bit right skewed.

So, we can conclude that the model “koyck_model_chem1” is a decent model with an appropriate composition of residuals.

Autoregressive Distributed Lag Model

We will now proceed with fitting a Auto-regressive Distributed lag model with the help of for loop “ardlDlm” function and select the best lag length before choosing the best model with respect to AIC, BIC and MASE.

```
for (i in 1:5){
  for(j in 1:5){
    autoreg_model = ardlDlm(x= as.vector(Temp+Chem1+Chem2+Size),y=as.vector(Mort), p = i , q
= j )
    cat("p =", i, "q =", j, "AIC =", AIC(autoreg_model$model), "BIC =", BIC(autoreg_model$mod
el), "MASE =", MASE(autoreg_model)$MASE, "\n")
  }
}
```

```
## p = 1 q = 1 AIC = 3662.811 BIC = 3683.954 MASE = 0.8376494
## p = 1 q = 2 AIC = 3544.746 BIC = 3570.106 MASE = 0.7383828
## p = 1 q = 3 AIC = 3540.477 BIC = 3570.049 MASE = 0.7375923
## p = 1 q = 4 AIC = 3536.323 BIC = 3570.104 MASE = 0.7388742
## p = 1 q = 5 AIC = 3530.81 BIC = 3568.795 MASE = 0.7371776
## p = 2 q = 1 AIC = 3641.169 BIC = 3666.529 MASE = 0.8321567
## p = 2 q = 2 AIC = 3546.696 BIC = 3576.282 MASE = 0.738951
## p = 2 q = 3 AIC = 3542.404 BIC = 3576.2 MASE = 0.7381645
## p = 2 q = 4 AIC = 3538.227 BIC = 3576.23 MASE = 0.7394541
## p = 2 q = 5 AIC = 3532.733 BIC = 3574.939 MASE = 0.7376235
## p = 3 q = 1 AIC = 3636.719 BIC = 3666.291 MASE = 0.8321717
## p = 3 q = 2 AIC = 3542.187 BIC = 3575.984 MASE = 0.7395773
## p = 3 q = 3 AIC = 3544.118 BIC = 3582.139 MASE = 0.7389992
## p = 3 q = 4 AIC = 3539.923 BIC = 3582.149 MASE = 0.740296
## p = 3 q = 5 AIC = 3534.443 BIC = 3580.87 MASE = 0.7386002
## p = 4 q = 1 AIC = 3605.136 BIC = 3638.916 MASE = 0.8049845
## p = 4 q = 2 AIC = 3518.822 BIC = 3556.826 MASE = 0.7310683
## p = 4 q = 3 AIC = 3520.559 BIC = 3562.784 MASE = 0.7295792
## p = 4 q = 4 AIC = 3521.213 BIC = 3567.662 MASE = 0.7304854
## p = 4 q = 5 AIC = 3514.937 BIC = 3565.584 MASE = 0.7273552
## p = 5 q = 1 AIC = 3599.747 BIC = 3637.732 MASE = 0.8049032
## p = 5 q = 2 AIC = 3513.153 BIC = 3555.359 MASE = 0.7299388
## p = 5 q = 3 AIC = 3514.819 BIC = 3561.246 MASE = 0.7282131
## p = 5 q = 4 AIC = 3515.473 BIC = 3566.12 MASE = 0.7291745
## p = 5 q = 5 AIC = 3516.933 BIC = 3571.801 MASE = 0.7273257
```

From the above results of sort score, the lowest value of MASE has been recorded at ARDL(5,5). Hence we will consider p=5 and q=5 while selecting and analyzing the best model.

```
ardl_model_temp = ardlDlm(x=as.vector(Temp), y=as.vector(Mort), p = 5, q = 5)
ardl_model_chem1 = ardlDlm(x=as.vector(Chem1), y=as.vector(Mort), p = 5, q = 5)
ardl_model_chem2 = ardlDlm(x=as.vector(Chem2), y=as.vector(Mort), p = 5, q = 5)
ardl_model_size = ardlDlm(x=as.vector(Size), y=as.vector(Mort), p = 5, q = 5)
ardl_model_temp.chem1 = ardlDlm(x=as.vector(Temp+Chem1), y=as.vector(Mort), p = 5, q = 5)
ardl_model_temp.chem2 = ardlDlm(x=as.vector(Temp+Chem2), y=as.vector(Mort), p = 5, q = 5)
ardl_model_temp.size = ardlDlm(x=as.vector(Temp+Size), y=as.vector(Mort), p = 5, q = 5)
ardl_model_chem1.chem2 = ardlDlm(x=as.vector(Chem1+Chem2), y=as.vector(Mort), p = 5, q = 5)
ardl_model_chem1.size = ardlDlm(x=as.vector(Chem1+Size), y=as.vector(Mort), p = 5, q = 5)
ardl_model_chem2.size = ardlDlm(x=as.vector(Chem2+Size), y=as.vector(Mort), p = 5, q = 5)
ardl_model_temp.chem1.size = ardlDlm(x=as.vector(Temp+Chem1+Size), y=as.vector(Mort), p = 5,
q = 5)
ardl_model_temp.chem2.size = ardlDlm(x=as.vector(Temp+Chem2+Size), y=as.vector(Mort), p = 5,
q = 5)
ardl_model_chem1.chem2.size = ardlDlm(x=as.vector(Temp+Chem1+Size), y=as.vector(Mort), p = 5,
q = 5)
ardl_model_temp.chem1.chem2.size = ardlDlm(x=as.vector(Temp+Chem1+Chem2+Size), y=as.vector(Mo
rt), p = 5, q = 5)
```

Comparing Sort Scores based on MASE

```
ardl_MASE <- MASE(ardl_model_temp,
                  ardl_model_chem1,ardl_model_chem2,
                  ardl_model_size,ardl_model_temp.chem1,
                  ardl_model_temp.chem2,ardl_model_temp.size,
                  ardl_model_chem1.chem2,ardl_model_chem1.size,
                  ardl_model_chem2.size,ardl_model_temp.chem1.size,
                  ardl_model_temp.chem2.size,ardl_model_chem1.chem2.size,
                  ardl_model_temp.chem1.chem2.size)

# Sorting Mase scores in ascending order
arrange(ardl_MASE,MASE)
```

```
##                                    n      MASE
## ardl_model_chem1.size            503 0.7221903
## ardl_model_chem1                 503 0.7265008
## ardl_model_chem2.size            503 0.7267610
## ardl_model_temp.chem1.chem2.size 503 0.7273257
## ardl_model_size                  503 0.7274093
## ardl_model_temp.chem1.size       503 0.7287897
## ardl_model_chem1.chem2.size      503 0.7287897
## ardl_model_temp.chem2.size       503 0.7292318
## ardl_model_chem1.chem2           503 0.7321395
## ardl_model_temp.size             503 0.7337199
## ardl_model_chem2                 503 0.7373449
## ardl_model_temp.chem2            503 0.7446913
## ardl_model_temp                  503 0.7457624
## ardl_model_temp.chem1            503 0.7544646
```

From the above outputs, we can observe that the best Autoregressive distributed lag model being selected based on MASE scores is the “ardl_model_chem1.size” which is combination of predictors Chemical 1 and particle size.Hence we will further analyse the model and do the summary and residual outputs.

Analysisng the model “ardl_model_chem1.size”

We will now analyse and summarize the best model from the Autoregressive distributed lag model based on the MASE scores.

```
ardl_best = ardlDlm(x=as.vector(Chem1+Size), y=as.vector(Mort))
summary(ardl_best)
```

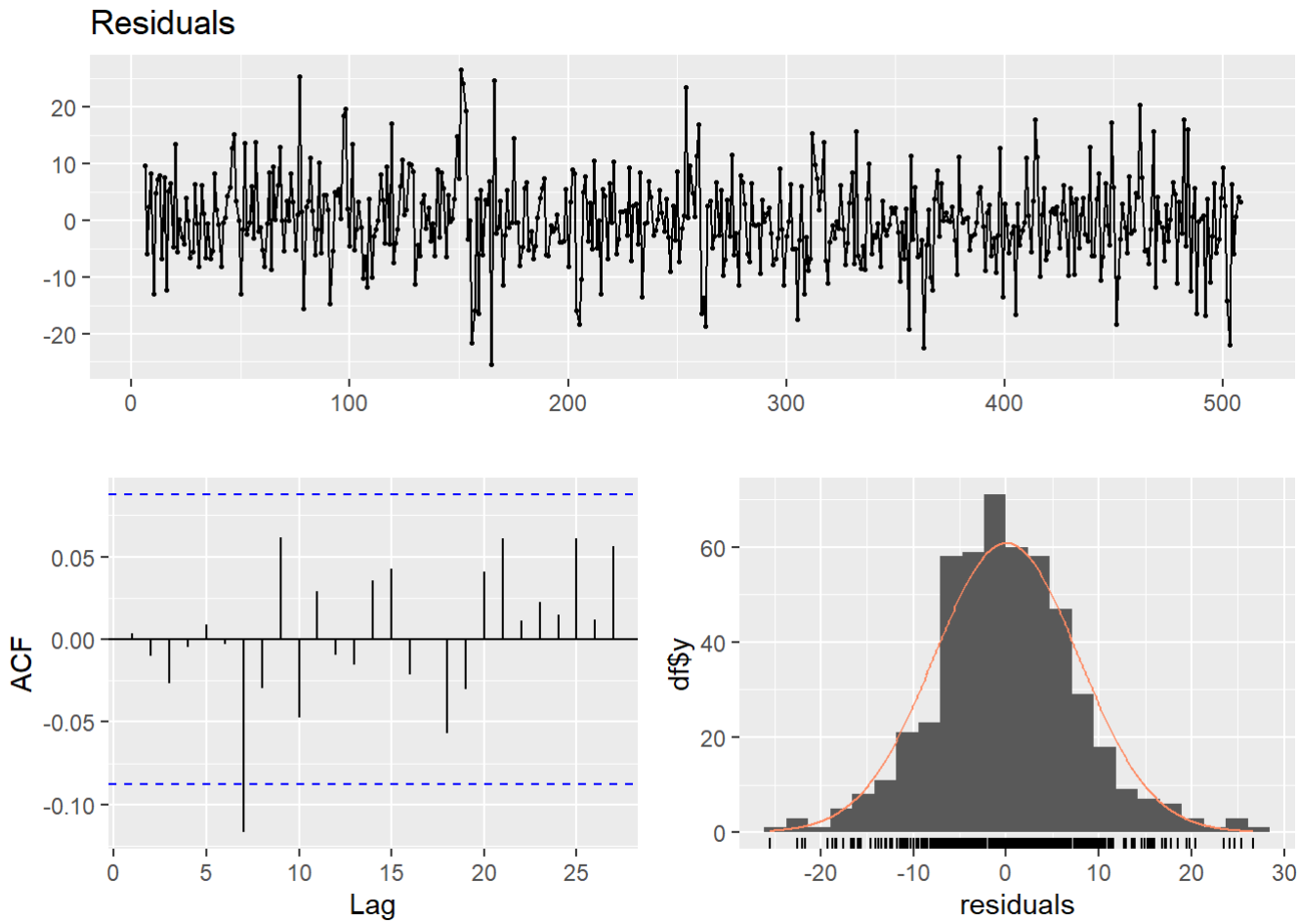
```
##
## Time series regression with "ts" data:
## Start = 2, End = 508
##
## Call:
## dynlm(formula = as.formula(model.text), data = data, start = 1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -27.117  -5.410  -0.361   6.100  33.665
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 47.15956    4.95374   9.520  < 2e-16 ***
## X.t          0.24983    0.02641   9.459  < 2e-16 ***
## X.1         -0.08313    0.02849  -2.918  0.00368 **
## Y.1          0.66639    0.03256  20.465  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.955 on 503 degrees of freedom
## Multiple R-squared:  0.6037, Adjusted R-squared:  0.6013
## F-statistic: 255.4 on 3 and 503 DF,  p-value: < 2.2e-16
```

From the outputs of **ardl_best** model and its summary :

- The residual standard error was 8.955 on 503 degrees of freedom..
- The adjusted R-squared value reported was 0.60 which is better than previous ones.
- All the coefficients of this model are significant.
- The F-statistic is reported at 255.4 on 3 and 503 Degrees of Freedom.
- The p-values was less than 5% level.

Residual Analysis of the model “ardl_model_chem1.size”

```
checkresiduals(ardl_model_chem1.size$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 15
##
## data:  Residuals
## LM test = 23.795, df = 15, p-value = 0.06866
```

From the outputs of Model “ardl_model_chem1.size” residuals :

- The p-value from Breusch-Godfrey test was greater than significance level of 0.05.
- The ACF plot reveals that there is less correlation exist in residual due to just one significant lag present in the ACF.
- The time-series plot of residual doesn’t seems to follow a trend overall.
- The histogram seems to follow normal distribution overall.

So, we can conclude that the model “ardl_model_chem1.size” is a better model as compared to previos ones with an appropriate composition of residuals.

Comparing all Lag models above wrt MASE:

```
mort_dlm_mase <- MASE(finite_best,polynomial_best,koyck_best,ardl_best)
print(mort_dlm_mase)
```

```
##                   n      MASE
## finite_best     498 0.8570098
## polynomial_best 498 0.8907160
## koyck_best      507 0.8530742
## ardl_best       507 0.8383696
```

Hence from the above output, we can confirm that “Finite_best” model from finite dlm is the best model based on MASE scores as compared to the rest. Hence model finite with “particle size” as predictor is the best model.

Dynamic Linear Models

We’ll utilize the dynlm() function in the “dynlm” package to fit the model. In order to incorporate a trend component and a seasonal component, the argument formula for the model can contain the functions trend() and season().

```
# Variables for Trend and Seasonality under Dynlm models
Y.t <- Mort
T <-  156   # first intervention point in year 2013(156 weeks)
S.t <-  1*(seq(Y.t) == T)
S.t.1 <-  Lag(S.t,+1)
```

```
# Fitting dynamic linear models
dynlm1 <-  dynlm(Y.t ~ L(Y.t , k = 1 ) + S.t + trend(Y.t) + season(Y.t))

dynlm2 <-  dynlm(Y.t ~ L(Y.t , k = 2 ) + S.t + trend(Y.t) + season(Y.t))

dynlm3 <-  dynlm(Y.t ~ L(Y.t , k = 1 ) + S.t + season(Y.t))

dynlm4 <-  dynlm(Y.t ~ L(Y.t , k = 1 ) + S.t + trend(Y.t))

dynlm5 <-  dynlm(Y.t ~ Chem1 + L(Y.t , k = 2 ) + S.t + trend(Y.t) + season(Y.t))

dynlm6 <-  dynlm(Y.t ~ Chem2 + L(Y.t , k = 2 ) + S.t + trend(Y.t) + season(Y.t))

dynlm7 <-  dynlm(Y.t ~ Temp + L(Y.t , k = 2 ) + S.t + trend(Y.t) + season(Y.t))

dynlm8 <-  dynlm(Y.t ~ Size + L(Y.t , k = 2 ) + S.t + trend(Y.t) + season(Y.t))
```

Dynamic Linear models compariosn

```
dynlm_mase <- MASE(lm(dynlm1), lm(dynlm2), lm(dynlm3), lm(dynlm4),lm(dynlm5),lm(dynlm6),lm(dy
nlm7),lm(dynlm8))
arrange(dynlm_mase,MASE)
```

```
##              n      MASE
## lm(dynlm8) 506 0.6935111
## lm(dynlm5) 506 0.6944631
## lm(dynlm6) 506 0.6967146
## lm(dynlm7) 506 0.7038379
## lm(dynlm2) 506 0.7277890
## lm(dynlm1) 507 0.7655602
## lm(dynlm3) 507 0.8000698
## lm(dynlm4) 507 0.9199715
```

From the above result of MASE scores , we can say that the model “dynlm8” with predictor “particle size” comes out to be the best among all with least MASE and AIC scores. Hence we will further analyse the model and do summary and residual analysis.

Analysisng the model “dynlm8”

We will now analyse and summarize the best model “dynlm8” from the Dynamic linear model based on the MASE scores.

```
dynlm_best = dynlm8 <-  dynlm(Y.t ~ Size + L(Y.t , k = 2 ) + S.t + trend(Y.t) + season(Y.t))
summary(dynlm_best)
```

```
##
## Time series regression with "ts" data:
## Start = 2010(3), End = 2019(40)
##
## Call:
## dynlm(formula = Y.t ~ Size + L(Y.t, k = 2) + S.t + trend(Y.t) +
##     season(Y.t))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -24.084  -4.384  -0.170   4.296  33.376
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    76.63876    8.24142   9.299  < 2e-16 ***
## Size            0.19519    0.03610   5.407 1.04e-07 ***
## L(Y.t, k = 2)   0.52070    0.03933  13.241  < 2e-16 ***
## S.t           -23.29617    8.58052  -2.715 0.006882 **
## trend(Y.t)     -0.72166    0.14370  -5.022 7.39e-07 ***
## season(Y.t)2    0.67879    3.76591   0.180 0.857042
## season(Y.t)3   -3.21644    3.67837  -0.874 0.382356
## season(Y.t)4   -0.53665    3.69802  -0.145 0.884682
## season(Y.t)5    3.73522    3.70151   1.009 0.313466
## season(Y.t)6   -0.67344    3.72986  -0.181 0.856799
## season(Y.t)7   -3.63671    3.73564  -0.974 0.330820
## season(Y.t)8   -0.19904    3.75089  -0.053 0.957705
## season(Y.t)9   -2.41029    3.78929  -0.636 0.525048
## season(Y.t)10  -0.51859    3.79160  -0.137 0.891271
## season(Y.t)11  -0.70132    3.78696  -0.185 0.853161
## season(Y.t)12  -3.76055    3.80844  -0.987 0.323964
## season(Y.t)13   0.23823    3.78203   0.063 0.949802
## season(Y.t)14  -4.38002    3.84843  -1.138 0.255671
## season(Y.t)15  -0.39575    3.80278  -0.104 0.917160
## season(Y.t)16  -1.90751    3.88486  -0.491 0.623658
## season(Y.t)17  -4.59618    3.81476  -1.205 0.228898
## season(Y.t)18  -3.63969    3.84015  -0.948 0.343740
## season(Y.t)19  -2.02974    3.83614  -0.529 0.596990
## season(Y.t)20   0.22668    3.82005   0.059 0.952707
## season(Y.t)21  -4.88058    3.82625  -1.276 0.202773
## season(Y.t)22  -6.71141    3.79059  -1.771 0.077313 .
## season(Y.t)23  -4.10830    3.85147  -1.067 0.286686
## season(Y.t)24  -4.47205    3.81316  -1.173 0.241498
## season(Y.t)25   1.80779    3.83165   0.472 0.637294
## season(Y.t)26  -5.01294    3.82821  -1.309 0.191042
## season(Y.t)27  -9.30437    3.78192  -2.460 0.014260 *
## season(Y.t)28  -6.62777    3.84042  -1.726 0.085071 .
## season(Y.t)29  -3.46486    3.84664  -0.901 0.368202
## season(Y.t)30  -1.89500    3.85398  -0.492 0.623172
## season(Y.t)31  -4.90142    3.81823  -1.284 0.199911
## season(Y.t)32  -3.06971    3.79673  -0.809 0.419222
## season(Y.t)33  -1.14445    3.80930  -0.300 0.763984
## season(Y.t)34  -3.27457    3.77451  -0.868 0.386104
## season(Y.t)35 -10.77984    3.76061  -2.867 0.004345 **
## season(Y.t)36  -4.53127    3.76606  -1.203 0.229537
## season(Y.t)37  -2.33463    3.81784  -0.612 0.541173
```

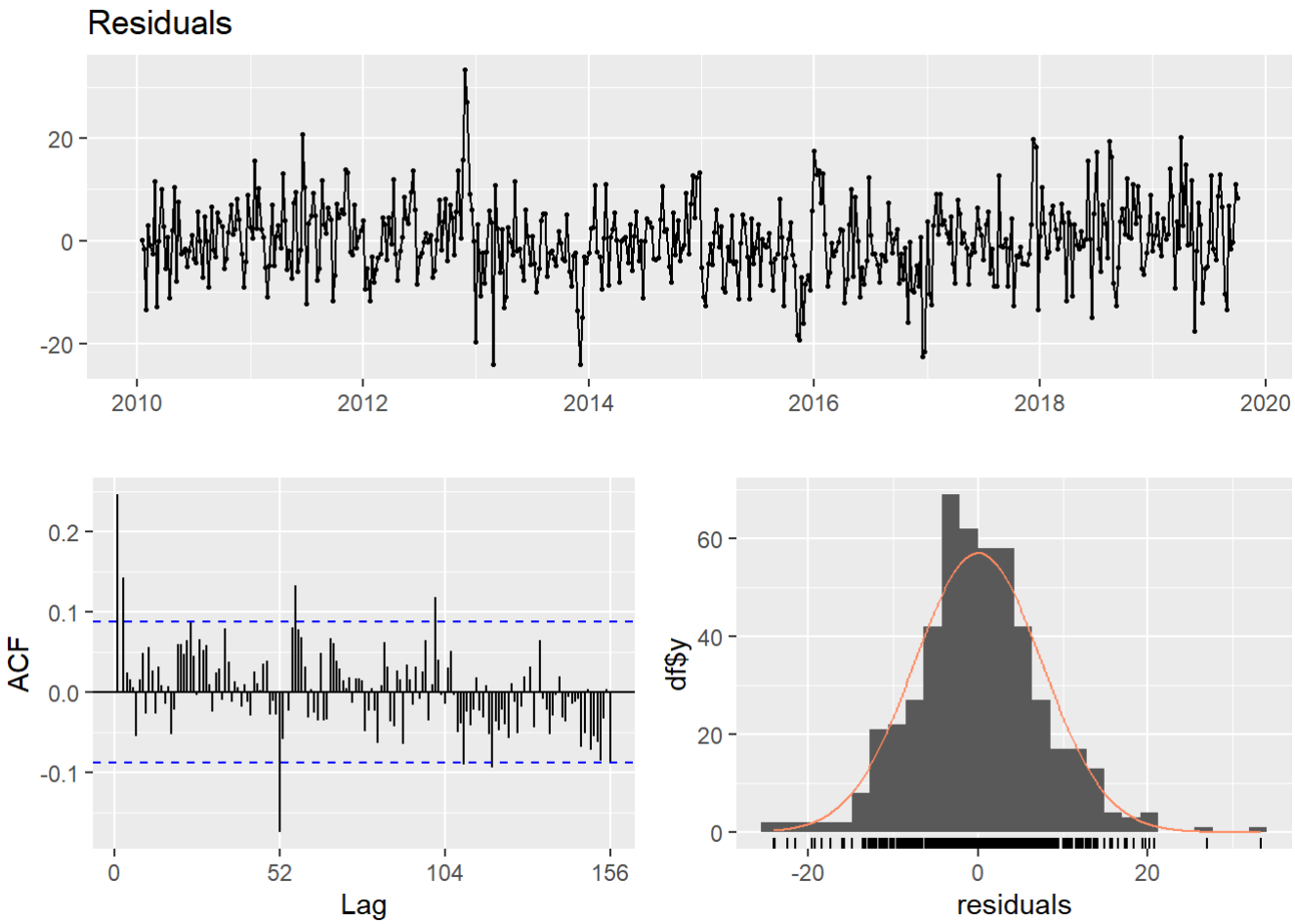
```
## season(Y.t)38  -1.64991    3.76419  -0.438 0.661366
## season(Y.t)39  -2.90864    3.78527  -0.768 0.442647
## season(Y.t)40  -4.83405    3.72869  -1.296 0.195486
## season(Y.t)41  -3.65024    3.85328  -0.947 0.343991
## season(Y.t)42  -0.03914    3.83637  -0.010 0.991865
## season(Y.t)43   0.79915    3.86474   0.207 0.836274
## season(Y.t)44  -1.95856    3.81912  -0.513 0.608322
## season(Y.t)45   5.00054    3.81081   1.312 0.190122
## season(Y.t)46  13.22991    3.80452   3.477 0.000555 ***
## season(Y.t)47  13.49617    3.77957   3.571 0.000394 ***
## season(Y.t)48   5.01448    3.76956   1.330 0.184107
## season(Y.t)49   4.94759    3.78649   1.307 0.192000
## season(Y.t)50   1.25049    3.76759   0.332 0.740114
## season(Y.t)51   0.95982    3.77229   0.254 0.799272
## season(Y.t)52  10.12698    3.87948   2.610 0.009345 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.974 on 450 degrees of freedom
## Multiple R-squared:  0.7175, Adjusted R-squared:  0.683
## F-statistic: 20.78 on 55 and 450 DF,  p-value: < 2.2e-16
```

From the outputs of **dynlm8** model and its summary :

- The residual standard error was 7.974 on 450 degrees of freedom..
- The adjusted R-squared value reported was 0.68 which is better than previous ones.
- Very few of the coefficients of this model are significant.
- The F-statistic is reported at 20.78 on 55 and 450 Degrees of Freedom.
- The p-values was less than 5% level.

Residual Analysis of the model “dynlm8”

```
checkresiduals(dynlm8)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 101
##
## data:  Residuals
## LM test = 137.8, df = 101, p-value = 0.0088
```

From the outputs of Model “dynlm8” residuals :

- The p-value from Breusch-Godfrey test was less than significance level of 0.05.
- The ACF plot reveals that there is less correlation exist in residual due to significant lags present in the ACF.
- The time-series plot of residual seems to follow a trend overall with seasonal repititions.
- The histogram seems to follow normal distribution overall.

So, we can conclude that the model “dynlm8” is a better model as compared to previous ones with an appropriate composition of residuals.

Exponential Smoothing Method

We’ll now test with exponential smoothing as a forecasting technique. We will only take into account models that feature either additive or multiplicative seasonality because we have discovered a substantial seasonal component in mortality series that we wish to make predictions for.

```
# Converting Mortality data series into monthly format

Mort_monthly = aggregate(zoo(Mort),as.yearmon,sum)

Mort_monthly[Mort_monthly==12.11] <- NA

Mort_monthly = na.omit(Mort_monthly)

# Converting into Time series format

Mort_monthly <- ts(as.vector(t(as.matrix(Mort_monthly[,2:13]))),start=c(2010,1),end = c(2019,
9),frequency = 12)

# Print
Mort_monthly
```

```
##         Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct
## 2010 913.04 713.48 705.14 851.91 691.87 679.72 840.88 696.56 702.46 896.10
## 2011 963.78 711.90 680.13 855.36 677.42 698.50 852.82 687.97 711.36 908.35
## 2012 897.31 656.03 684.92 833.02 675.90 697.77 812.01 645.44 695.71 903.45
## 2013 850.55 649.11 671.85 781.07 656.56 639.04 772.32 655.37 660.61 845.76
## 2014 885.88 676.66 669.02 802.01 638.39 626.91 791.24 649.91 660.36 857.10
## 2015 865.65 658.96 634.53 764.68 628.18 614.19 761.53 619.34 642.44 846.91
## 2016 944.11 676.38 646.66 780.69 630.22 604.93 747.50 608.05 641.54 800.51
## 2017 816.00 679.75 657.28 792.25 612.97 631.55 775.86 607.53 635.01 812.60
## 2018 881.25 669.59 646.80 784.83 624.77 647.22 806.38 674.23 650.88 856.67
## 2019 875.23 675.84 658.92 854.95 638.46 614.07 792.92 648.05 646.56
##         Nov    Dec
## 2010 805.59 789.13
## 2011 824.51 798.77
## 2012 850.28 844.24
## 2013 718.37 678.03
## 2014 762.88 809.61
## 2015 676.04 686.34
## 2016 700.28 670.17
## 2017 698.62 752.11
## 2018 758.30 716.41
## 2019
```

Holt-Winters' Trend and Seasonality Method

```
# Estimating the model based on several scores

seasonal = c("additive","multiplicative")
damped = c(TRUE,FALSE)
expand = expand.grid(seasonal,damped)
hw_AIC = array(NA,4)
hw_BIC = array(NA,4)
hw_MASE=array(NA,4)
steps = array(NA, dim=c(4,2))

# Iterating through a for loop

for(i in 1:4){hw_AIC[i]= hw(Mort_monthly,
                            seasonal = toString(expand[i ,1]),
                            damped = expand[i ,2])$model$aic
              hw_BIC[i]= hw(Mort_monthly,
                            seasonal = toString(expand[i ,1]),
                            damped = expand[i ,2])$model$bic

               hw_MASE[i]= accuracy(hw(Mort_monthly,
                            seasonal = toString(expand[i ,1]),
                            damped = expand[i ,2]))[,6]
steps[i,1]= toString(expand[i ,1])
steps[i,2]= expand[i ,2]
}
accuracy_hw = data.frame(steps, hw_MASE, hw_AIC,hw_BIC)
colnames(accuracy_hw)= c("seasonal","Damped","MASE", "AIC","BIC")

# Printing accuracy scores

arrange(accuracy_hw,MASE)
```

```
##         seasonal Damped      MASE      AIC      BIC
## 1 multiplicative   TRUE 0.6691194 1356.636 1406.356
## 2       additive   TRUE 0.6703678 1363.971 1413.690
## 3       additive  FALSE 0.6738379 1362.750 1409.707
## 4 multiplicative  FALSE 0.6868037 1359.280 1406.237
```

From the results presented, it's evident that the Holt-Winter's multiplicative seasonality model outperforms others in terms of MASE and AIC. Therefore, we should delve deeper into exploring all potential Holt-Winter's multiplicative seasonality models since they seem to surpass the additive ones in terms of precision metrics.

Fitting Holt-Winters' Trend and Seasonality models

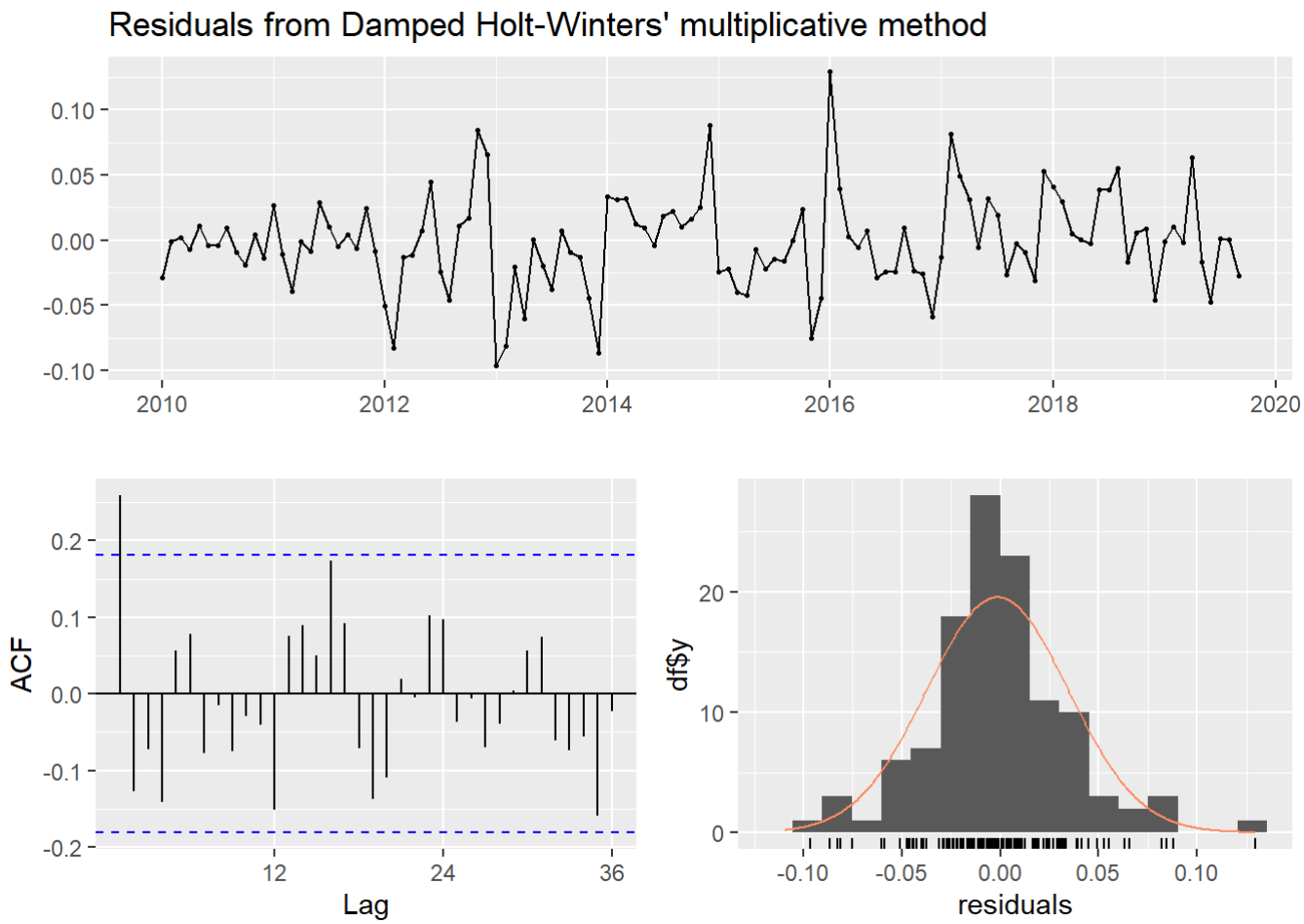
Fitting multiplicative model with damped as True

```
hw1 <- hw(Mort_monthly, seasonal = "multiplicative", damped = TRUE)

# Checking summary
summary(hw1$model)
```

```
## Damped Holt-Winters' multiplicative method
##
## Call:
##  hw(y = Mort_monthly, seasonal = "multiplicative", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.1668
##     beta  = 0.0073
##     gamma = 1e-04
##     phi   = 0.977
##
##   Initial states:
##     l = 778.2929
##     b = 0.8527
##     s = 1.0272 1.031 1.1705 0.9081 0.8856 1.0838
##            0.8761 0.8808 1.1034 0.9065 0.9205 1.2067
##
##   sigma:  0.0388
##
##      AIC     AICc      BIC
## 1356.636 1363.616 1406.356
##
## Training set error measures:
##                      ME    RMSE      MAE        MPE     MAPE      MASE      ACF1
## Training set -1.278416 27.3346 19.39936 -0.2724535 2.614002 0.6691194 0.2298084
```

```
# Checking Residuals
checkresiduals(hw1)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt-Winters' multiplicative method
## Q* = 33.416, df = 23, p-value = 0.07408
##
## Model df: 0.   Total lags used: 23
```

From the outputs of above Model “hw1” and its residuals :

- The critical values of AIC is 1356.636, AICc is 1363.616 and BIC is 1406.356.
- The MASE value was reported at 0.66911.
- The is No trend observed in the residual time series.
- The p-value from the Ljung-Box test was above **5%** level of significance.
- The histogram seems to be normally distributed overall.
- The ACF plot have first significant **Lag** at the stating point.

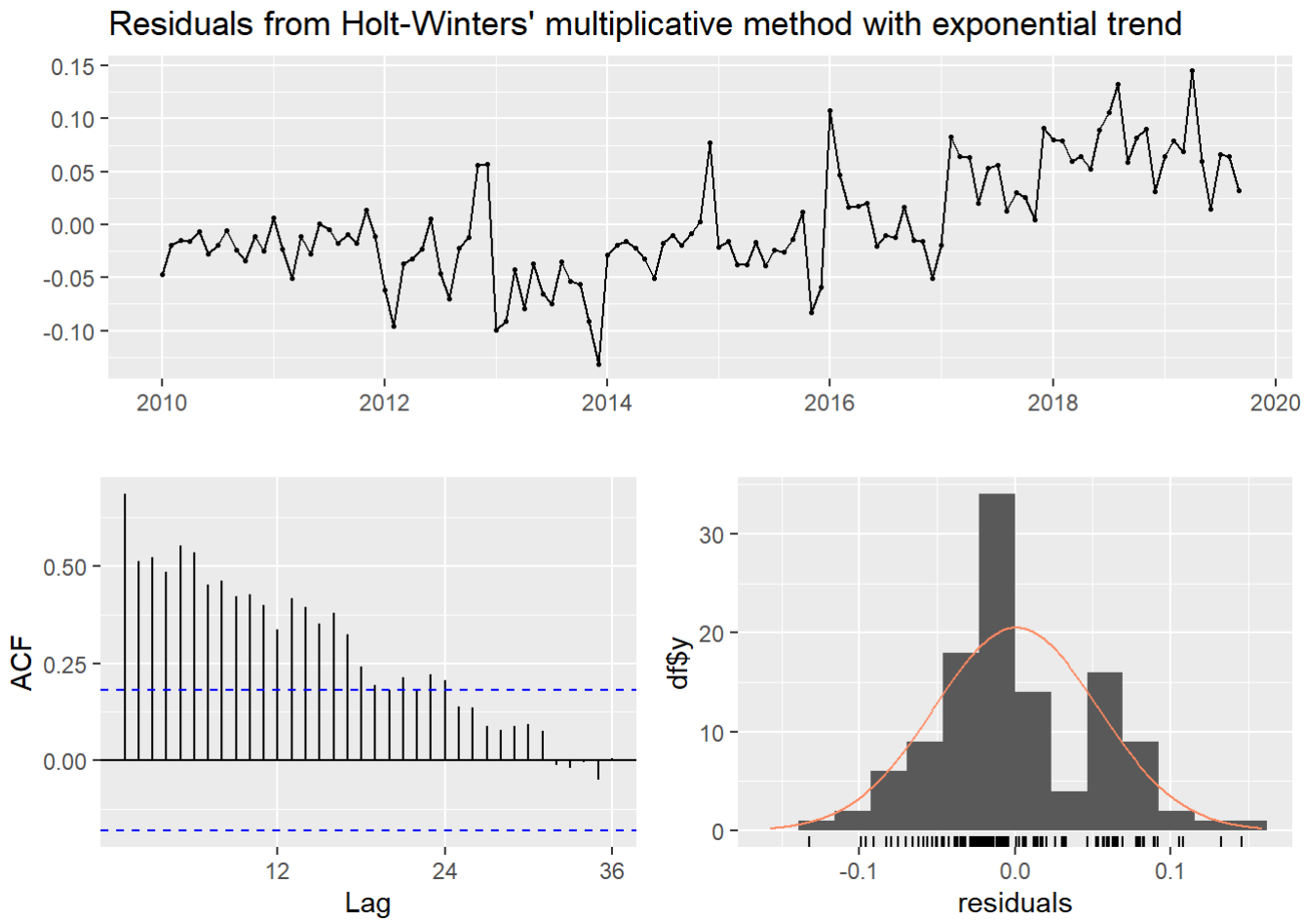
Fitting multiplicative model with exponential as TRUE

```
hw2 <- hw(Mort_monthly, seasonal = "multiplicative", exponential = TRUE)

#Checking summary
summary(hw2$model)
```

```
## Holt-Winters' multiplicative method with exponential trend
##
## Call:
##  hw(y = Mort_monthly, seasonal = "multiplicative", exponential = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.0168
##     beta  = 0.0018
##     gamma = 0.0493
##
##   Initial states:
##     l = 789.6523
##     b = 1.0007
##     s = 1.0228 1.0292 1.1714 0.9094 0.8853 1.0842
##            0.8839 0.8809 1.0943 0.9053 0.9209 1.2124
##
##   sigma:  0.0564
##
##      AIC     AICc      BIC
## 1443.296 1449.478 1490.253
##
## Training set error measures:
##                      ME     RMSE      MAE         MPE     MAPE     MASE      ACF1
## Training set -1.093152 38.60066 30.58244 -0.2054225 4.165848 1.054844 0.6328161
```

```
# Checking Residuals
checkresiduals(hw2)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' multiplicative method with exponential trend
## Q* = 492.14, df = 23, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 23
```

From the outputs of above Model “hw2” and its residuals :

- The critical values of AIC is 1443.296, AICc is 1449.478 and BIC is 1490.253.
- The MASE value was reported at 1.054844.
- The is an upward trend observed in the residual time series.
- The p-value from the Ljung-Box test was below **5%** level of significance.
- The histogram doesn’t seems to be normally distributed overall.
- The ACF plot have many significant **Lags** at the stating point and overall.

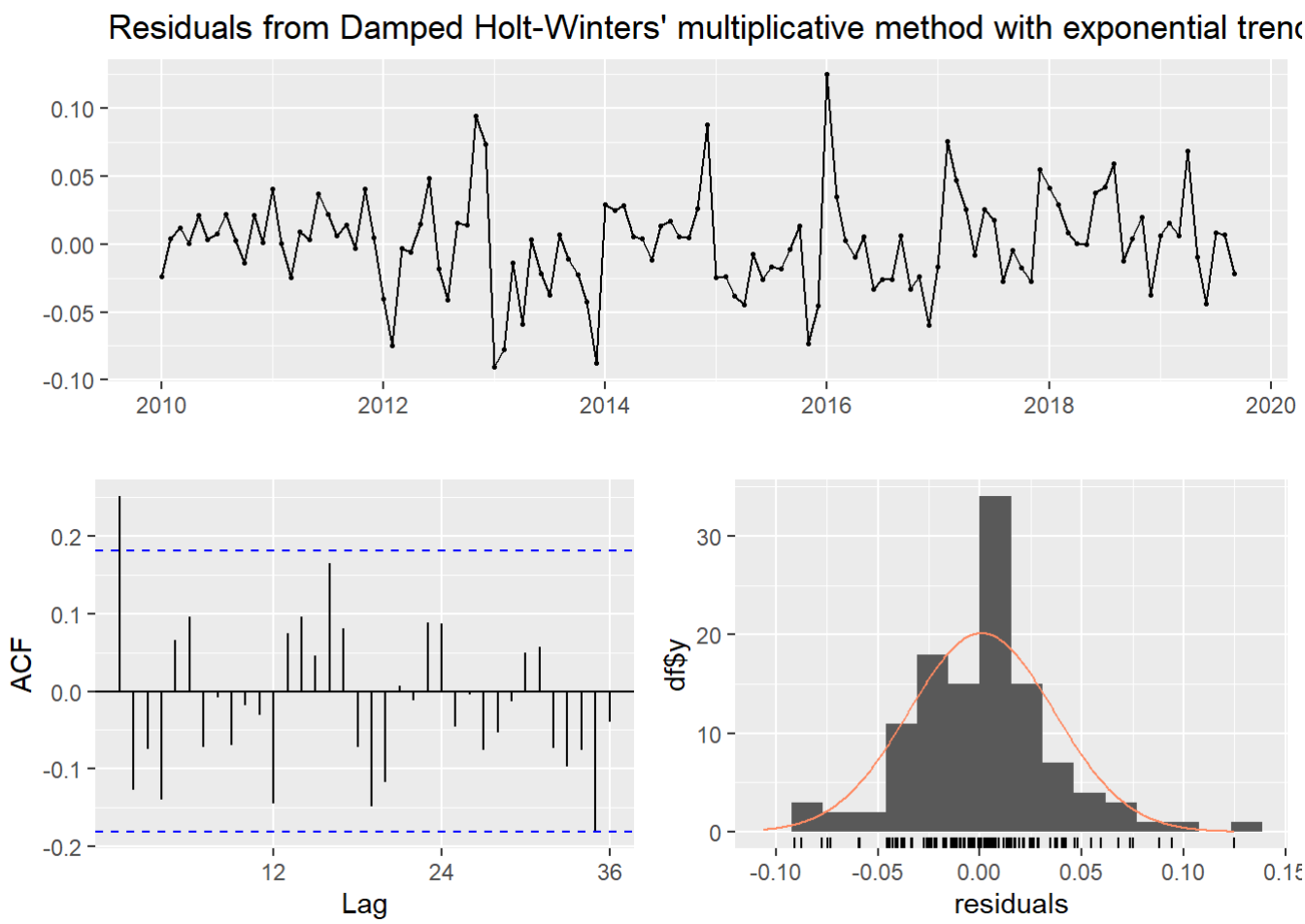
Fitting multiplicative model with exponential and damped as TRUE

```
hw3 <- hw(Mort_monthly, seasonal = "multiplicative", exponential = TRUE, damped = TRUE)

#Checking summary
summary(hw3$model)
```

```
## Damped Holt-Winters' multiplicative method with exponential trend
##
## Call:
##  hw(y = Mort_monthly, seasonal = "multiplicative", damped = TRUE,
##
##  Call:
##      exponential = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.1615
##     beta  = 1e-04
##     gamma = 1e-04
##     phi   = 0.98
##
##   Initial states:
##     l = 777.7363
##     b = 0.9982
##     s = 1.0245 1.0269 1.1787 0.9083 0.8853 1.0834
##            0.8782 0.88 1.1044 0.9041 0.9209 1.2053
##
##   sigma:  0.0385
##
##      AIC     AICc      BIC
## 1354.114 1361.094 1403.833
##
## Training set error measures:
##                      ME    RMSE     MAE         MPE     MAPE      MASE      ACF1
## Training set 0.4102562 27.0577 19.3956 -0.05197752 2.609351 0.6689894 0.2221417
```

```
# Checking Residuals
checkresiduals(hw3)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt-Winters' multiplicative method with exponential trend
## Q* = 32.523, df = 23, p-value = 0.08976
##
## Model df: 0.   Total lags used: 23
```

From the outputs of above Model “hw3” and its residuals :

- The critical values of AIC is 1354.114, AICc is 1361.094 and BIC is 1403.833.
- The MASE value was reported at 0.6689894.
- The is No trend observed in the residual time series.
- The p-value from the Ljung-Box test was above **5%** level of significance.
- The histogram doesn’t seems to be normally distributed overall.
- The ACF plot have one significant **Lag** at the stating point.

Best Holt-Winter’s model

From the model fitting of three different Holt-Winter’s model above, we can confirm that the Holt-Winter’s model with **exponential** and **damped** (hw3) component is resulted to be the best model based on the MASE score at 0.6689 followed by Holt-Winter’s **damped** component model(hw2).

State Space Models

```
# Pre-defined function for State-space models

models = c("AAA","MAA","MAM","MMM")
dampeds = c(TRUE,FALSE)
expand = expand.grid(models,dampeds)
fit.AICc = array(NA,8)
fit.MASE=array(NA,8)
levels = array(NA, dim=c(8,2))

for(i in 1:6){fit.AICc[i]= ets(Mort_monthly,
                               model = toString(expand[i ,1]),
                               damped = expand[i ,2])$aicc
              fit.MASE[i]= accuracy(ets(Mort_monthly,
                               model = toString(expand[i ,1]),
                               damped = expand[i ,2]))[,6]
levels[i,1]= toString(expand[i ,1])
levels[i,2]= expand[i ,2]
}
# Estimating ANN and ANA separately with damped=True.
fit.AICc[7]= ets(Mort_monthly, model ="ANN")$aicc
fit.AICc[8]= ets(Mort_monthly, model ="ANA")$aicc
fit.MASE[7]= accuracy(ets(Mort_monthly, model ="ANN"))[,6]
fit.MASE[8]= accuracy(ets(Mort_monthly, model ="ANA"))[,6]
levels[7,1]="ANN"
levels[8,1]="ANA"
levels[7,2]=FALSE
levels[8,2]=FALSE
output = data.frame(levels, fit.MASE, fit.AICc)
colnames(output)= c("Model","Damped","MASE","AIC")
output
```

```
##   Model Damped      MASE      AIC
## 1   AAA   TRUE 0.6703678 1370.950
## 2   MAA   TRUE 0.6692330 1363.431
## 3   MAM   TRUE 0.6662357 1362.706
## 4   MMM   TRUE 0.6720527 1361.115
## 5   AAA  FALSE 0.6738379 1368.932
## 6   MAA  FALSE 0.6704884 1360.276
## 7   ANN  FALSE 2.8583676 1619.643
## 8   ANA  FALSE 0.6677170 1363.667
```

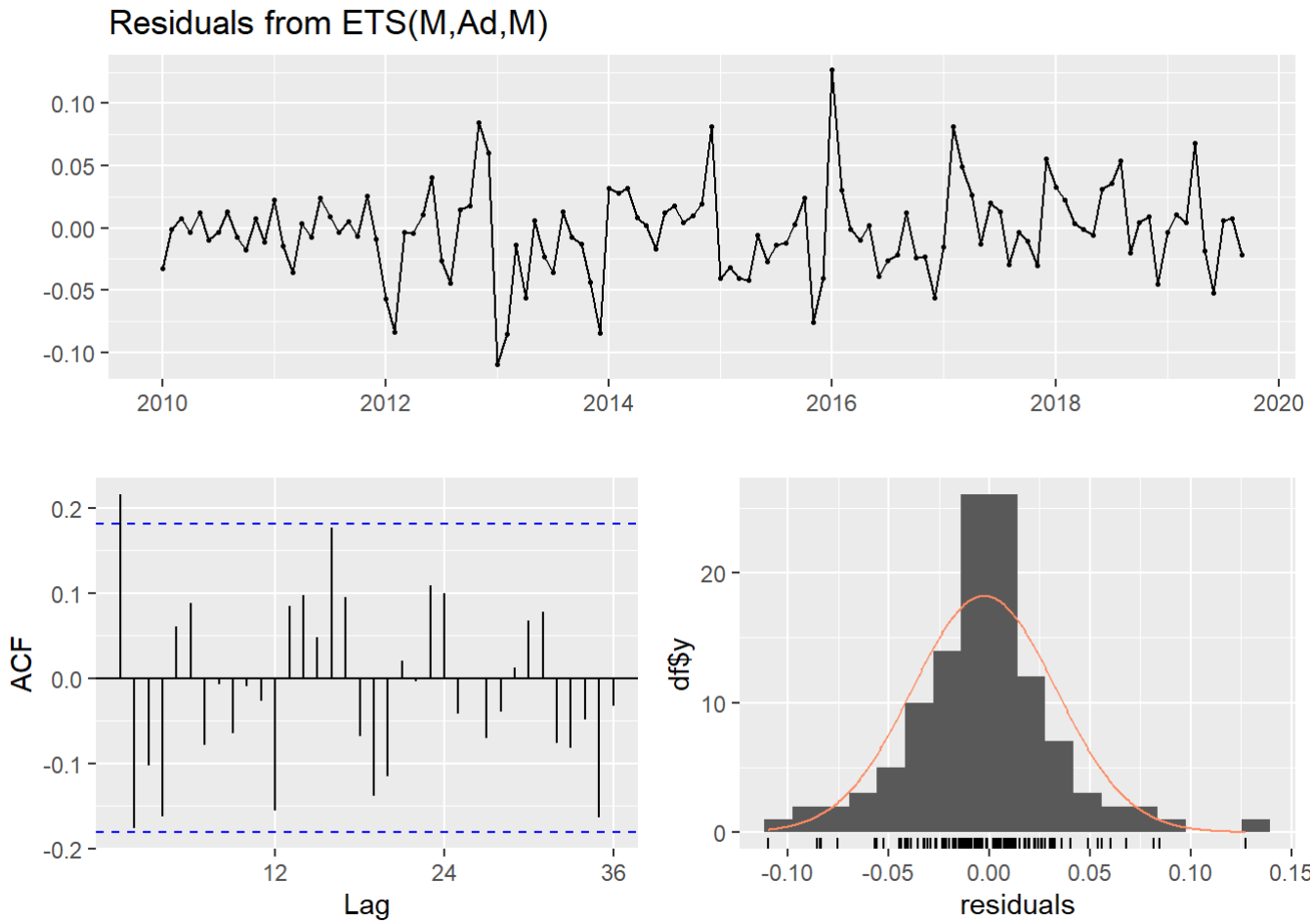
From the above plotted models, the model “MAM” has the least MASE score as 0.666. We will further delve into computing another types of models using the ETS-auto technique.

Automated ETS model

```
fit.auto.mort = ets(Mort_monthly,model="MAM", damped = TRUE)
summary(fit.auto.mort)
```

```
## ETS(M,Ad,M)
##
## Call:
##  ets(y = Mort_monthly, model = "MAM", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.2306
##     beta  = 0.0023
##     gamma = 1e-04
##     phi   = 0.9774
##
##   Initial states:
##     l = 778.2106
##     b = 0.7482
##     s = 1.0246 1.0288 1.1694 0.9064 0.8841 1.0849
##            0.881 0.8806 1.1013 0.9038 0.9233 1.2119
##
##   sigma:  0.0386
##
##      AIC     AICc      BIC
## 1355.726 1362.706 1405.445
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -2.177882 27.47795 19.31576 -0.3921488 2.603957 0.6662357
##                   ACF1
## Training set 0.1892813
```

```
# Checking Residuals
checkresiduals(fit.auto.mort)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,Ad,M)
## Q* = 35.227, df = 23, p-value = 0.04937
##
## Model df: 0.   Total lags used: 23
```

From the outputs of above Model “auto-ETS” and its residuals :

- The critical values of AIC is 1350.160, AICc is 1354.913 and BIC is 1391.593.
- The MASE value was reported at 0.6662357.
- The is No trend observed in the residual time series.
- The p-value from the Ljung-Box test was below **5%** level of significance.
- The histogram seems to be normally distributed overall.
- The ACF plot have one significant **Lag** at the stating point.

Summary Analysis of Chosen models

```
mase_dlm <- rbind(mort_dlm_mase, dynlm_mase)
arrange(mase_dlm, MASE)
```

```
##                   n      MASE
## lm(dynlm8)      506 0.6935111
## lm(dynlm5)      506 0.6944631
## lm(dynlm6)      506 0.6967146
## lm(dynlm7)      506 0.7038379
## lm(dynlm2)      506 0.7277890
## lm(dynlm1)      507 0.7655602
## lm(dynlm3)      507 0.8000698
## ardl_best       507 0.8383696
## koyck_best      507 0.8530742
## finite_best     498 0.8570098
## polynomial_best 498 0.8907160
## lm(dynlm4)      507 0.9199715
```

From the above comparison , we can clearly see that the model "**dynlm8**" is considered to be the best model with respect to the least MASE scores.

Next we will compare and detect the best model from the Holt-Winter's method from Exponential smoothing.

Models from Exponential Smoothing

```
# Based on Accuracy scores for all models with damped and seasonality

arrange(accuracy_hw,MASE)
```

```
##         seasonal Damped      MASE      AIC      BIC
## 1 multiplicative   TRUE 0.6691194 1356.636 1406.356
## 2       additive   TRUE 0.6703678 1363.971 1413.690
## 3       additive  FALSE 0.6738379 1362.750 1409.707
## 4 multiplicative  FALSE 0.6868037 1359.280 1406.237
```

As we can see that the best model from above table is seasonal multiplicative with damped as True with respect to the least MASE score.

Lets further evaluate the accuracy of model "hw3"

```
accuracy(hw3)
```

```
##                     ME    RMSE     MAE         MPE     MAPE      MASE      ACF1
## Training set 0.4102562 27.0577 19.3956 -0.05197752 2.609351 0.6689894 0.2221417
```

So the best model **Holt-Winter's** with **exponential** and **damped** component **hw3** is the best with MASE score as 0.668.

Lastly we will check the models from **ETS** method.

```
# Models from auto-ETS method
arrange(output,MASE)
```

```
##   Model Damped      MASE      AIC
## 1   MAM   TRUE 0.6662357 1362.706
## 2   ANA  FALSE 0.6677170 1363.667
## 3   MAA   TRUE 0.6692330 1363.431
## 4   AAA   TRUE 0.6703678 1370.950
## 5   MAA  FALSE 0.6704884 1360.276
## 6   MMM   TRUE 0.6720527 1361.115
## 7   AAA  FALSE 0.6738379 1368.932
## 8   ANN  FALSE 2.8583676 1619.643
```

From the results of ETS_models, we can clearly see that the model **MAM** is the best with respect to the MASE scores as 0.666.

```
accuracy(fit.auto.mort)
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -2.177882 27.47795 19.31576 -0.3921488 2.603957 0.6662357
##                    ACF1
## Training set 0.1892813
```

Hence, among all the ETS models, the model **fit.auto.mort** or **MAM** turns out to be the best with respect to the MASE scores.

Best Model Slection

We have now successfully done the modelling process and selected the best models from each methods. Among the best ones, we will finally select one best model with respect to the MASE scores.

```
final_model <- data.frame(Model = c("dynlm1 dynamic linear model","Holt-Winter's ES model","M
MM_NA ETS model"), MASE = c(0.6935111,0.6662357,0.6691194))

arrange(final_model, MASE)
```

```
##                         Model      MASE
## 1      Holt-Winter's ES model 0.6662357
## 2            MMM_NA ETS model 0.6691194
## 3 dynlm1 dynamic linear model 0.6935111
```

Based on the outcomes presented, the most suitable model for the monthly mortality data-set is the Holt-Winter’s Exponential Smoothing (ES) model, which incorporates both exponential and damped elements. Conversely, for the weekly dataset, the dynlm1 dynamic linear model is the best fit. Predictions will be made for both the weekly and monthly mortality series using the optimal models determined for each, grounded in metrics such as MASE, R2, and other accuracy indicators.

Forecasting

Forecasting the Next Four Weeks by Holt-Winter’s method

```
# Fitting model

fit.forecast <- hw(Mort_monthly, seasonal = "multiplicative",damped = TRUE, exponential = TRU
E, h=frequency(Mort_monthly))

# Generating forecasts

fitting <- fit.forecast$mean
upper_limit <- fit.forecast$upper[,2]
lower_limit <- fit.forecast$lower[,2]

predict_Mort <- ts.intersect(ts(lower_limit, start = c(2019,1), frequency = 12), ts(fitting,s
tart = c(2019,1), frequency = 12), ts(upper_limit,start = c(2019,1), frequency = 12))

colnames(predict_Mort) <- c("Lower Limit", "Prediction Points", "Upper limit")

# Displaying forecasts

predict_Mort
```

```
##          Lower Limit Prediction Points Upper limit
## Jan 2019    789.5726          854.5077    919.5471
## Feb 2019    690.2801          744.3436    801.5491
## Mar 2019    685.2178          742.5072    799.9825
## Apr 2019    805.3272          873.3763    941.4173
## May 2019    615.7016          667.2073    719.1244
## Jun 2019    602.3574          654.9201    707.6296
## Jul 2019    733.8211          799.9088    866.1175
## Aug 2019    585.2753          637.2720    689.2720
## Sep 2019    582.7033          635.9277    688.6884
## Oct 2019    717.8365          784.3617    849.8397
## Nov 2019    585.4807          640.8454    695.6747
## Dec 2019    600.5720          657.4018    714.6656
```

```
# Load required packages
library(forecast)

# Fitting model
fit.forecast <- hw(Mort_monthly, seasonal = "multiplicative", damped = TRUE, exponential = TR
UE, h=4) # Setting h=1

# Plotting
plot(fit.forecast, fcol = "white", main = "Figure 17: Mortality series with one month forecas
ts using Holt-Winter's model", ylab = "Mortality", xlab = "Time period")
#lines(fitted(fit.forecast), col = "blue")
lines(fit.forecast$mean, col = "blue", lwd = 2)
legend("bottomleft", lty = 1,cex = 0.75, col = c("black", "blue"), c("Data", "Forecasts"))
```

Figure 17: Mortality series with one month forecasts using Holt-Winter's m

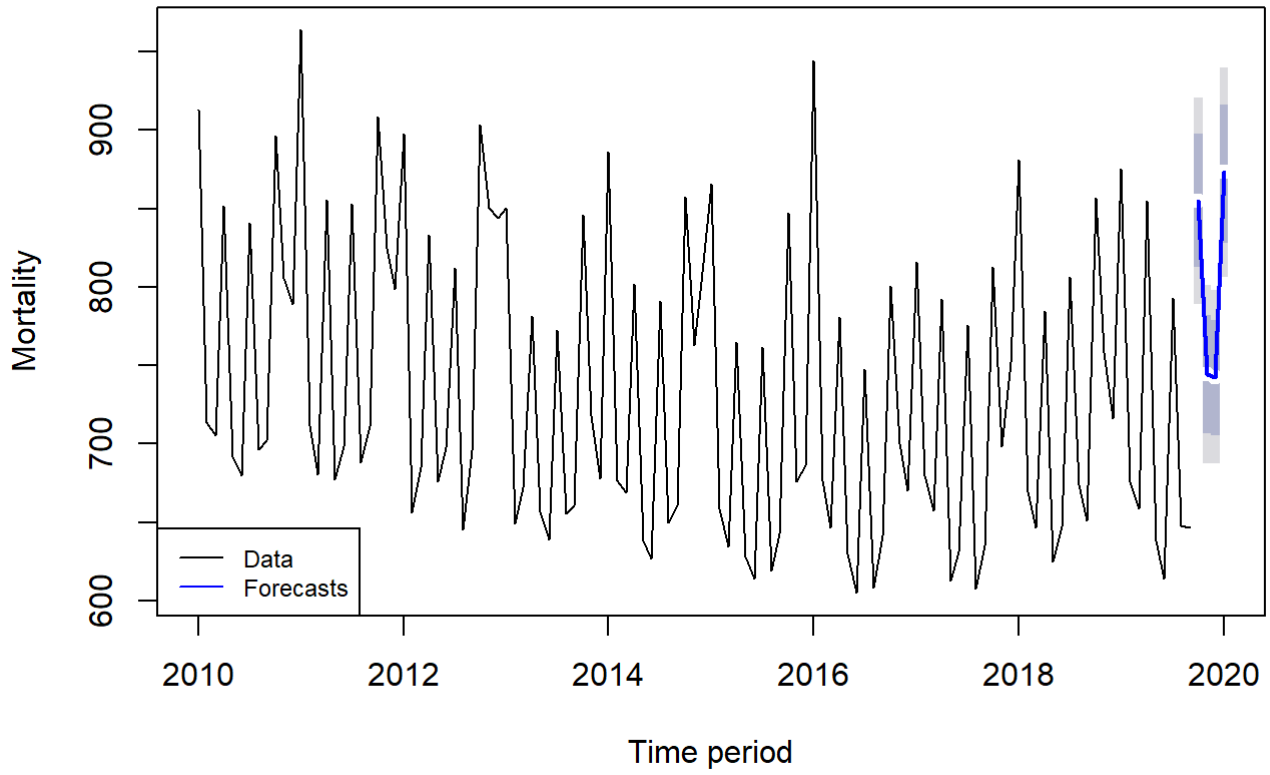


Figure 17

The predictions produced are visually represented in the charts above. The model struggles to fully capture the nuances of the original data and overlooks significant intervention points. While the Damped & Exponential HW Multiplicative model provides forecasts for monthly data (formerly weekly), the accuracy of these predictions seems descent overall.

Forecasting the Next Four Weeks by ETS method

```
forecast_result <- forecast.ets(fit.auto.mort, h=4)
```

```
plot_forecast <- function(forecast_result) {
  p <- autoplot(forecast_result, main="Forecasts from ETS(M,Ad,M)") +
    guides(fill=guide_legend(title="Prediction Interval")) +
    theme_minimal()

  return(p)
}

plot_forecast(forecast_result)
```

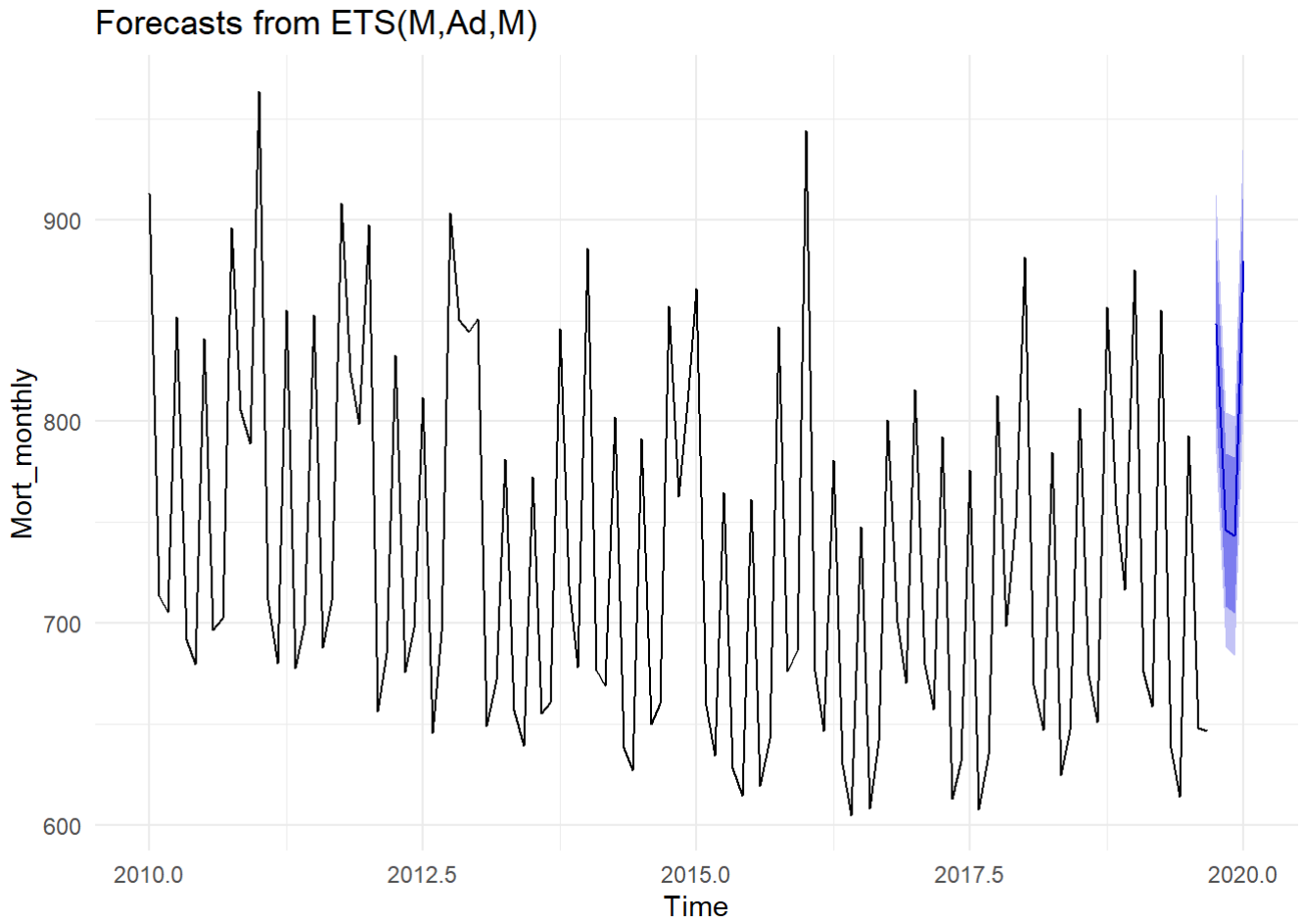


Figure 18

Based on the displayed graph, it suggests that mortality will be on a roller-coaster of highs and lows in the upcoming four weeks, as denoted by the blue line. This forecast seems more trustworthy than the prior model since it aligns more closely with the data.

Task 2 - Time series analysis & forecast of FFD series

Data Description

In a 2021 study, Hudson & Keatley explored the influence of various climatic factors, such as rainfall, temperature, radiation levels, and relative humidity, on the initial bloom date of a species (referred to as the First Flowering Day or FFD, a value ranging from 1-365). Their study concentrated on examining the effects of long-term climate changes on the FFD for 81 different plant species spanning from 1984 to 2014.

The dataset ‘FFD.csv’ provides details on one of these 81 species and includes five distinct time series. This captures the FFD trend for the specified plant species alongside the average annual climate metrics observed from 1984 to 2014, a period covering 31 years. The climatic variables recorded include temperature, rainfall, relative humidity, and radiation levels.

Objectives and Methodology

The primary aim of task 2 is to delve deep into the FFD data and provide a forecast for the upcoming four years. Here’s the approach:

1. **Data Preprocessing**: We'll start by cleaning the data from the 'FFD.csv' file. This involves identifying and handling any missing, unique, or unfamiliar data points. We'll transform each series from the file into a time series format, enabling us to display yearly data values across all series, which is essential for time series analysis.
2. **Covariate File Loading**: We will import the covariate file, which will play a crucial role in the forecasting phase.
3. **Time Series Exploration**: By visualizing individual time series plots and juxtaposing them, we'll gain insights into each series' behavior and characteristics. A key aspect of this phase will be to determine if the series are stationary. Tools like the Autocorrelation Function (ACF), Partial Autocorrelation Function (PACF), and the Dicker-Fuller Unit tests will be instrumental in this analysis.
4. **Correlation Analysis**: To understand the interdependence between the series, we'll generate a correlation matrix. This will show how each series relates to the others.
5. **Modeling with Distributed Lag Models (DLMs)**: Various DLMs will be employed to understand the relationship between the series.
6. **Advanced Modeling**: Depending on the findings from our preliminary analysis, we'll fit several models to the data. These might include DynLMs, Exponential Smoothing methods, and State-Space models.
7. **Forecasting and Model Evaluation**: Using the models we've chosen, we'll produce forecasts. These projections will be evaluated using a range of metrics like R^2, F-test, Akaike's Information Criterion (AIC), and the Bayesian Information Criterion (BIC). The objective is to compare the forecasted values to known data.
8. **Model Selection**: Lastly, based on the various accuracy metrics, we'll identify the most fitting model for our dataset.

By following these steps, we aim to provide a comprehensive analysis and a reliable forecast of the FFD for the next four years.

```
FFD_series <- read_csv("C:/Users/Rakshit Chandna/OneDrive/Desktop/DataMain/Forecasting/FFD.cs
v")
```

```
## Rows: 31 Columns: 6
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (6): Year, Temperature, Rainfall, Radiation, RelHumidity, FFD
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(FFD_series)
```

```
## # A tibble: 6 × 6
##    Year Temperature Rainfall Radiation RelHumidity   FFD
##   <dbl>       <dbl>    <dbl>     <dbl>       <dbl> <dbl>
## 1  1984        18.7     2.49      14.9        54.6   314
## 2  1985        19.3     2.48      14.7        55.0   314
## 3  1986        18.6     2.42      14.5        55.0   320
## 4  1987        19.1     2.32      14.7        53.9   306
## 5  1988        20.4     2.47      14.7        53.1   306
## 6  1989        19.6     2.74      14.8        55.4   314
```

The dataset captures annual environmental measurements spanning various years. For each year, represented as a double (dbl) type, five metrics are documented: Temperature, Rainfall, Radiation, Relative Humidity (all as double types), and FFD. For instance, in 1984, the recorded temperature was approximately 18.71°C, with a rainfall measurement of roughly 2.49 units, radiation at 14.87 units, relative humidity at 54.65%, and an FFD value of 314. Similar data points are presented for subsequent years, providing insights into the yearly environmental changes.

```
task2_covariate <- read_csv("C:/Users/Rakshit Chandna/OneDrive/Desktop/DataMain/Forecasting/C
ovariate x-values for Task 2.csv")
```

```
## Rows: 6 Columns: 5
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (5): Year, Temperature, Rainfall, Radiation, RelHumidity
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
print(task2_covariate)
```

```
## # A tibble: 6 × 5
##    Year Temperature Rainfall Radiation RelHumidity
##   <dbl>       <dbl>    <dbl>     <dbl>       <dbl>
## 1  2015        20.7     2.27      14.6        52.2
## 2  2016        20.5     2.38      14.6        52.9
## 3  2017        20.5     2.26      14.8        52.6
## 4  2018        20.6     2.27      14.8        52.5
## 5    NA          NA       NA        NA          NA
## 6    NA          NA       NA        NA          NA
```

The dataset provides a yearly breakdown from 2015 to 2018 of various environmental metrics. The “Year” column is a double data type, indicating the year of the observation. The “Temperature” column, also a double, gives the average temperature for that year. “Rainfall” provides the annual rainfall measurements, “Radiation” details the average radiation levels, and “RelHumidity” shows the average relative humidity percentage for each respective year. Notably, there are two rows with missing data across all columns at the end of the dataset.

Checking Class

```
# For FFD Series
class(FFD_series)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

```
# For Covariate series
class(task2_covariate)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

Checking Missing Values

```
colSums(is.na(FFD_series))
```

```
##        Year Temperature    Rainfall   Radiation RelHumidity         FFD
##           0           0           0           0           0           0
```

```
colSums(is.na(task2_covariate))
```

```
##        Year Temperature    Rainfall   Radiation RelHumidity
##           2           2           2           2           2
```

Based on the results shown, the FFD_series doesn't have any missing values, but there are two absent values in the task2_covariate series. We'll proceed to eliminate these missing entries from the task2_covariate dataset.

Removing NA values

```
# Removing NA values

task2_covariate <- na.omit(task2_covariate)

# Checking missing values have been removed

head(task2_covariate)
```

```
## # A tibble: 4 × 5
##    Year Temperature Rainfall Radiation RelHumidity
##   <dbl>       <dbl>    <dbl>     <dbl>       <dbl>
## 1  2015        20.7     2.27      14.6        52.2
## 2  2016        20.5     2.38      14.6        52.9
## 3  2017        20.5     2.26      14.8        52.6
## 4  2018        20.6     2.27      14.8        52.5
```

We have now successfully removed the missing values form the covariate dataset.

Converting Data Variables into Time series

```
# Converting 'FFD_series' to time series and storing in 'climate'

climate <- ts(FFD_series[,2:6], start = c(1984,1), frequency = 1)

# Converting Temperature from 'FFD_series' to time series and storing in 'temperature'

temperature <- ts(FFD_series$Temperature, start = c(1984,1), frequency = 1)


# Converting Rainfall from 'FFD_series' to time series and storing in 'rainfall'

rainfall <- ts(FFD_series$Rainfall, start = c(1984,1), frequency = 1)


# Converting Radiation from 'FFD_series' to time series and storing in 'radiation'

radiation <- ts(FFD_series$Radiation, start = c(1984,1), frequency = 1)


# Converting RelHumidity from 'FFD_series' to time series and storing in 'humidity'

humidity <- ts(FFD_series$RelHumidity, start = c(1984,1), frequency = 1)


# Converting FFD from 'FFD_series' to time series and storing in 'FFD'

FFD <- ts(FFD_series$FFD, start = c(1984,1), frequency = 1)
```

Checking class to confirm conversion

```
class(climate)
```

```
## [1] "mts"    "ts"     "matrix" "array"
```

```
cbind(c(class(temperature),class(rainfall),class(humidity),class(radiation),class(FFD)))
```

```
##      [,1]
## [1,] "ts"
## [2,] "ts"
## [3,] "ts"
## [4,] "ts"
## [5,] "ts"
```

The transformation was completed effectively. The FFD_series data was transitioned into a time series format. Each of the five attributes within the FFD_series (excluding Year) was transformed into separate series, each containing yearly time-series data for temperature, radiation, rainfall, humidity, and FFD respectively.

Data Exploration and Visualisation

We will now plot the converted Time Series Data for each of the five series variables and interpret their important characteristics using the **5 Bullet Points**:

- **Trend**
- **Seasonality**
- **Changing Variance**
- **Behavior**
- **Intervention Point**

Time-series plot of FFD series

```
plot(FFD, ylab='FFD', xlab='Time period', type='o', col='blue', main = 'Figure 19: Time-serie
s plot of change in yearly  values of FFD')
```

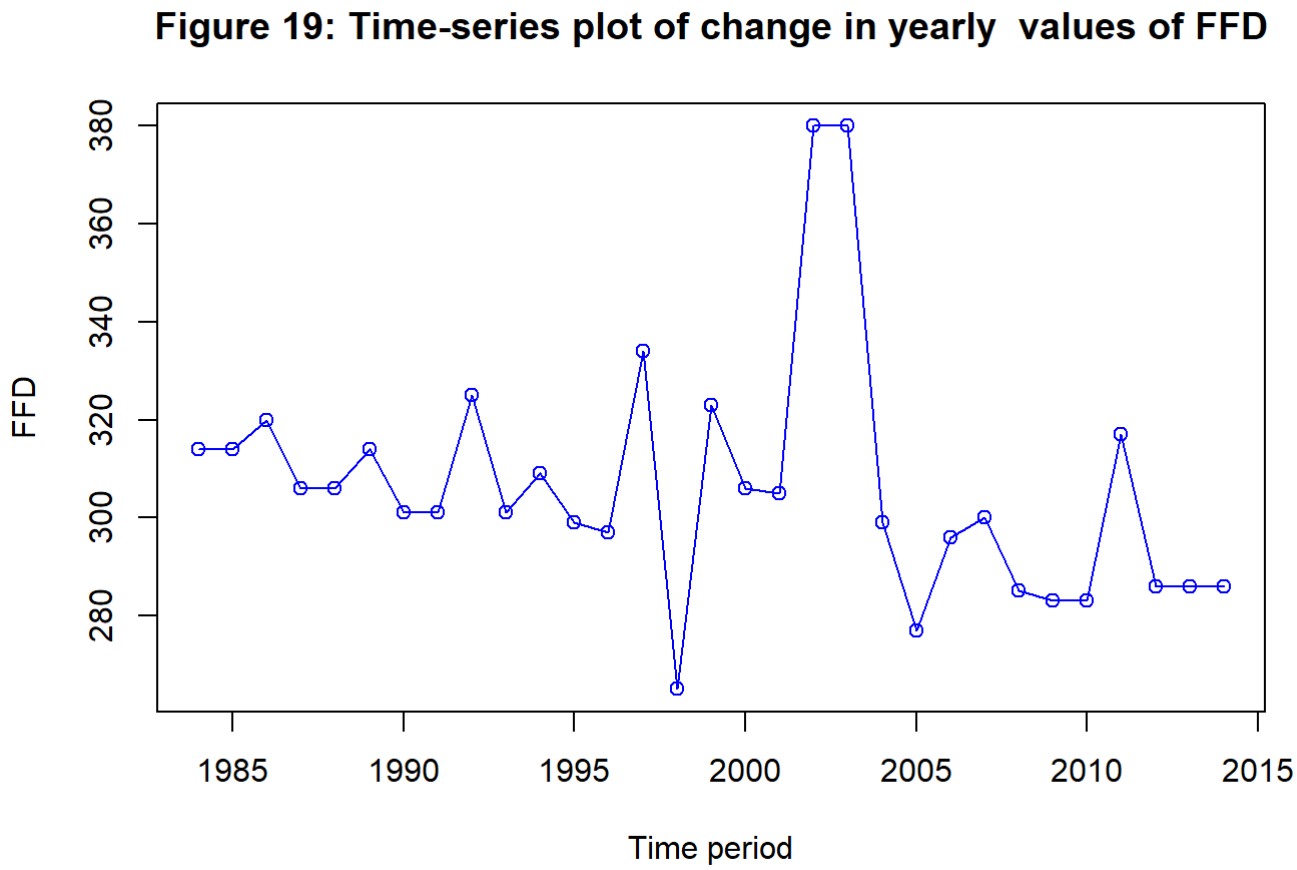


Figure 19

From the above plot, the Time Series plot shows us moderate level of **Seasonality** and **Downward Trend** with successive Auto regressive points and moving average overall from the year 1960 to 2014 respectively.

Checking **5 bullet points**:

- **Trend** - There is a Downward Trend present in the series.
- **Seasonality** - There is very moderate level of repeating patterns (seasonality) can be seen in the graph from year 1960-2014
- **Changing Variance** - There is a significant level of changing variance which is also in repeating patterns, can be seen if we look at the years 1985 and 2003.
- **Behavior** - A mixed Auto regressive and Moving average behavior can be seen in the graph.
- **Intervention point** - There is a sudden peak point in year 2003 observed in the series.

Time-series plot of Temperature series

```
plot(temperature, ylab='Temperature', xlab='Time period', type='o', col='red', main = 'Figure 20: Time-series plot of change in yearly  values of temperature')
```

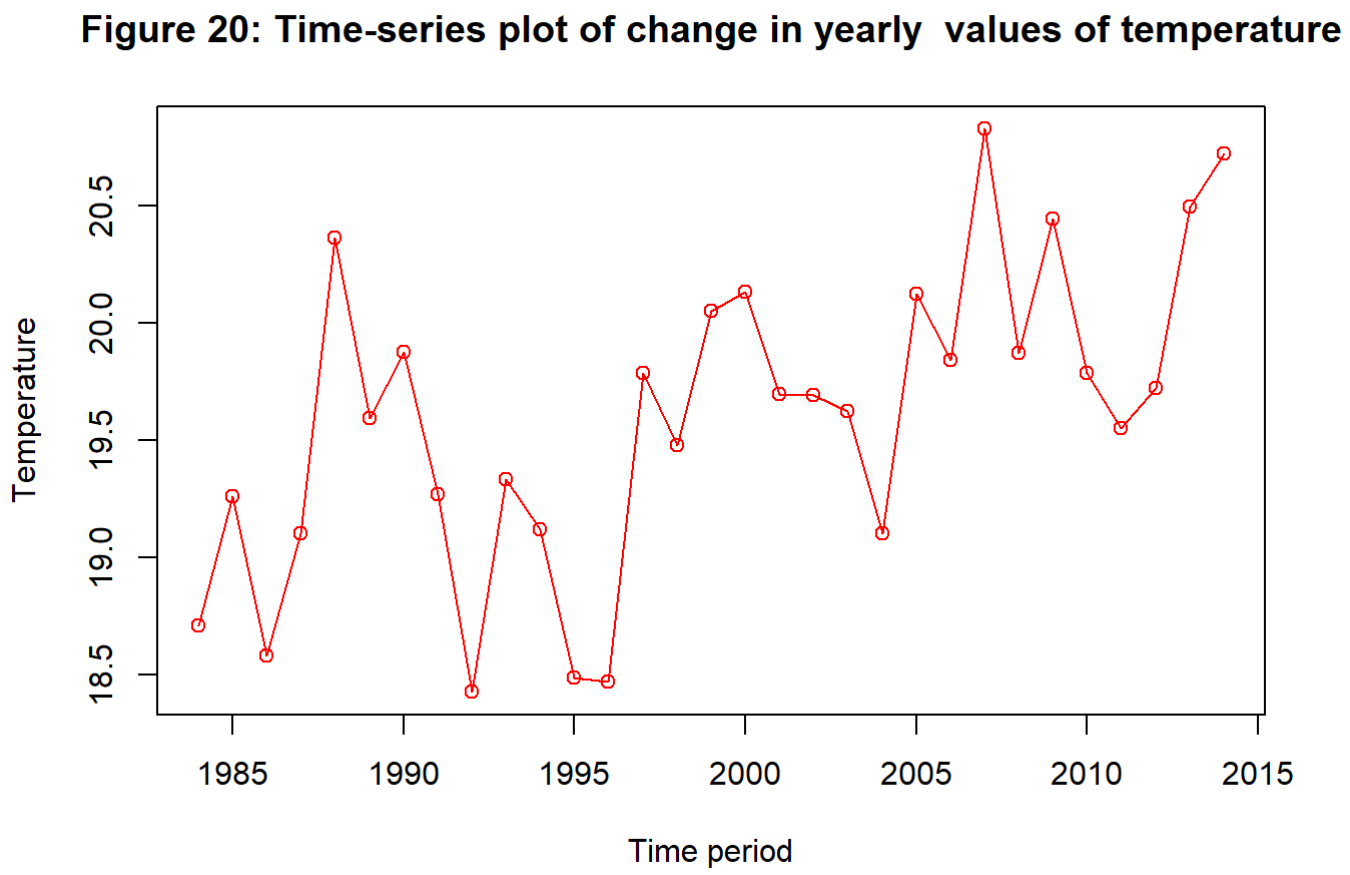


Figure 20

From the above plot, the Time Series plot shows us moderate level of **Seasonality** and **Upward Trend** with successive Auto regressive points and moving average overall from the year 1960 to 2014 respectively.

Checking **5 bullet points**:

- **Trend** - There is a Upward Trend present in the series.
- **Seasonality** - There is very moderate level of repeating patterns (seasonality) can be seen in the graph from year 1960-2014
- **Changing Variance** - There is a significant level of changing variance which is also in repeating patterns, can be seen if we look at the years 1985 and 2005.
- **Behavior** - A mixed Auto regressive and Moving average behavior can be seen in the graph.
- **Intervention point** - There is a sudden change point in year 1992 observed in the series.

Time-series plot of Rainfall series

```
plot(rainfall, ylab='Rainfall', xlab='Time period', type='o', col='green', main = 'Figure 21: Time-series plot of change in yearly  values of rainfall')
```

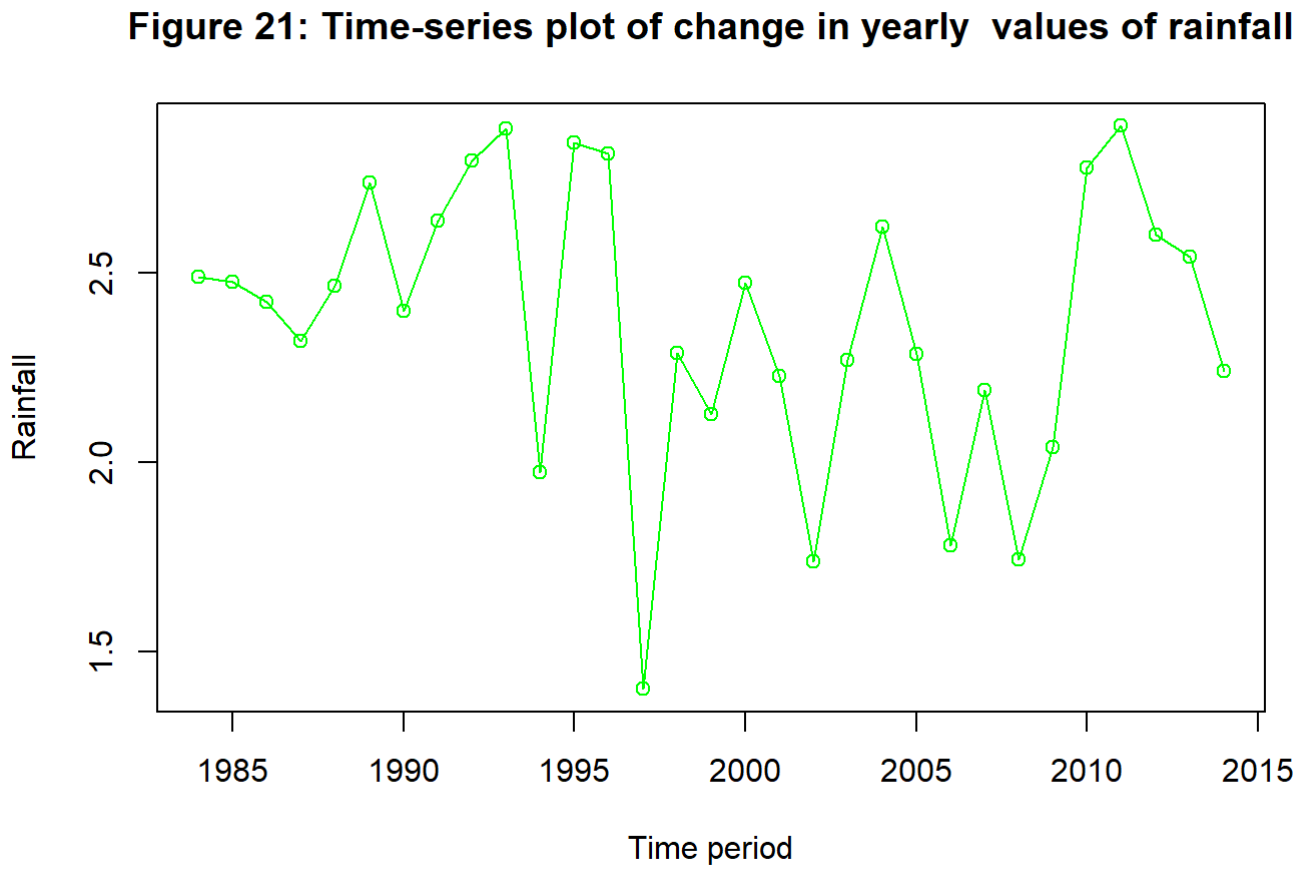


Figure 21

From the above plot, the Time Series plot shows us moderate level of **Seasonality** with successive Auto regressive points and moving average overall from the year 1960 to 2014 respectively.

Checking **5 bullet points**:

- **Trend** - There is a nearly No Trend present in the series.
- **Seasonality** - There is very high level of repeating patterns (seasonality) can be seen in the graph from year 1960-2014
- **Changing Variance** - There is a significant level of changing variance which is also in repeating patterns, can be seen if we look at the years 1995 and 2005.
- **Behavior** - A mixed Auto regressive and Moving average behavior can be seen in the graph.
- **Intervention point** - There is a sudden change point in year 1996 observed in the series.

Time-series plot of Radiation series

```
plot(radiation, ylab='Radiation', xlab='Time period', type='o', col='darkmagenta', main = 'Fi
gure 22: Time-series plot of change in yearly  values of radiation')
```

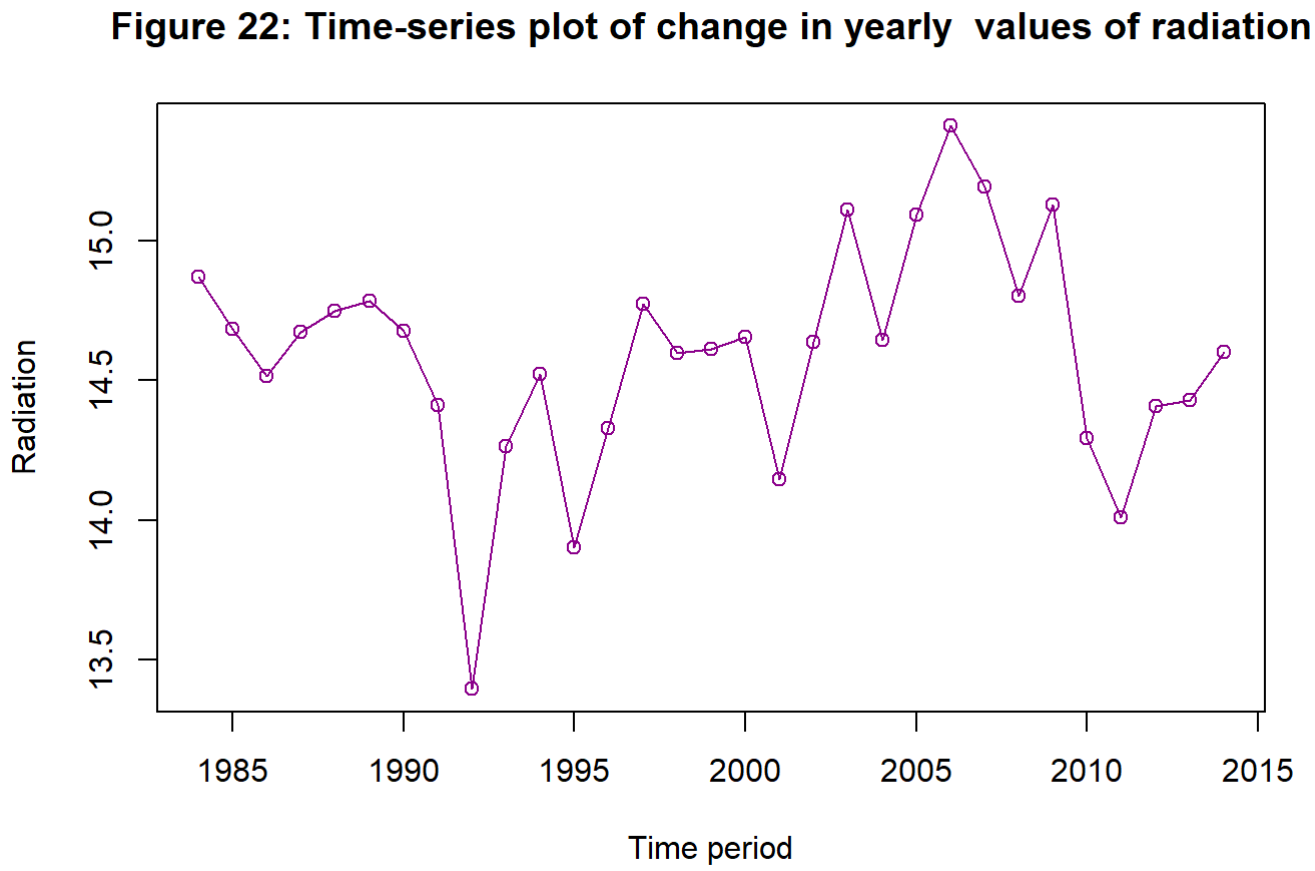


Figure 22

From the above plot, the Time Series plot shows us moderate level of **Seasonality** with successive Auto regressive points and moving average overall from the year 1960 to 2014 respectively.

Checking **5 bullet points**:

- **Trend** - There is a nearly No Trend present in the series.
- **Seasonality** - There is moderate level of repeating patterns (seasonality) can be seen in the graph from year 1960-2014
- **Changing Variance** - There is a significant level of changing variance which is also in repeating patterns, can be seen if we look at the years 1992 and 2007.
- **Behavior** - A mixed Auto regressive and Moving average behavior can be seen in the graph.
- **Intervention point** - There is a sudden change point in year 1992 observed in the series.

Time-series plot of Humidity series

```
plot(humidity, ylab='Humidity', xlab='Time period', type='o', col='black', main = 'Figure 23:
Time-series plot of change in yearly  values of humidity')
```

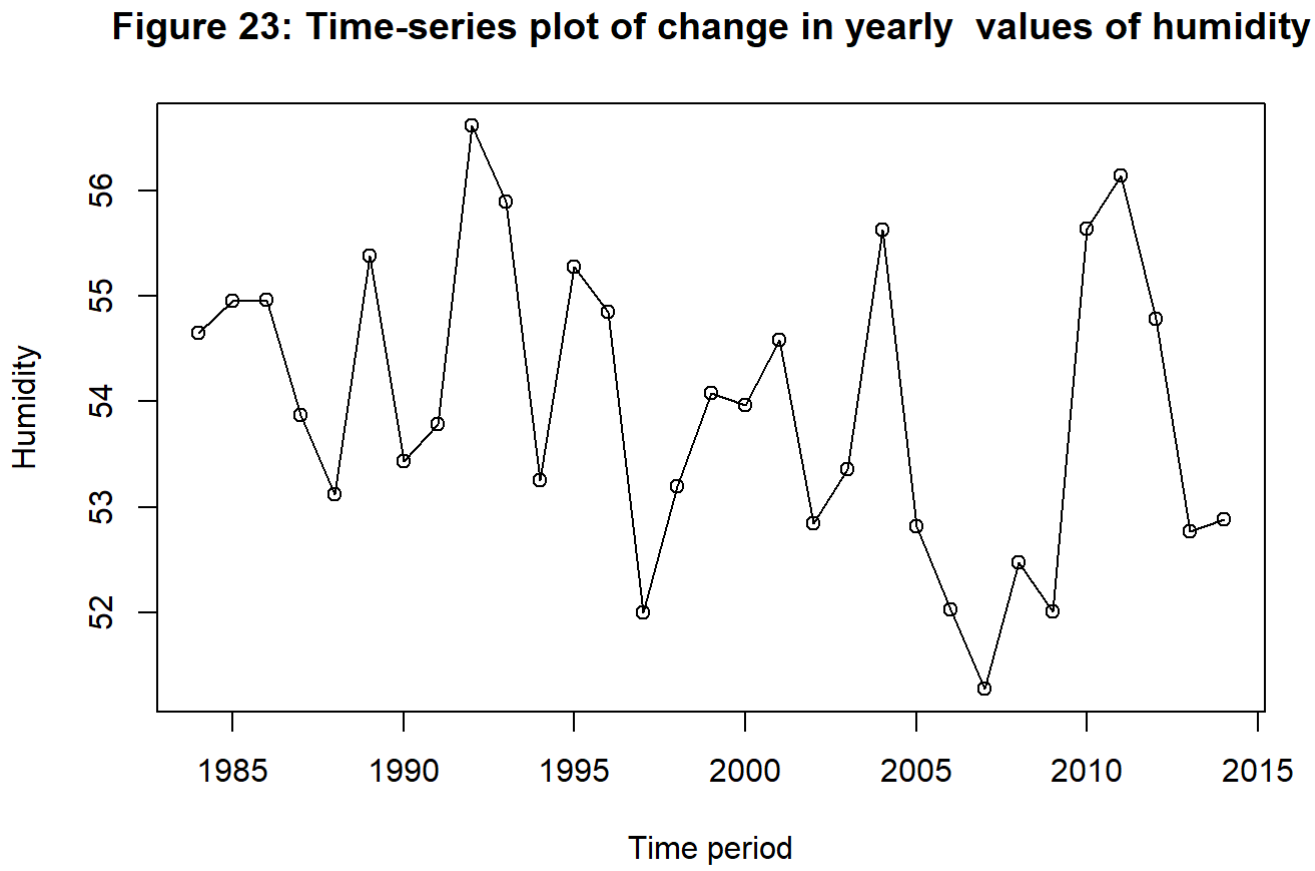


Figure 23

From the above plot, the Time Series plot shows us high level of **Seasonality** and **Downward Trend** with successive Auto regressive points and moving average overall from the year 1960 to 2014 respectively.

Checking **5 bullet points**:

- **Trend** - There is a Downward Trend present in the series.
- **Seasonality** - There is moderate level of repeating patterns (seasonality) can be seen in the graph from year 1960-2014
- **Changing Variance** - There is a significant level of changing variance which is also in repeating patterns, can be seen if we look at the years 1987 and 2011.
- **Behavior** - A mixed Auto regressive and Moving average behavior can be seen in the graph.
- **Intervention point** - There is a sudden change point in year 2007 observed in the series.

Scaled Time-series plot of data set Climate

```
# Scaling climate data

scaled2  = scale(climate)

# Plotting time-series plot containing all the five series together

plot(scaled2, plot.type = "s", col = c("blue", "red","green","darkmagenta","black"), main =
"Figure 24: Time Series plot of  scaled climate",xlab="Time period")
legend("topleft",lty=1,cex=0.65, text.width = 5, col=c("blue","red", "green", "darkmagent
a","black"), c("FFD", "Temperature", "Rainfall", "Radiation","Humidity"))
```

Figure 24: Time Series plot of scaled climate

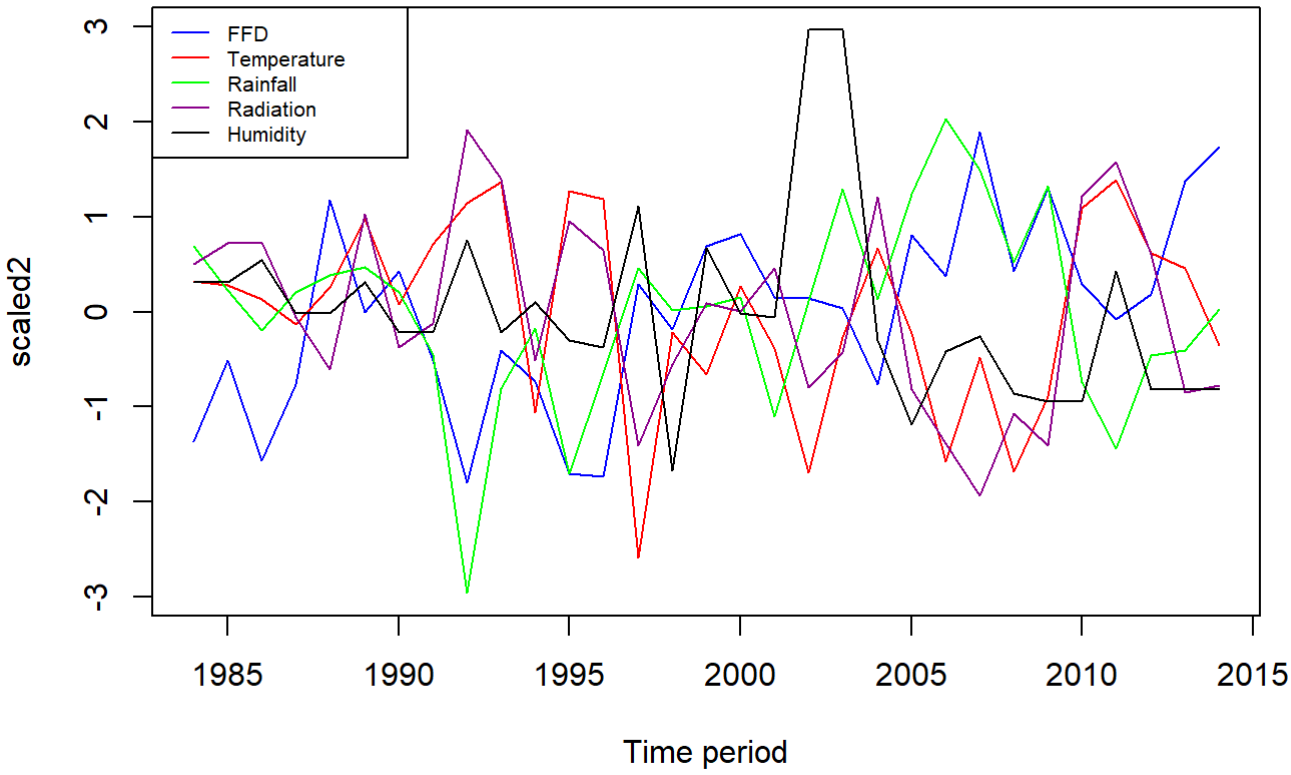


Figure 24

Each of the five series appears to be interconnected and doesn't exhibit any seasonal tendencies. To further validate these visual observations, we'll proceed to compute the correlation among these series mathematically.

Estimating correlation for climate series

```
flattenCorrMatrix <- function(cormat, pmat) {
  ut <- upper.tri(cormat)
  data.frame(
    row = rownames(cormat)[row(cormat)[ut]],
    column = rownames(cormat)[col(cormat)[ut]],
    cor  =(cormat)[ut],
    p = pmat[ut]
    )
}

# Calculating correlation

res1<-rcorr(as.matrix(climate))
flattenCorrMatrix(res1$r, res1$P)
```

```
##            row      column          cor            p
## 1  Temperature    Rainfall -0.391507194 2.940424e-02
## 2  Temperature   Radiation  0.519357599 2.753070e-03
## 3     Rainfall   Radiation -0.581316101 6.046040e-04
## 4  Temperature RelHumidity -0.662197976 4.955000e-05
## 5     Rainfall RelHumidity  0.791107864 1.174543e-07
## 6    Radiation RelHumidity -0.735408669 2.443007e-06
## 7  Temperature         FFD -0.198447254 2.845333e-01
## 8     Rainfall         FFD -0.220347796 2.335997e-01
## 9    Radiation         FFD  0.003593172 9.846945e-01
## 10 RelHumidity         FFD  0.060176754 7.477762e-01
```

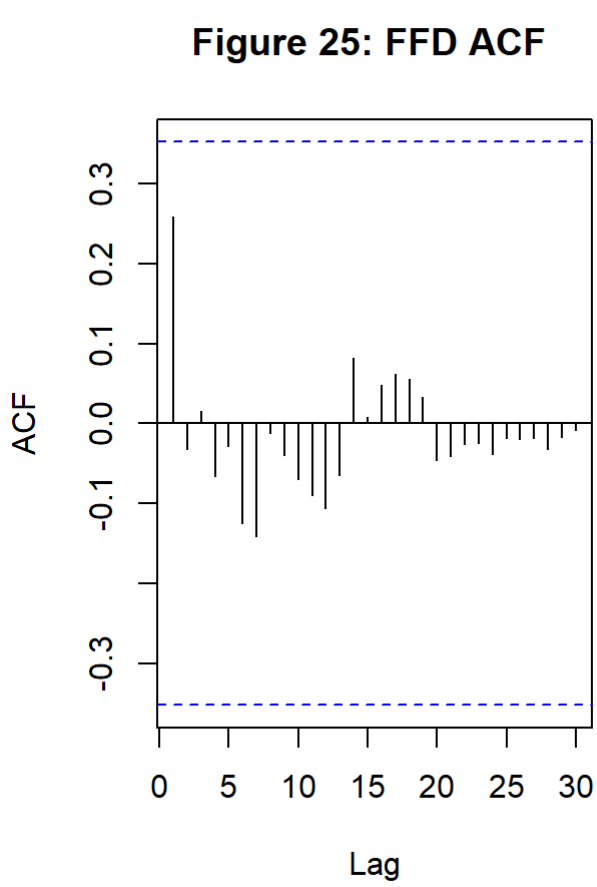
From the findings presented, it's evident that there is a notable negative correlation between rainfall and radiation, registering at -0.58. Similarly, there's a significant positive correlation of 0.52 between temperature and radiation. However, other variables don't show any substantial correlation with FFD. These observations imply that **univariate** models would be more suitable for our predictions, given the lack of substantial inter-variable correlations to account for when forecasting FFD.

Check for Non-stationarity

Plotting ACF and PACF for FFD series

```
# Plotting ACF and PACF

par(mfrow=c(1,2))
acf(FFD, lag.max = 48, main = "Figure 25: FFD ACF ")
pacf(FFD, lag.max = 48, main = "Figure 25: FFD PACF")
```



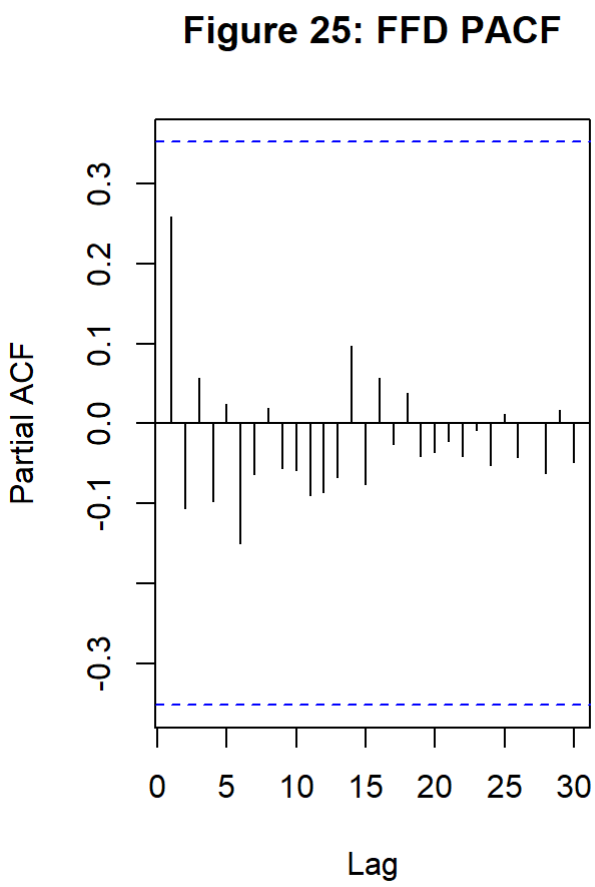
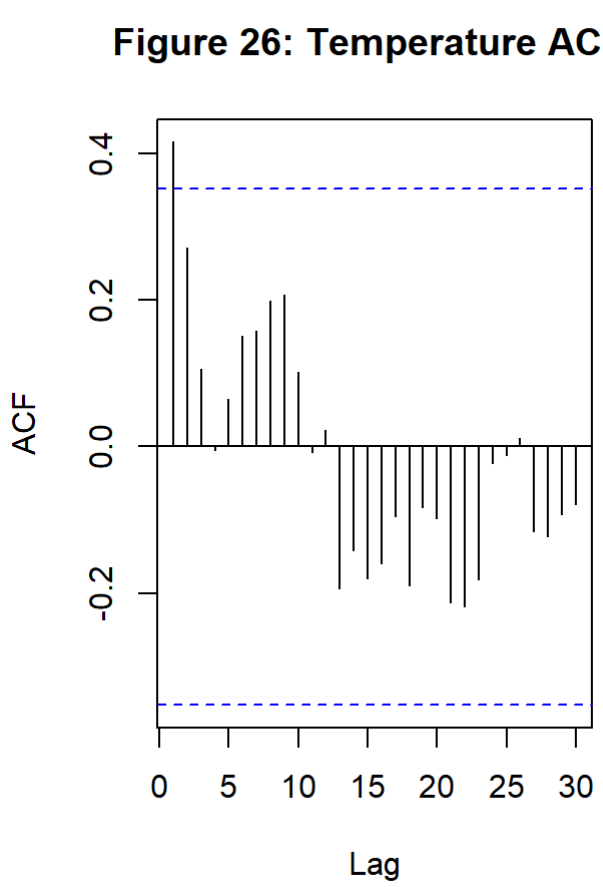


Figure 25

In the ACF plot, we observe a slowly decaying pattern of both positive and negative lags, indicating a possible trend. Meanwhile, the PACF graph shows just one significant lag, implying that the FFD series might not be stationary.

Plotting ACF and PACF for Temperature

```
par(mfrow=c(1,2))
acf(temperature, lag.max = 48, main = "Figure 26: Temperature ACF ")
pacf(temperature, lag.max = 48, main = "Figure 26: Temperature PACF")
```



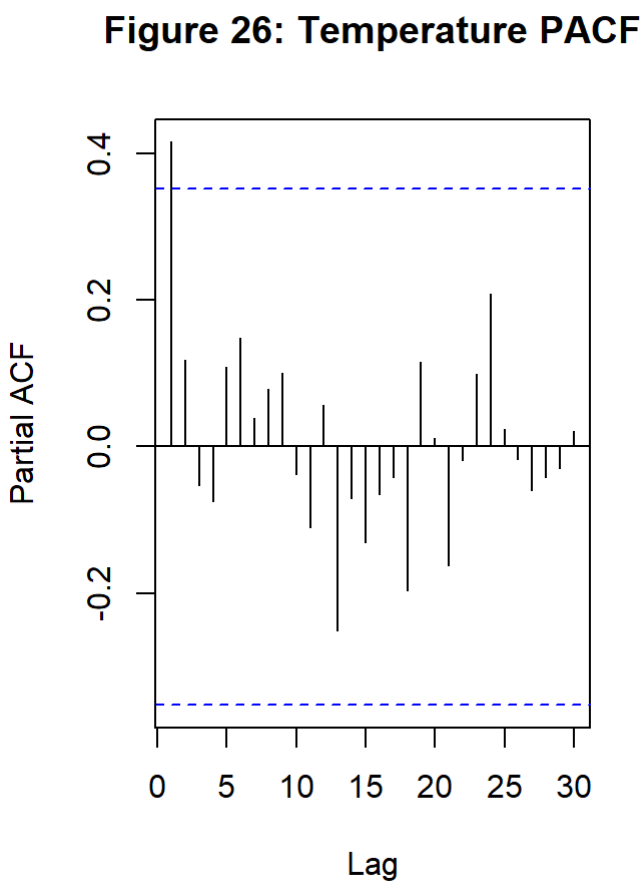
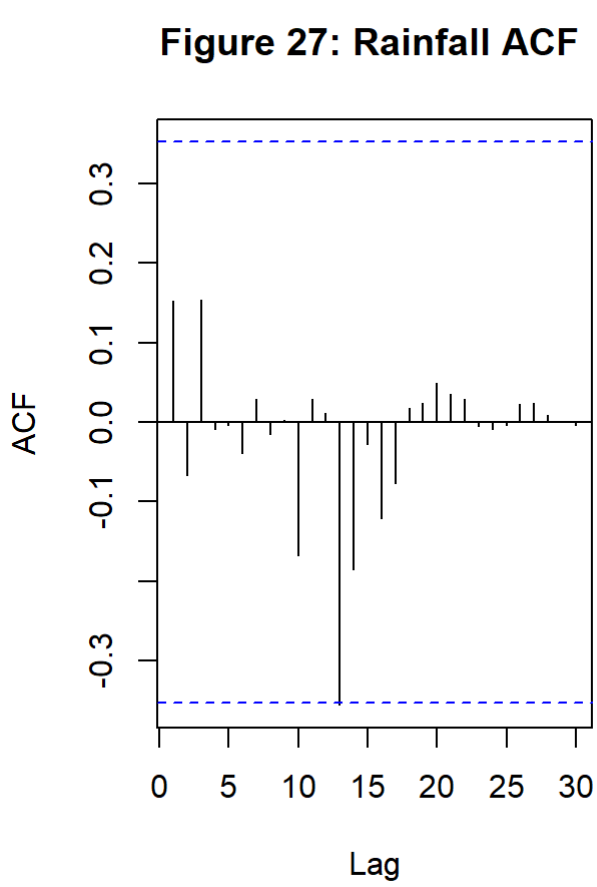


Figure 26

From the ACF plot, we observe a slowly decaying pattern of both positive and negative lags, indicating a possible trend. Meanwhile, the PACF graph shows just one significant lag, implying that the Temperature series could be non-stationary.

Plotting ACF and PACF for Rainfall

```
par(mfrow=c(1,2))
acf(rainfall, lag.max = 48, main = "Figure 27: Rainfall ACF ")
pacf(rainfall, lag.max = 48, main = "Figure 27: Rainfall PACF")
```



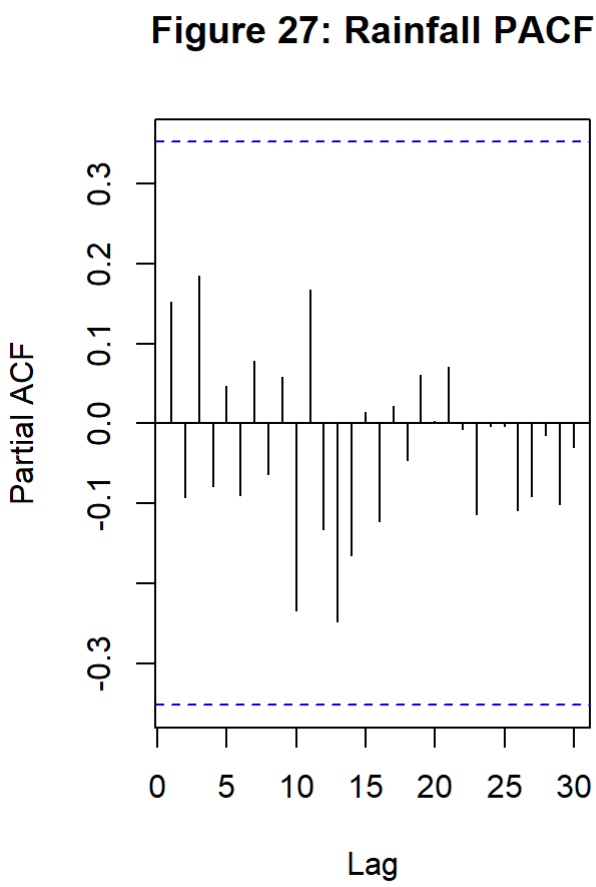
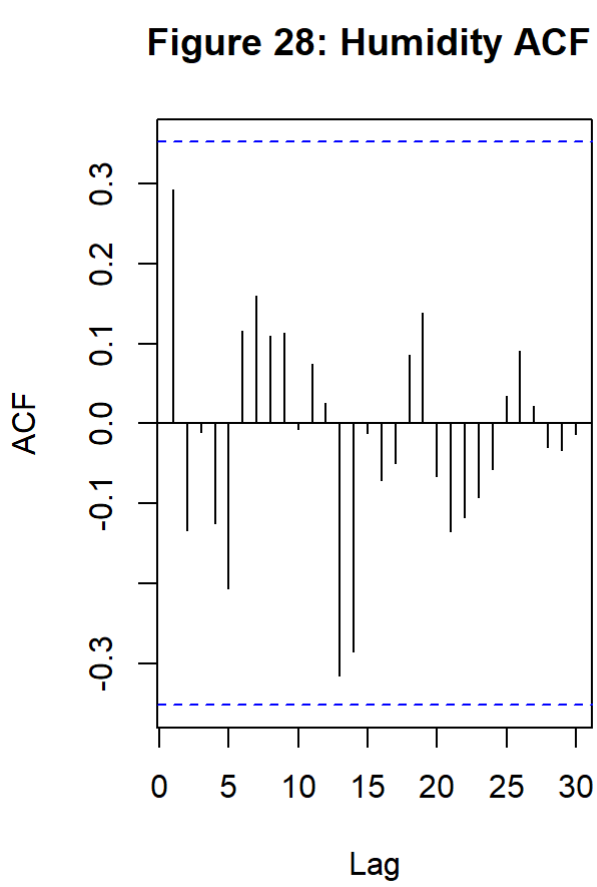


Figure 27

From the above ACF plot, there is not any slowly decaying pattern while in PACF, no lag is significant which suggests series is stationary.

Plotting ACF and PACF for Humidity

```
par(mfrow=c(1,2))
acf(humidity, lag.max = 48, main = "Figure 28: Humidity ACF ")
pacf(humidity, lag.max = 48, main = "Figure 28: Humidity PACF")
```



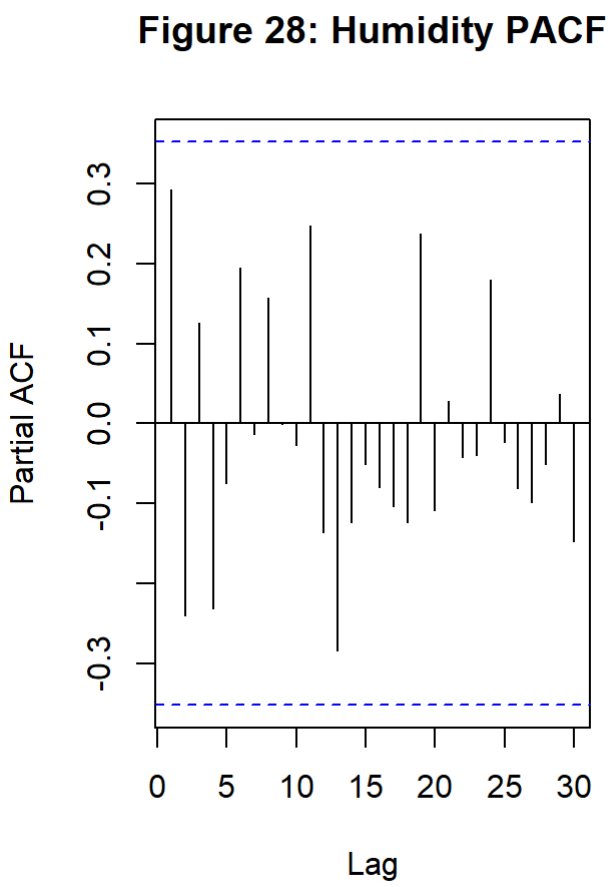
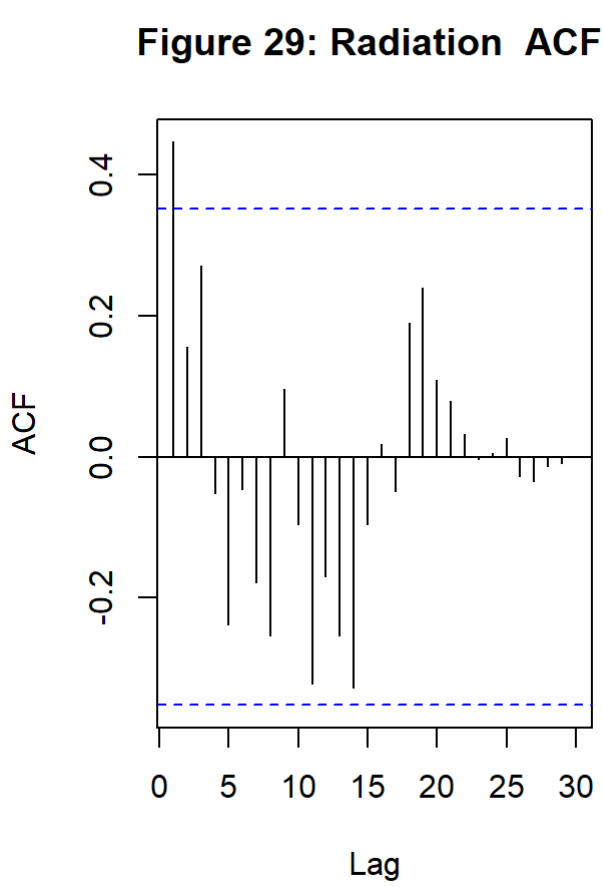


Figure 28

From the above ACF plot, there is not any slowly decaying pattern while in ACF, no first lag is significant in PACF which suggests series is stationary.

Plotting ACF and PACF for Radiation

```
par(mfrow=c(1,2))
acf(radiation, lag.max = 48, main = "Figure 29: Radiation  ACF ")
pacf(radiation, lag.max = 48, main = "Figure 29: Radiation  PACF")
```



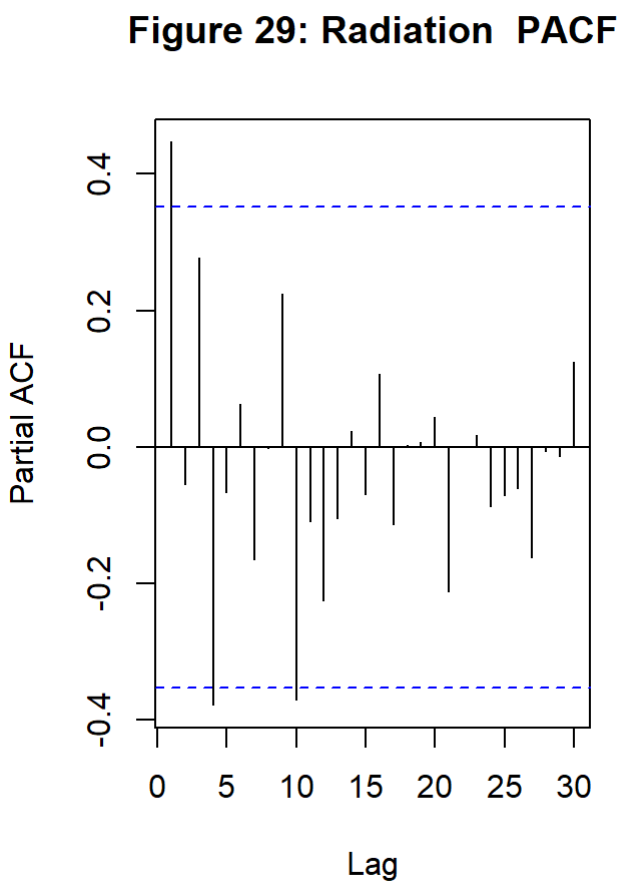


Figure 29

From the above ACF plot, there is a any slowly decaying pattern while in ACF and first lag is significant in PACF which suggests series is non-stationary.

From the ACF and PACF diagrams, we observe that the FFD, temperature, and radiation series are not stationary. In contrast, the rainfall and humidity series appear to be stationary. To confirm these observations, we will perform ADF tests in the subsequent steps.

Performing Stationarity Check

Utilizing Augmented Dickey-Fuller Test

ADF test on FFD series

```
adf.test(FFD, k=ar(FFD)$order)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  FFD
## Dickey-Fuller = -3.53, Lag order = 1, p-value = 0.0571
## alternative hypothesis: stationary
```

From the above ADF test of FFD series, we can observe that the p-values is grater than 5% level of significance indicating that the series is Non-stationary.

ADF test on Temperature series

```
adf.test(temperature, k=ar(temperature)$order)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  temperature
## Dickey-Fuller = -2.9938, Lag order = 1, p-value = 0.1902
## alternative hypothesis: stationary
```

From the above ADF test of Temperature series, we can observe that the p-values is grater than 5% level of significance indicating that the series is Non-stationary.

ADF test on Rainfall series

```
adf.test(rainfall, k=ar(rainfall)$order)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  rainfall
## Dickey-Fuller = -4.5622, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

From the above ADF test of Temperature series, we can observe that the p-values is less than 5% level of significance indicating that the series is Stationary.

ADF test on Humidity series

```
adf.test(humidity, k=ar(humidity)$order)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  humidity
## Dickey-Fuller = -4.1163, Lag order = 1, p-value = 0.01789
## alternative hypothesis: stationary
```

From the above ADF test of Humidity series, we can observe that the p-values is less than 5% level of significance indicating that the series is Stationary.

ADF test on Radiation series

```
adf.test(radiation, k=ar(radiation)$order)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  radiation
## Dickey-Fuller = -2.7317, Lag order = 4, p-value = 0.2911
## alternative hypothesis: stationary
```

From the above ADF test of Radiation series, we can observe that the p-values is grater than 5% level of significance indicating that the series is Non-stationary.

Transformation of Non-Stationary series

Based on earlier findings, FFD, temperature, and radiation are non-stationary series. To advance with the analysis, these series need to be transformed into stationary ones. Therefore, we will apply successive orders of **differencing**, such as first, second, third, and so on, to each of these series until they achieve stationarity.

Transformation of FFD series

Applying first order differencing on FFD series

```
FFDdiff = diff(FFD)
plot(FFDdiff,ylab='FFD',xlab='Time period', col="blue", main = "Figure 30: Time series plot o
f first differenced FFD")
```

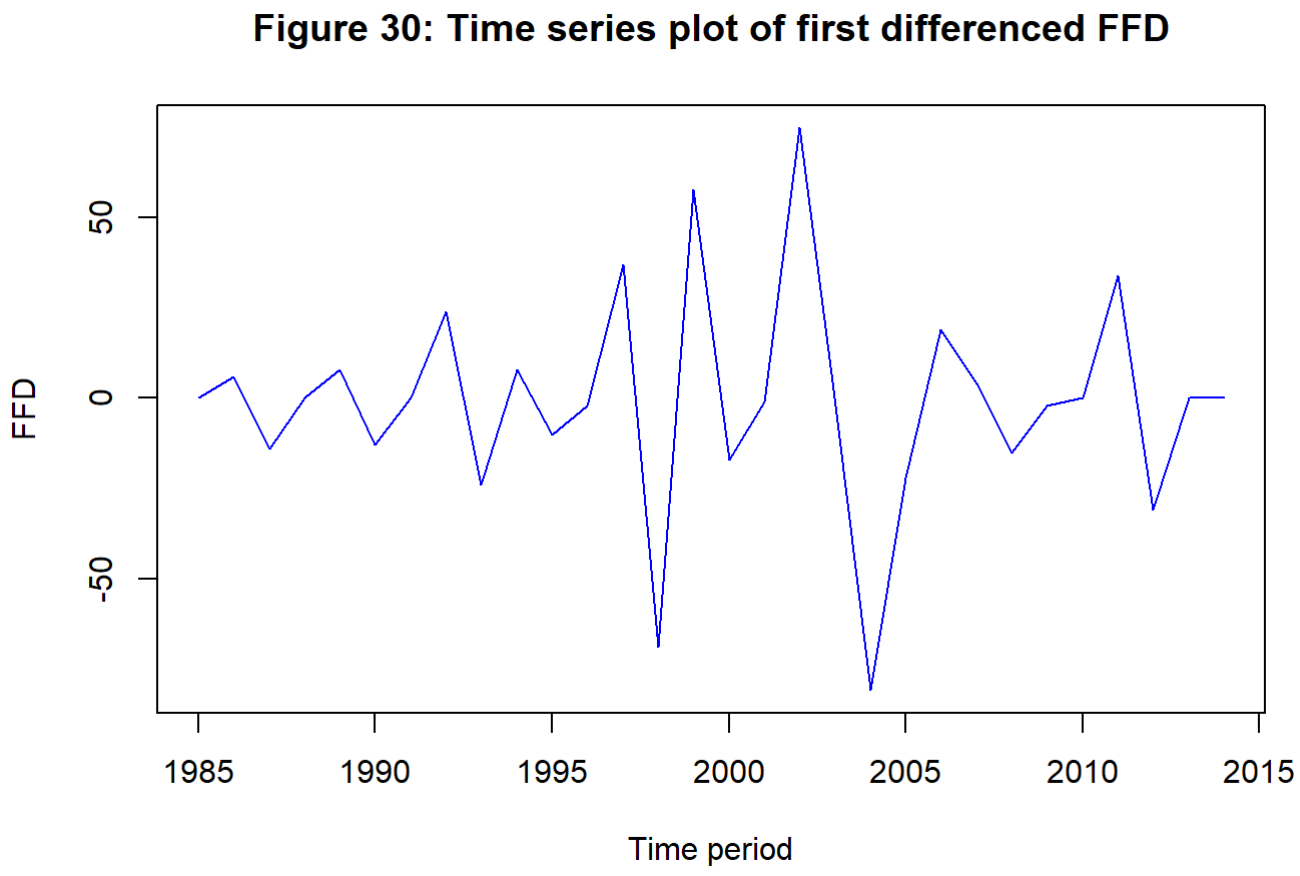


Figure 30

After applying the first difference to the FFD series, it appears to be stationary. To validate this observation, we'll employ the ADF test on the series.

ADF Test on first order differenced FFD

```
adf.test(FFDdiff)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  FFDdiff
## Dickey-Fuller = -3.816, Lag order = 3, p-value = 0.03304
## alternative hypothesis: stationary
```

We can observe that the series is now completely Stationary as the p-values is less than 5% level of significance.

Transformation of Temperature series

Applying first order differencing on Temperature series

```
temperaturediff = diff(temperature)
plot(temperaturediff,ylab='Temperature',xlab='Time period', col="red", main = "Figure 31: Tim
e series plot of first differenced temperature series")
```

Figure 31: Time series plot of first differenced temperature series

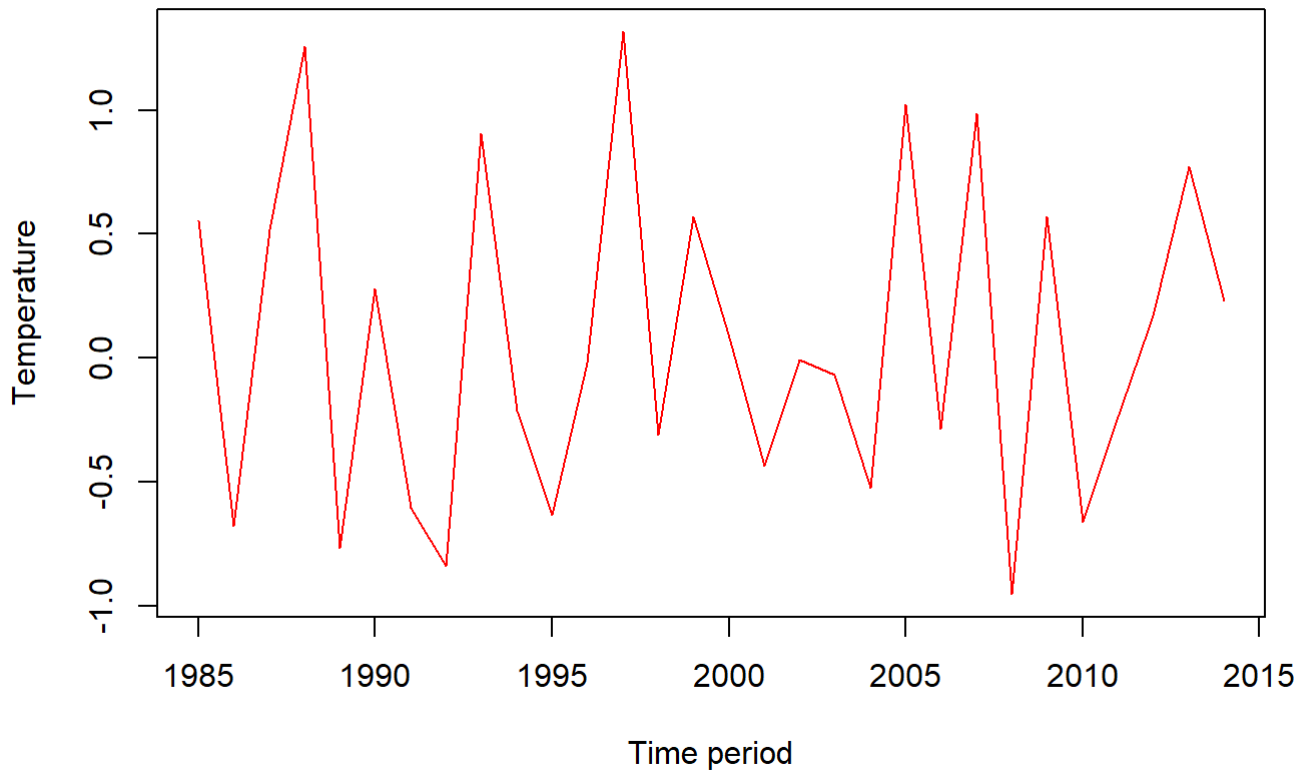


Figure 31

After applying the first difference to the Temperature series, it appears to be stationary. To validate this observation, we'll employ the ADF test on the series.

ADF Test on first order differenced Temperature

```
adf.test(temperaturediff)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  temperaturediff
## Dickey-Fuller = -3.4245, Lag order = 3, p-value = 0.07255
## alternative hypothesis: stationary
```

The above results show, there was significant change in the temperature series but its still not completely stationary. Let's apply second order differencing to it.

Applying second order differencing on Temperature series

```
temperaturediff2 = diff(temperaturediff)
plot(temperaturediff2,ylab='Temperature',xlab='Time period', col="red", main = "Figure 32: Ti
me series plot of second differenced temperature series")
```

Figure 32: Time series plot of second differenced temperature series

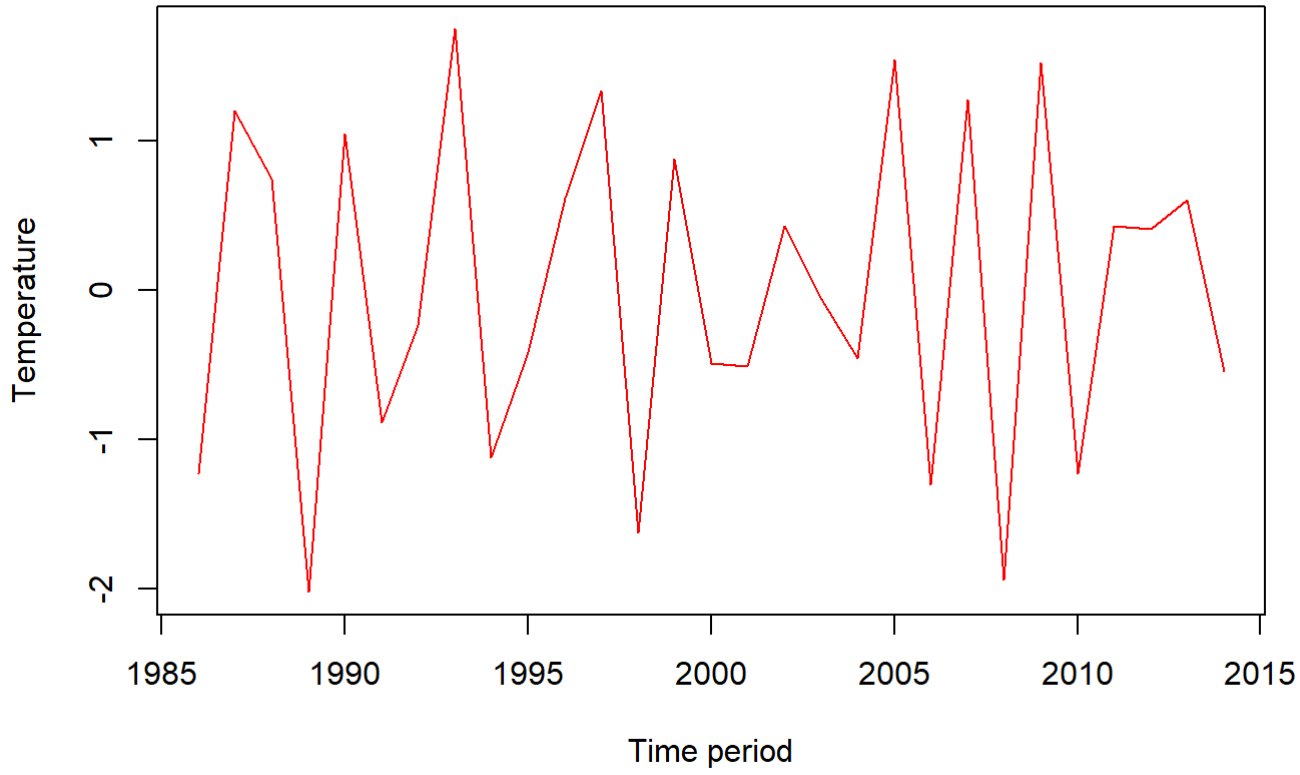


Figure 32

ADF Test on second order differenced Temperature series

```
adf.test(temperaturediff2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  temperaturediff2
## Dickey-Fuller = -3.2642, Lag order = 3, p-value = 0.09558
## alternative hypothesis: stationary
```

The series is still not stationary. Let's apply third order differencing and check it again

Applying thord order differencing on Temperature series

```
temperaturediff3 = diff(temperaturediff2)
plot(temperaturediff3,ylab='Temperature',xlab='Time period', col="red", main = "Figure 33: Ti
me series plot of third order differenced temperature series")
```

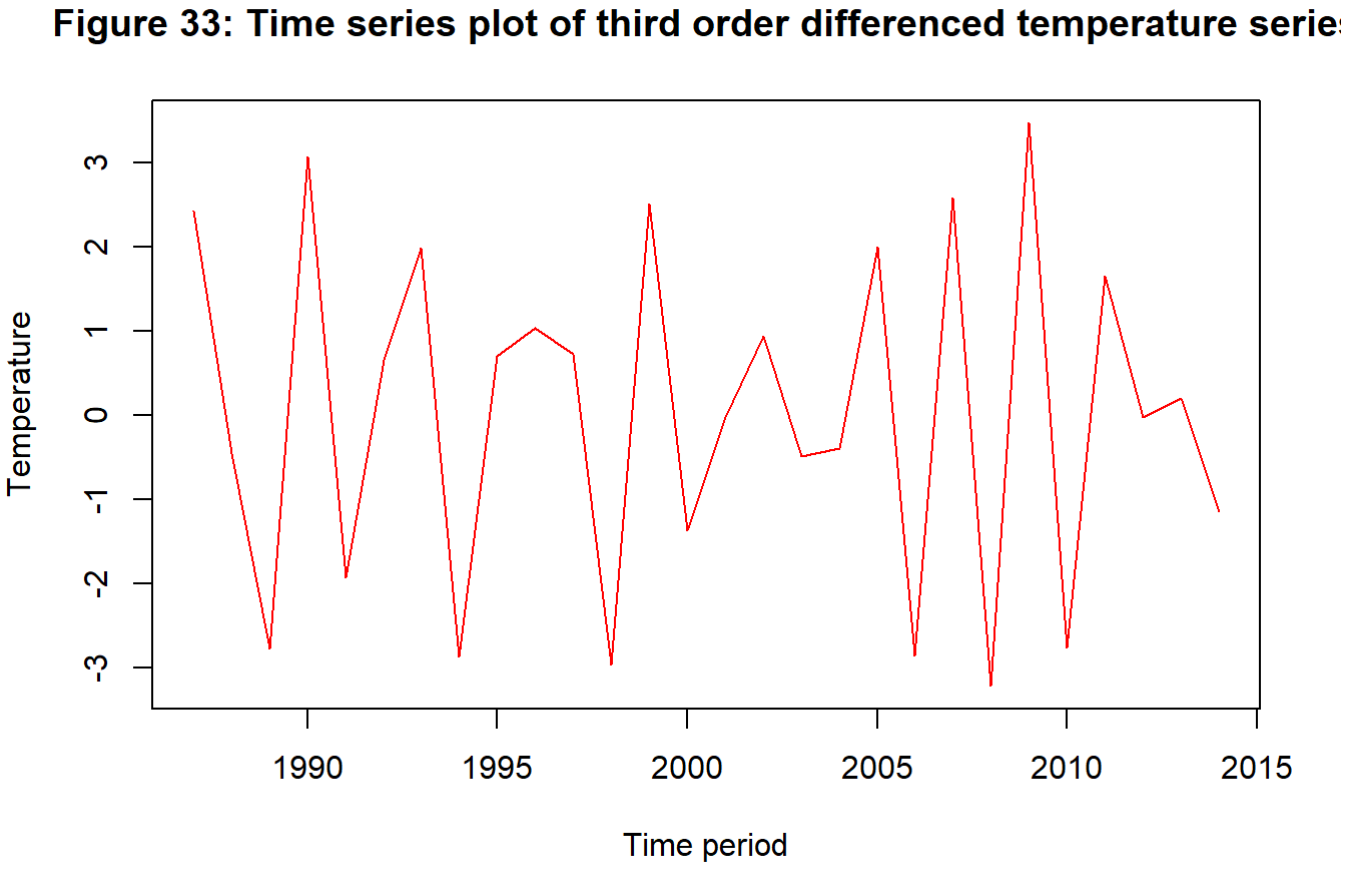


Figure 33

The series has changed from the second differenced series especially from the initial & end points. Let's check whether it has achieved stationarity or not.

Performing ADF test on third differenced temperature series

```
adf.test(temperaturediff3)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  temperaturediff3
## Dickey-Fuller = -4.0851, Lag order = 3, p-value = 0.01979
## alternative hypothesis: stationary
```

The temperature series achieved stationarity after differencing it three times, as indicated by a p-value below the 5% significance threshold.

Transformation of Radiation series

Applying first order differencing on Radiation series

```
radiationdiff = diff(radiation)
plot(radiationdiff,ylab='Radiation',xlab='Time period', col="darkmagenta", main = "Figure 34:
Time series plot of first differenced radiation series")
```

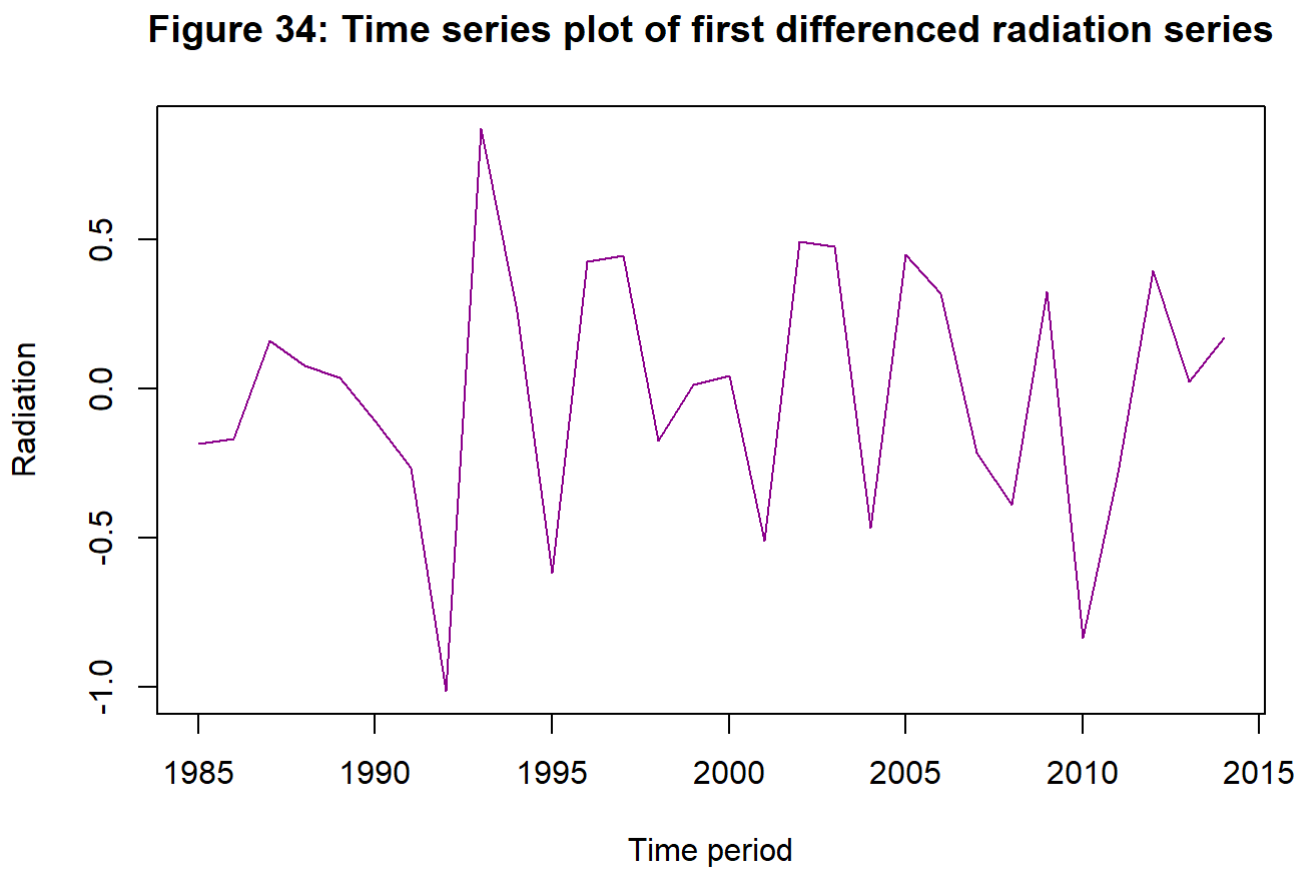


Figure 34

After applying the first difference to the Radiation series, it appears to be stationary. To validate this observation, we'll employ the ADF test on the series.

ADF Test on first order differenced Radiation

```
adf.test(radiationdiff)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  radiationdiff
## Dickey-Fuller = -2.6911, Lag order = 3, p-value = 0.3072
## alternative hypothesis: stationary
```

The above ADF test indicates that the series is still Non-stationary as the p-values is greater than 5% level of significance.Hence we will apply second differencing to the series.

Applying second order differencing on Radiation series

```
radiationdiff2 = diff(radiationdiff)
plot(radiationdiff2,ylab='Radiation',xlab='Time period', col="darkmagenta", main = "Figure 3
5: Time series plot of second differenced radiation series")
```

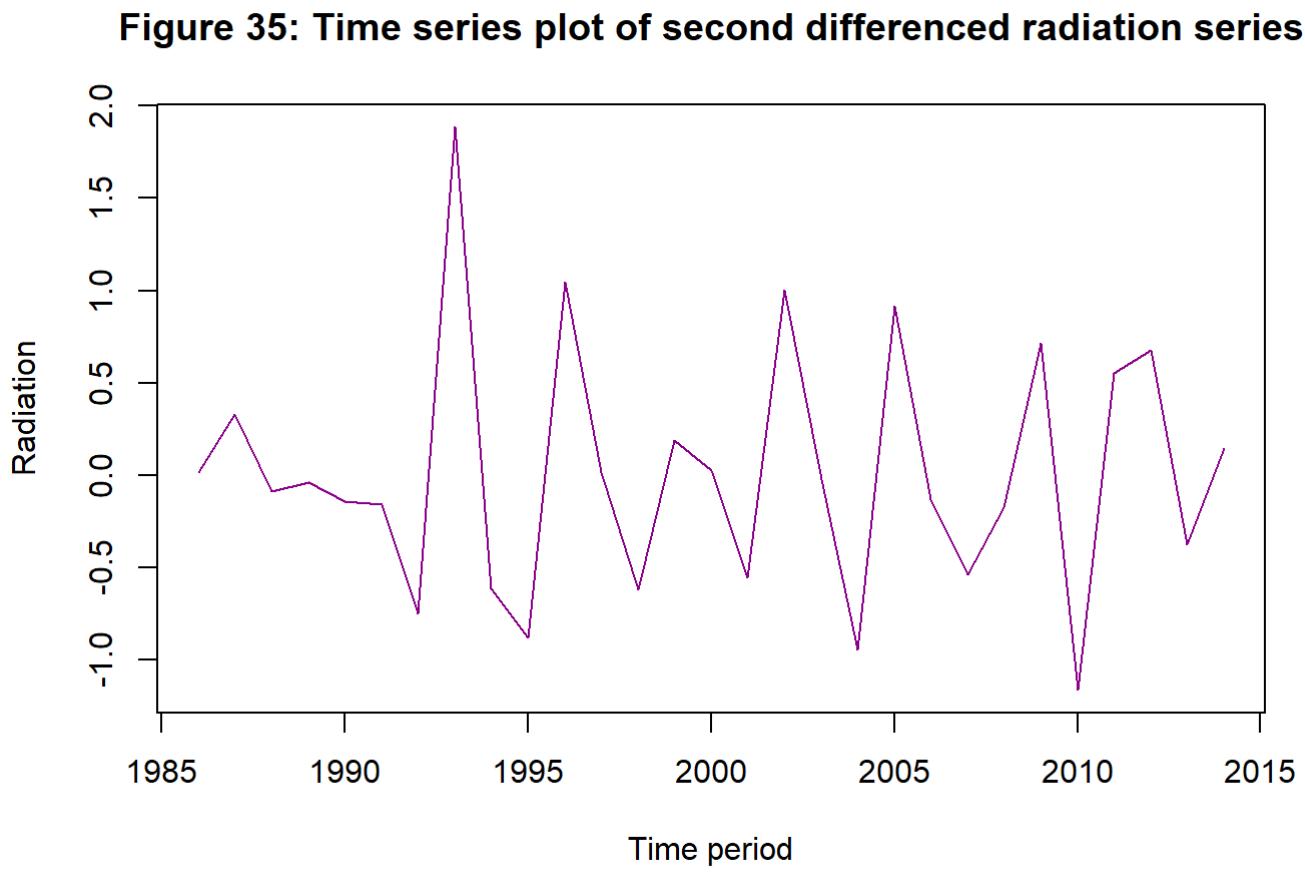


Figure 35

ADF Test on second order differenced Radiation

```
adf.test(radiationdiff2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  radiationdiff2
## Dickey-Fuller = -3.0559, Lag order = 3, p-value = 0.1678
## alternative hypothesis: stationary
```

The series is still non-stationary as the p-values is greater han 5% level. Let's apply third order differencing.

Applying third order differencing on Radiation series

```
radiationdiff3 = diff(radiationdiff2)
plot(radiationdiff3 ,ylab='Radiation',xlab='Time period', col="darkmagenta", main = "Figure 3
6: Time series plot of third differenced radiation series")
```

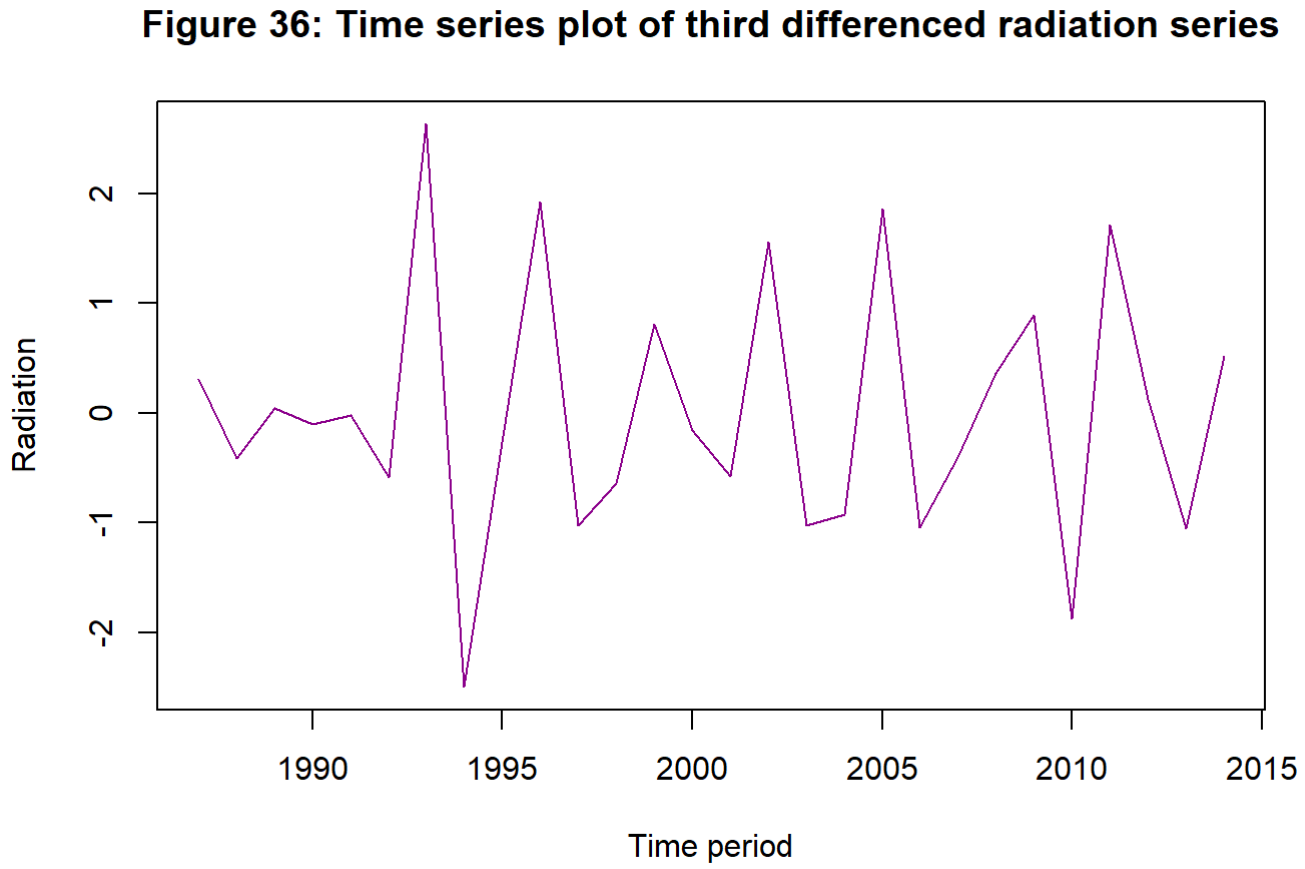


Figure 36

The series now more looks stationary, lets check the ADF test for check.

ADF Test on third order differenced Radiation

```
adf.test(radiationdiff3)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  radiationdiff3
## Dickey-Fuller = -4.1613, Lag order = 3, p-value = 0.01709
## alternative hypothesis: stationary
```

We can now finally observe that the series is now Stationary as the ADF test confirms the p-value lesser than 5% level of significance.

Hence, as a result, after implementing the second, third, and fourth order differencing, all three series (FFD, temperature, and radiation) have been effectively converted to a stationary state.Hence we can now proceed with modelling of the series.

Time Series Regression Models

We will try fitting distributed lag models, which incorporate an independent explanatory series and its lags to assist explain the general variance and correlation structure in our dependent series, in an effort to identify a good model for predicting Climate series.

To determine the model's finite lag length, we create a procedure that computes measurements of accuracy like AIC/BIC and MASE for models with various lag lengths, then select the model with the lowest values.

Distributed Lag Model

The Distributed Lag Model from the Regression models describes that the effect of an independent variable on the dependent variables occurs over the time. Therefore we require to build distributed lag models to reduce the multi-collinearity and dependency for each variable. Some of the most important used methods are:

- **Finite Distributed Lag Model**
- **Polynomial Distributed Lag Model**
- **Koyck Distributed Lag Model**
- **Autoregressive Distributed Lag Model**

Finite Distributed Lag Model

To identify the optimal model using finite DLM, we'll examine each of the four series (excluding FFD) individually as predictors and implement the finite DLM based on the suitable lag duration.

We will determine the appropriate lag length for the model, where temperature serves as the predictor and FFD as the outcome variable. This is done using the **finiteDLMauto** function, which compares based on AIC, BIC, and MASE scores for lags between 1 to 10. The model with the smallest AIC, BIC, and MASE values will be selected.

```
# Changing the column names of data-set

colnames(climate) <- c("temperature","rainfall","radiation","humidity","FFD")

# Performing finiteDLMauto to calculate best model based on AIC, BIC and MASE scores

finiteDLMauto( x=as.vector(temperature), y= as.vector(FFD), q.min = 1, q.max = 10, model.type
="dlm", error.type = "AIC",trace=TRUE)
```

```
##    q - k    MASE      AIC      BIC     GMRAE   MBRAE R.Adj.Sq Ljung-Box
## 10    10 0.54444 202.4974 216.0762  95.21457 0.75880  0.30833 0.8682028
## 9      9 0.69138 217.5864 230.6789  98.30030 0.78249  0.07685 0.1363229
## 8      8 0.70080 225.7275 238.2179  88.30203 1.10743  0.11344 0.4591795
## 7      7 0.69979 236.6118 248.3923  40.87507 0.55008  0.00966 0.8081728
## 6      6 0.75788 248.1290 259.0989 134.74503 0.47069 -0.13573 0.3385028
## 5      5 0.73794 255.3456 265.4104 102.43406 0.47497 -0.09659 0.2385812
## 4      4 0.84199 264.5275 273.5984 101.88888 0.38987 -0.15287 0.2012179
## 3      3 0.85614 270.8700 278.8632 204.15267 0.34556 -0.09628 0.2069108
## 2      2 0.84819 277.4305 284.2670 171.03798 0.47663 -0.04610 0.2278731
## 1      1 0.82301 284.3083 289.9131 144.59119 0.52710 -0.02241 0.1931572
```

Based on the results shown, the optimal lag-length is determined to be 10, as suggested by the **AIC, BIC, and MASE** metrics. The function **finiteDLMauto** was utilized to examine the lag length by substituting all other predictor variables. For every predictor variable in our dataset, the best lag length was consistently found to be **10**. Therefore, a lag length of 10 will be adopted for all the finite DLMs.

Modelling finite DLMs with several possible combinations

```
# Fitting a finite DLM model using temperature as the predictor variable
model.temperature = dlm(x=as.vector(temperature), y=as.vector(FFD), q=10)

# Fitting a finite DLM model using rainfall as the predictor variable
model.rainfall = dlm(x=as.vector(rainfall), y=as.vector(FFD), q=10)

# Fitting a finite DLM model using humidity as the predictor variable
model.humidity = dlm(x=as.vector(humidity), y=as.vector(FFD), q=10)

# Fitting a finite DLM model using radiation as the predictor variable
model.radiation = dlm(x=as.vector(radiation), y=as.vector(FFD), q=10)

# Fitting a finite DLM model using FFD and temperature with no intercept
model.nointercept = dlm(formula = FFD ~ temperature - 1, data=data.frame(climate), q=10)

# Fitting a finite DLM model using FFD and rainfall with no intercept
model.nointercept2 = dlm(formula = FFD ~ rainfall - 1, data=data.frame(climate), q=10)

# Fitting a finite DLM model using FFD and humidity with no intercept
model.nointercept3 = dlm(formula = FFD ~ humidity - 1, data=data.frame(climate), q=10)

# Fitting a finite DLM model using FFD and radiation with no intercept
model.nointercept4 = dlm(formula = FFD ~ radiation - 1, data=data.frame(climate), q=10)
```

We have fitted the possible finite dlm models above. Lets evaluate them based on AIC, BIC and MASE scores.

Evaluating Model based on AIC scores

```
sort.score(AIC(model.temperature$model,model.rainfall$model,model.humidity$model,model.radiat
ion$model,model.nointercept$model,model.nointercept2$model,model.nointercept3$model,model.noi
ntercept4$model), score = "aic")
```

```
##                           df      AIC
## model.humidity$model      13 197.6230
## model.rainfall$model      13 200.1260
## model.radiation$model     13 200.6556
## model.temperature$model   13 202.4974
## model.nointercept3$model  12 204.2049
## model.nointercept2$model  12 206.9233
## model.nointercept$model   12 216.5454
## model.nointercept4$model  12 217.1227
```

Evaluating Model based on BIC scores

```
sort.score(BIC(model.temperature$model,model.rainfall$model,model.humidity$model,model.radiat
ion$model,model.nointercept$model,model.nointercept2$model,model.nointercept3$model,model.noi
ntercept4$model), score = "bic")
```

```
##                           df      BIC
## model.humidity$model      13 211.2018
## model.rainfall$model      13 213.7048
## model.radiation$model     13 214.2344
## model.temperature$model   13 216.0762
## model.nointercept3$model  12 216.7392
## model.nointercept2$model  12 219.4576
## model.nointercept$model   12 229.0797
## model.nointercept4$model  12 229.6569
```

Evaluating Model based on MASE scores

```
sort.score(MASE(model.temperature$model,model.rainfall$model,model.humidity$model,model.radia
tion$model,model.nointercept$model,model.nointercept2$model,model.nointercept3$model,model.no
intercept4$model), score = "mase")
```

```
##                            n      MASE
## model.radiation$model     21 0.5096810
## model.humidity$model      21 0.5150247
## model.temperature$model   21 0.5444386
## model.rainfall$model      21 0.5575248
## model.nointercept3$model  21 0.5640817
## model.nointercept2$model  21 0.6250600
## model.nointercept$model   21 0.7811826
## model.nointercept4$model  21 0.8342048
```

From the outputs of above result, the best model based on AIC and BIC score is “**model.humidity**” whereas the best model based on MASE scores is “**model.radiation**” at 0.5096. Hence we will further analyse the model with least MASE scores.

Analysing Best Finite DLM model

```
model.finite2 <- dlm(x=as.vector(radiation), y=as.vector(FFD), q=10)
summary(model.finite2)
```

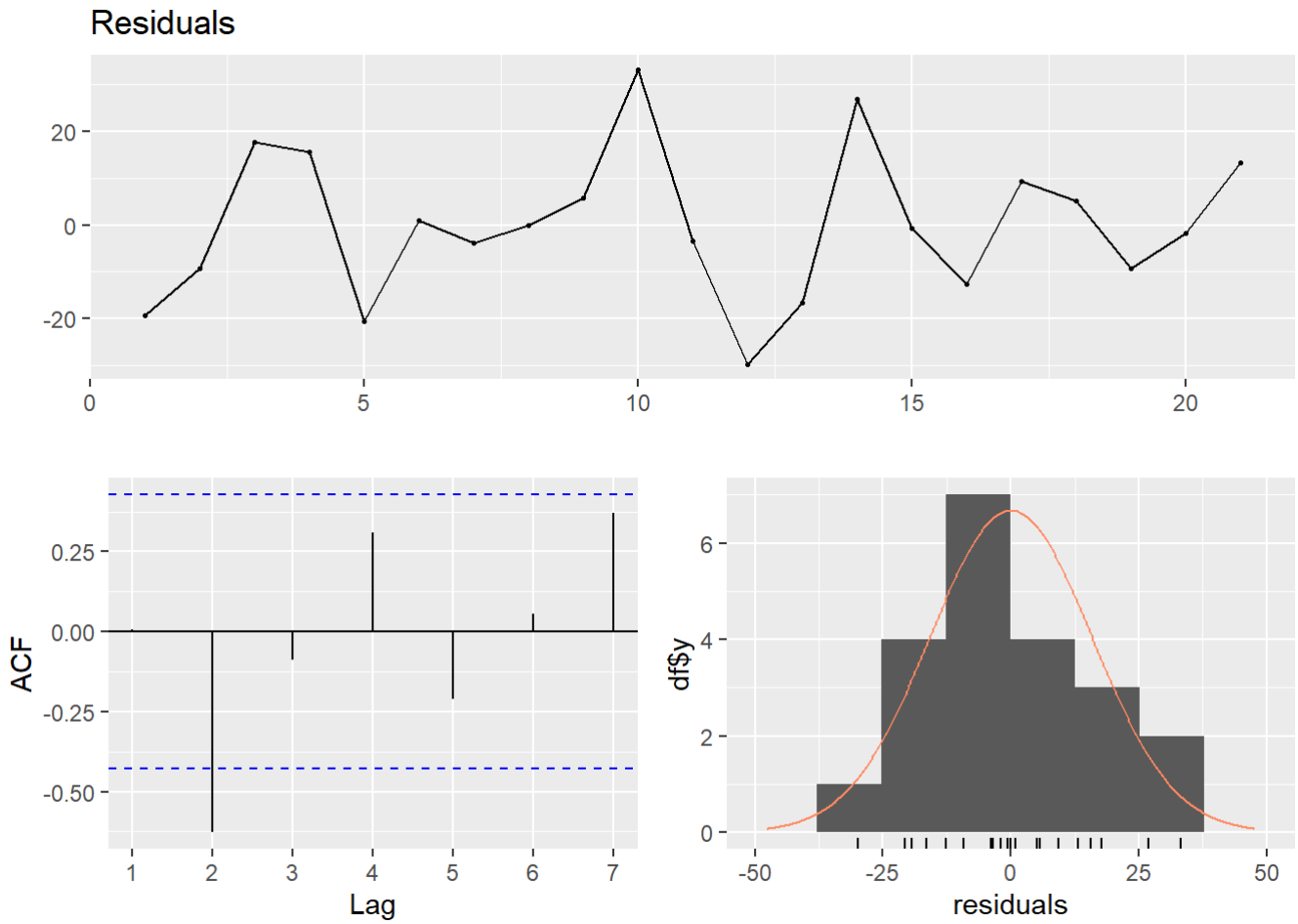
```
##
## Call:
## lm(formula = model.formula, data = design)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -29.915  -9.263  -0.685   9.272  33.240
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1692.078    475.093   3.562   0.0061 **
## x.t            6.888     28.587   0.241   0.8150
## x.1           -4.317     34.127  -0.126   0.9021
## x.2          -31.223     20.493  -1.524   0.1619
## x.3          -10.062     19.570  -0.514   0.6195
## x.4           21.511     20.497   1.049   0.3213
## x.5          -11.387     19.484  -0.584   0.5733
## x.6           26.797     20.209   1.326   0.2175
## x.7          -46.991     19.816  -2.371   0.0418 *
## x.8            2.182     20.139   0.108   0.9161
## x.9           -8.869     24.882  -0.356   0.7297
## x.10         -39.904     25.003  -1.596   0.1450
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23.65 on 9 degrees of freedom
## Multiple R-squared:  0.7149, Adjusted R-squared:  0.3664
## F-statistic: 2.051 on 11 and 9 DF,  p-value: 0.1451
##
## AIC and BIC values for the model:
##        AIC      BIC
## 1 200.6556 214.2344
```

From the outputs of **finite_best2** model and its summary :

- The **AIC** and **BIC** scores of the model have been reported 200.65 and 214.23.
- The adjusted R-squared value reported was 0.36 which is very poor.
- Very few of the coefficients are significant.
- The F-statistic is reported at 2.051 on 11 and 9 Degrees of Freedom.

Residual Analysis of the model “finite_best2”

```
checkresiduals(model.finite2$model,test=FALSE)
```



From the outputs of Model “finite_best2” residuals :

- The residuals appear to be distributed randomly without exhibiting a specific trend.
- The ACF plot reveals that there is seasonality and correlation exist in residual due to significant lags present in the ACF.
- The time-series plot of residual does not seems to follow any trend overall.

So, we can conclude that the model “finite_best2” is a decent model with a appropriate composition of residuals and a MASE score of 0.5096

Polynomial Distributed Lag Model

Having observed that the finite DLM model provided a satisfactory fit, our next step is to explore the potential of a polynomial model. However, prior to fitting the polynomial model, we’ll utilize the `finiteDLMauto` function to determine the optimal lag length, basing our decision on the AIC, BIC, and MASE values of the fitted models.

```
# Computing lag length based on AIC, BIC & MASE

finiteDLMauto(x = as.vector(temperature), y = as.vector(FFD), q.min = 1, q.max = 10, k.order
= 2,
              model.type = "poly", error.type ="AIC", trace = TRUE)
```

```
##     q - k    MASE      AIC      BIC     GMRAE   MBRAE R.Adj.Sq Ljung-Box
## 10 10 - 2 0.72684 202.7983 208.0209 103.40160 0.48786  0.20420 0.9729531
## 9   9 - 2 0.79308 214.5153 219.9706  97.02563 0.64438  0.07287 0.9318007
## 8   8 - 2 0.76111 217.1387 222.8161  92.37623 1.18951  0.29643 0.9162658
## 7   7 - 2 0.74470 228.9782 234.8685  49.02174 0.24043  0.18027 0.9598896
## 6   6 - 2 0.80934 242.7381 248.8325 145.27851 0.53302 -0.02054 0.7399512
## 5   5 - 2 0.78912 250.9262 257.2167  93.93372 0.50845 -0.00641 0.3962142
## 4   4 - 2 0.84852 261.1783 267.6575 100.01464 0.45172 -0.07830 0.2710057
## 3   3 - 2 0.86130 269.4499 276.1109 232.57145 0.38461 -0.07259 0.2822331
## 2   2 - 2 0.84819 277.4305 284.2670 171.03798 0.47663 -0.04610 0.2278731
## 1   1 - 2 0.82301 284.3083 289.9131 144.59119 0.52710 -0.02241 0.1931572
```

Based on the presented results, the optimal lag-length is determined to be 10, as evidenced by the AIC, BIC, and MASE values. By employing the finiteDLMauto function, we assessed different predictors to ascertain the appropriate lag length. It was observed that the best lag length for all predictor variables in the dataset is 10. Consequently, for every polynomial DLM, we will use a lag length of 10 and set the order to 2.

Fitting all combination of polynomial models

```
# Fitting a polynomial DLM model using temperature as the predictor variable
model.temperaturepoly = polyDlm(x = as.vector(temperature), y = as.vector(FFD), q = 10, k =
2)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value P(>|t|)
## beta.0     5.680       5.91   0.960  0.3580
## beta.1     3.650       3.94   0.925  0.3750
## beta.2     1.630       2.82   0.577  0.5750
## beta.3    -0.387       2.60  -0.149  0.8850
## beta.4    -2.390       2.85  -0.839  0.4190
## beta.5    -4.390       3.10  -1.420  0.1850
## beta.6    -6.380       3.22  -1.980  0.0731
## beta.7    -8.360       3.32  -2.520  0.0286
## beta.8   -10.300       3.71  -2.780  0.0178
## beta.9   -12.300       4.72  -2.600  0.0245
## beta.10  -14.300       6.47  -2.210  0.0496
```

```
# Fitting a polynomial DLM model using rainfall as the predictor variable
model.rainfallpoly = polyDlm(x = as.vector(rainfall), y = as.vector(FFD), q = 10, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value P(>|t|)
## beta.0     -7.88      12.20  -0.643  0.5330
## beta.1     -8.21       8.14  -1.010  0.3350
## beta.2     -7.78       6.36  -1.220  0.2470
## beta.3     -6.58       6.41  -1.030  0.3260
## beta.4     -4.62       6.83  -0.676  0.5130
## beta.5     -1.89       6.75  -0.279  0.7850
## beta.6      1.61       6.03   0.267  0.7950
## beta.7      5.87       5.23   1.120  0.2860
## beta.8     10.90       6.06   1.800  0.0998
## beta.9     16.70       9.63   1.730  0.1110
## beta.10    23.30      15.30   1.520  0.1560
```

```
# Fitting a polynomial DLM model using humidity as the predictor variable
model.humiditypoly = polyDlm(x = as.vector(humidity), y = as.vector(FFD), q = 10, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error  t value P(>|t|)
## beta.0   0.13000       2.67  0.04870  0.9620
## beta.1  -0.81100       2.03 -0.40000  0.6970
## beta.2  -1.29000       1.76 -0.73700  0.4770
## beta.3  -1.32000       1.70 -0.77500  0.4550
## beta.4  -0.88800       1.69 -0.52600  0.6090
## beta.5   0.00165       1.61  0.00103  0.9990
## beta.6   1.35000       1.50  0.89800  0.3880
## beta.7   3.15000       1.53  2.07000  0.0632
## beta.8   5.41000       1.93  2.80000  0.0173
## beta.9   8.13000       2.80  2.91000  0.0142
## beta.10 11.30000       4.04  2.80000  0.0173
```

```
# Fitting a polynomial DLM model using radiation as the predictor variable
model.radiationpoly = polyDlm(x = as.vector(radiation), y = as.vector(FFD), q = 10, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value P(>|t|)
## beta.0    -10.60       8.53  -1.240  0.2420
## beta.1     -6.63       5.46  -1.210  0.2500
## beta.2     -3.84       3.79  -1.010  0.3330
## beta.3     -2.20       3.55  -0.618  0.5490
## beta.4     -1.70       3.87  -0.440  0.6690
## beta.5     -2.35       3.99  -0.589  0.5680
## beta.6     -4.15       3.77  -1.100  0.2950
## beta.7     -7.09       3.54  -2.000  0.0707
## beta.8    -11.20       4.24  -2.640  0.0232
## beta.9    -16.40       6.45  -2.540  0.0273
## beta.10   -22.80       9.94  -2.290  0.0425
```

We have successfully fitted all the possible models uning polynomial method. We will now check the Accuracy scores of each model and sort them out.

Evaluating based on AIC, BIC and MASE scores

```
# AIC scores
sort.score(AIC(model.temperaturepoly$model,model.rainfallpoly$model,model.humiditypoly$model,
model.radiationpoly$model), score = "aic")
```

```
##                             df      AIC
## model.humiditypoly$model     5 202.2992
## model.temperaturepoly$model  5 202.7983
## model.radiationpoly$model    5 203.0988
## model.rainfallpoly$model     5 206.0399
```

```
# BIC scores
sort.score(BIC(model.temperaturepoly$model,model.rainfallpoly$model,model.humiditypoly$model,
model.radiationpoly$model), score = "bic")
```

```
##                             df      BIC
## model.humiditypoly$model     5 207.5218
## model.temperaturepoly$model  5 208.0209
## model.radiationpoly$model    5 208.3214
## model.rainfallpoly$model     5 211.2625
```

```
# MASE scores
sort.score(MASE(model.temperaturepoly$model,model.rainfallpoly$model,model.humiditypoly$mode
l,model.radiationpoly$model), score = "mase")
```

```
##                              n      MASE
## model.temperaturepoly$model 21 0.7268353
## model.humiditypoly$model    21 0.7461637
## model.radiationpoly$model   21 0.7781601
## model.rainfallpoly$model    21 0.8045022
```

From the results of above scores, the best model turns out is “**model.temperaturepoly**”. Hence we will further analyze and summarise the model.

Analyzing and summarising model

```
summary(model.temperaturepoly$model)
```

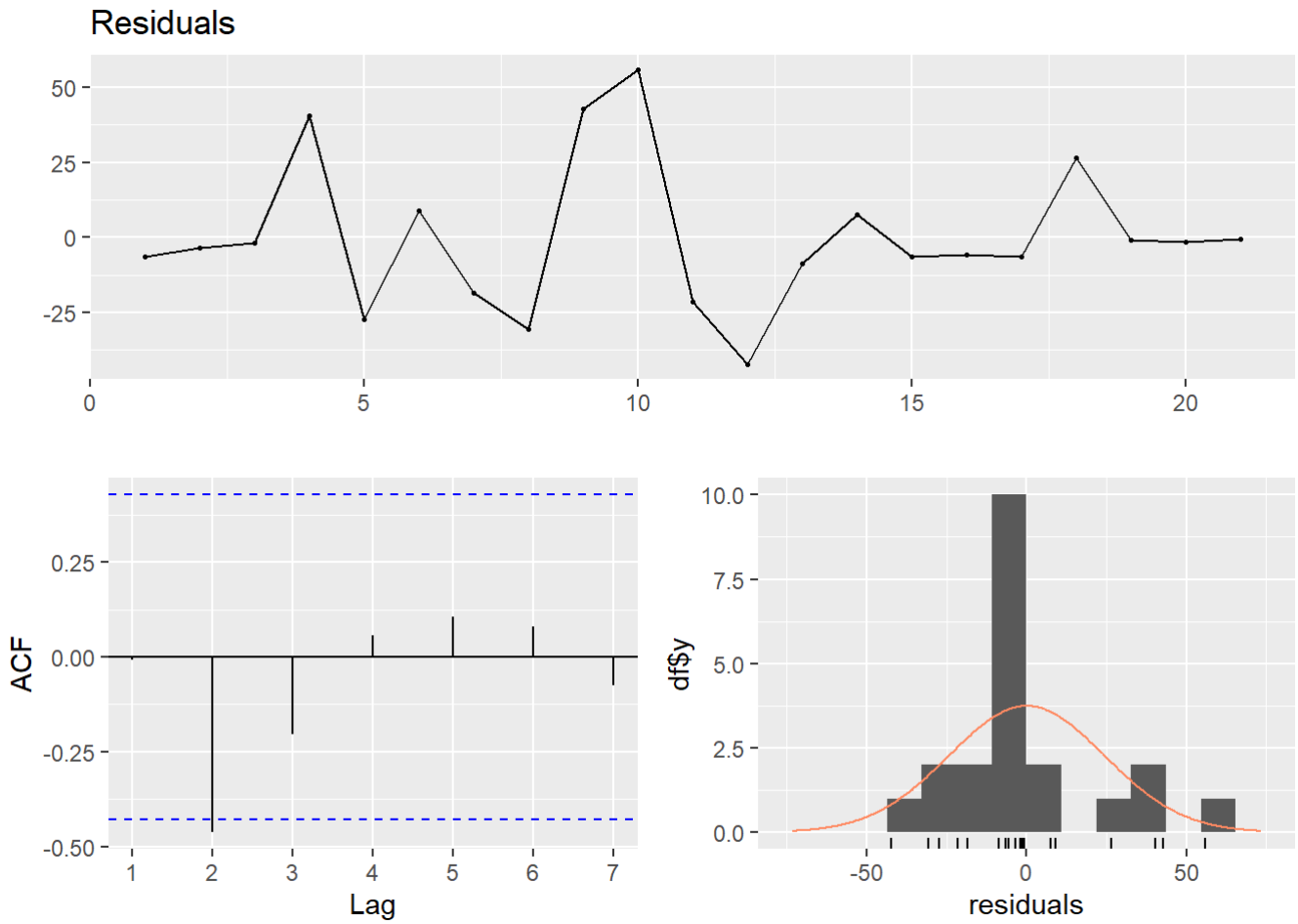
```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -42.389  -8.563  -3.514   7.604  55.874
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.233e+03  3.929e+02   3.138    0.006 **
## z.t0         5.676e+00  5.915e+00   0.960    0.351
## z.t1        -2.033e+00  2.709e+00  -0.750    0.463
## z.t2         3.899e-03  2.516e-01   0.016    0.988
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.5 on 17 degrees of freedom
## Multiple R-squared:  0.3236, Adjusted R-squared:  0.2042
## F-statistic: 2.711 on 3 and 17 DF,  p-value: 0.07751
```

From the outputs of **model.temperaturepoly** model and its summary :

- The p-values is reported at 0.077 which is greater than 5% level of significance.
- The adjusted R-squared value reported was 0.20 which is very poor.
- Very few of the coefficients are significant.
- The F-statistic is reported at 2.71 on 3 and 17 Degrees of Freedom.

Residual Analyisi

```
checkresiduals(model.temperaturepoly$model,test=FALSE)
```



From the outputs of Model "**model.temperaturepoly**" residuals :

- The residuals appear to be distributed randomly without exhibiting a specific trend.
- The ACF plot reveals that there is seasonality and correlation exist in residual due to significant lag present in the ACF.
- The time-series plot of residual does not seems to follow any trend overall.

So, we can conclude that the model "**model.temperaturepoly**" is a decent model with a appropriate composition of residuals and a MASE score of 0.726.

Forecasting the **model.temperaturepoly**

```
# Creating point predictions utilizing the task2_covariate dataset for radiation.

forecastFFDpoint <- dLagM::forecast(model.temperaturepoly,x = c(14.60,14.56,14.79,14.79) ,h =
4)

forecastFFDpoint <- round(forecastFFDpoint$forecasts,2)

# Generating prediction intervals

forecastFFDinterval <- forecast(model = model.temperaturepoly, x = c(14.60,14.56,14.79,14.79)
,
                h = 4 , interval = TRUE)

round(forecastFFDinterval$forecasts,2)
```

```
##    Lower Estimate  Upper
## 1 190.11   240.49 295.65
## 2 168.36   219.83 272.73
## 3 156.68   207.12 257.23
## 4 169.69   220.57 271.32
```

Plotting the Forecast

```
plot(ts(c(as.vector(FFD),forecastFFDpoint),start=1984),type="o",col="black", ylab="FFD_foreca
sted",
main="Figure 37: FFD four year ahead predicted values from 2015-2018")
lines(ts(c(as.vector(FFD),forecastFFDpoint),start=1984),type="o",col="darkmagenta")
legend("topleft", lty = 1,pch=1,text.width=11, col = c("black","darkmagenta"), c("Data","mode
l.radiationpoly"))
```

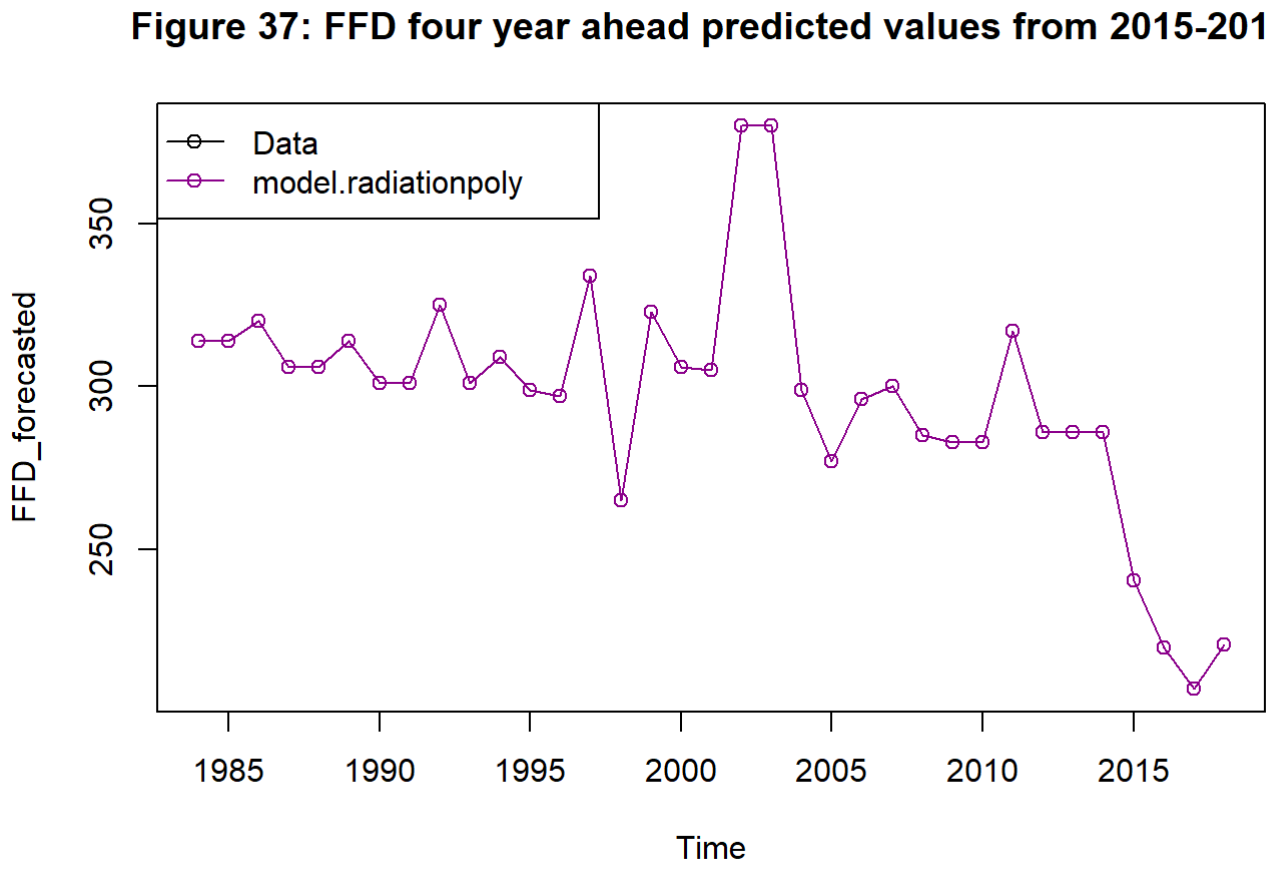


Figure 37

The displayed graph presents a four-year forecast for FFD based on the model.radiationpoly. The predictions appear trustworthy since the model aligns seamlessly with the FFD dataset, and the forecasted intervals are suitably narrow, indicating a solid model performance.

Koyck Distributed Lag Model

Fitting Koyck models with all possible combinations

```
# Fitting a Koyck DLM model using temperature as the predictor variable
model.temperaturekoyck = koyckDlm(x = as.vector(temperature), y = as.vector(FFD))

# Fitting a Koyck DLM model using rainfall as the predictor variable
model.rainfallkoyck = koyckDlm(x = as.vector(rainfall), y = as.vector(FFD))

# Fitting a Koyck DLM model using humidity as the predictor variable
model.humiditykoyck = koyckDlm(x = as.vector(humidity), y = as.vector(FFD))

# Fitting a Koyck DLM model using radiation as the predictor variable
model.radiationkoyck = koyckDlm(x = as.vector(radiation), y = as.vector(FFD))
```

Above we have fitted several models based on koyck method.

Evaluating base on MASE scores

```
# Sorting based on  MASE scores
MASE_koyck <- MASE(model.temperaturekoyck,model.rainfallkoyck,model.humiditykoyck,model.radia
tionkoyck)


# Arranging in ascending order

arrange(MASE_koyck,MASE)
```

```
##                          n      MASE
## model.temperaturekoyck 30 0.8454533
## model.humiditykoyck    30 0.8921521
## model.radiationkoyck   30 0.9751565
## model.rainfallkoyck    30 1.0169315
```

Analyzing and summarising the best model **model.temperaturekoyck**

```
summary(model.temperaturekoyck,diagnostics = TRUE)
```

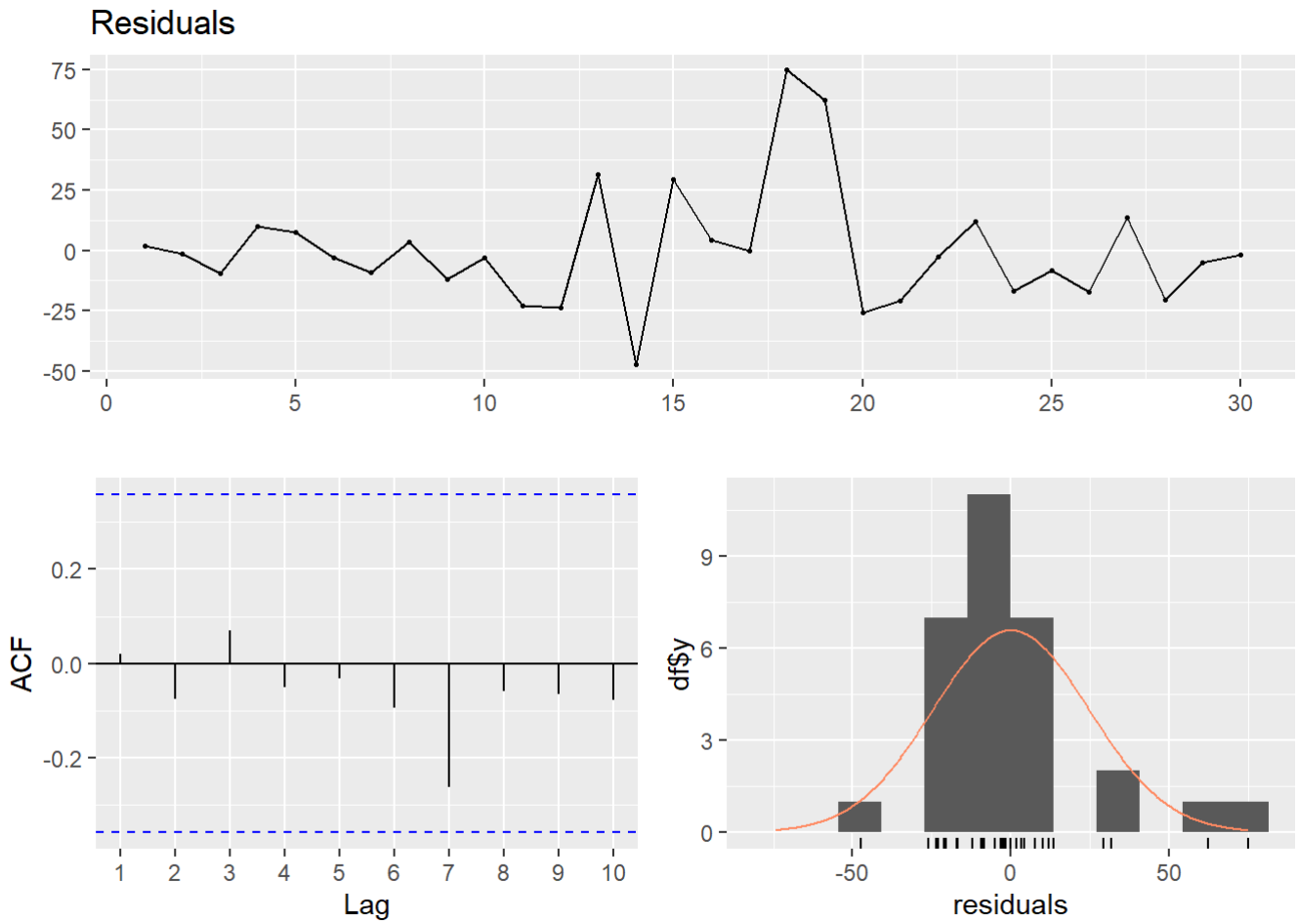
```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -47.451 -15.487  -2.648   6.757  75.055
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 526.5488   401.9192   1.310    0.201
## Y.1           0.1585     0.2372   0.668    0.510
## X.t         -13.7092    18.0543  -0.759    0.454
##
## Residual standard error: 25.66 on 27 degrees of freedom
## Multiple R-Squared: 0.03631, Adjusted R-squared: -0.03508
## Wald test: 1.255 on 2 and 27 DF,  p-value: 0.3011
##
## Diagnostic tests:
##                  df1 df2 statistic    p-value
## Weak instruments   1  27 6.2474539 0.01881608
## Wu-Hausman         1  26 0.3063724 0.58464375
##
##                             alpha      beta       phi
## Geometric coefficients:  625.7113 -13.70923 0.1584797
```

From the outputs of **model.temperaturekoyck** model and its summary :

- The p-values is reported at 0.301 which is much greater than 5% level of significance.
- The adjusted R-squared value reported was 0.03508 which is very poor.
- None of the coefficients are significant.
- The Wu-Hausman test is reported at 0.30 on 1 and 26 Degrees of Freedom.

Residual analysis of model **model.temperaturekoyck**

```
checkresiduals(model.temperaturekoyck$model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 0.86475, df = 6, p-value = 0.9902
##
## Model df: 0.   Total lags used: 6
```

From the outputs of Model "**model.temperaturekoyck**" residuals :

- The p-value from the Ljung-Box test was reported at 0.99 which is much greater than 5% level of significance.
- The residuals appear to be distributed randomly without exhibiting a specific trend.
- The ACF plot reveals that there is seasonality and correlation exist in residual due to significant lag present in the ACF.
- The Histogram is unevenly distributed overall.
- The time-series plot of residual does not seems to follow any trend overall.

So, we can conclude that the model "**model.temperaturekoyck**" is a decent model with a appropriate composition of residuals and a MASE score of 0.845.

Forecasting **model.temperaturekoyck**

Let's produce the forecast intervals and visualize the predictions for the upcoming four years.

```
# Producing point predictions utilizing the radiation values from task2_covariate
forecastFFDpoint2 <- dLagM::forecast(model.temperaturekoyck,x = c(14.60,14.56,14.79,14.79) ,h
= 4)

forecastFFDpoint2 <- round(forecastFFDpoint2$forecasts,2)

# Generating prediction intervals of forecast

forecastFFDinterval2 <- forecast(model = model.temperaturekoyck, x = c(14.60,14.56,14.79,14.7
9) , h = 4 ,interval = TRUE)

round(forecastFFDinterval2$forecasts,2)
```

```
##    Lower Estimate  Upper
## 1 321.39   371.72 420.69
## 2 332.74   385.85 435.91
## 3 330.17   384.94 441.15
## 4 328.78   384.79 438.53
```

The predictions derived from the Koyck model aren't as satisfactory as those from the poly model. According to the Koyck model, the projected FFD values for the upcoming four years will hover around 371-385, showing a bit fluctuation. Moreover, the prediction intervals for the Koyck model are broader compared to the Poly model as it is capturing the **intervention peak from the previous data**.

Plotting the forecast

```
plot(ts(c(as.vector(FFD),forecastFFDpoint2),start=1984),type="o",col="black", ylab="FFD_forec
asted",
main="Figure 38: FFD four year ahead predicted values from 2015-2018")
lines(ts(c(as.vector(FFD),forecastFFDpoint2),start=1984),type="o",col="navy")
legend("topright", lty = 1,pch=1,text.width=11, col = c("black","navy"), c("Data","model.temp
eraturekoyck"))
```

Figure 38: FFD four year ahead predicted values from 2015-2018

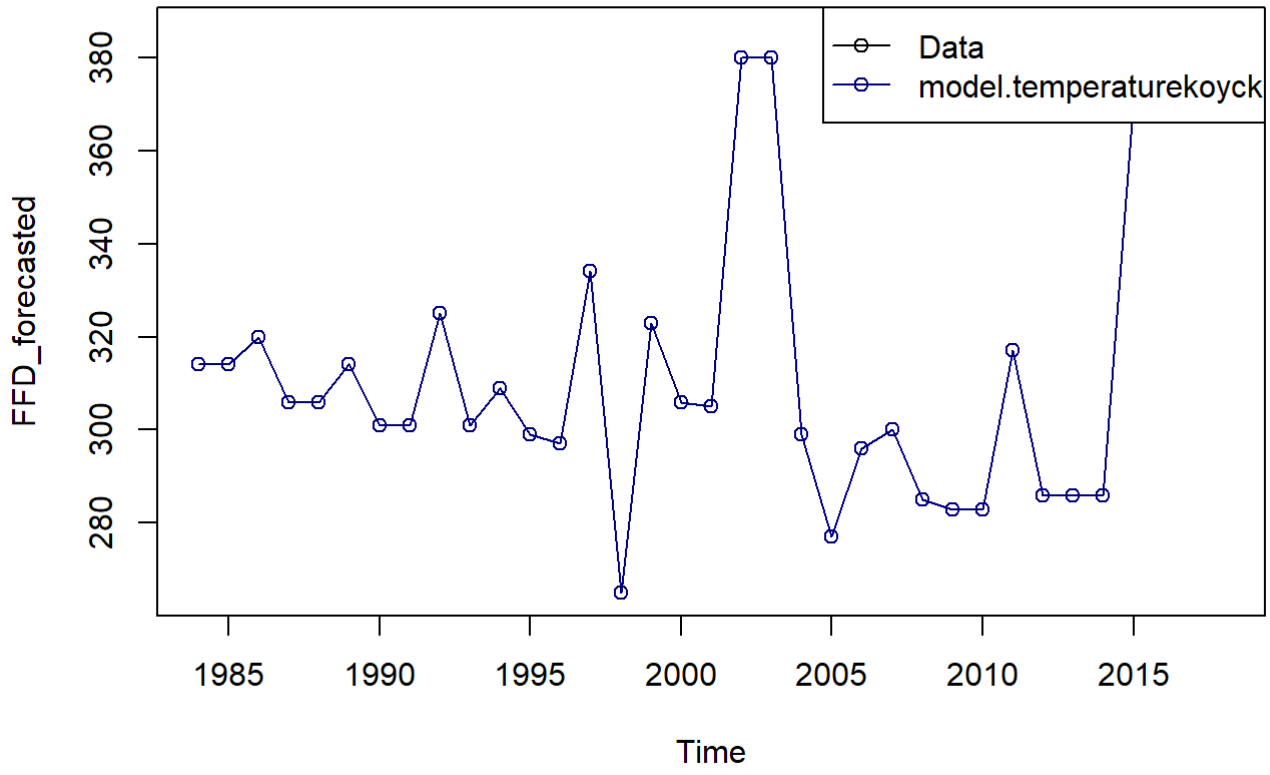


Figure 38

The Koyck model aligns seamlessly with the historical FFD data. According to the model's projections, over the upcoming four years, FFD values will remain relatively stable, hovering around 371-385. Given the intervention point peak in prediction intervals, the forecasted values appear to be trustworthy.

Autoregressive Distributed Lag Model

Calculating Autoregressive DLMs over various lag duration and AR process levels through a loop, then selecting the model that has the minimal information criterion for fitting.

```
for (i in 1:5){
  for(j in 1:5){
    model.autoreg = ardlDlm(x= as.vector(temperature),y=as.vector(FFD), p = i , q = j )
    cat("p =", i, "q =", j, "AIC =", AIC(model.autoreg$model), "BIC =", BIC(model.autoreg$mod
el), "MASE =", MASE(model.autoreg)$MASE, "\n")
  }
}
```

```
## p = 1 q = 1 AIC = 284.8533 BIC = 291.8593 MASE = 0.8228112
## p = 1 q = 2 AIC = 278.0806 BIC = 286.2844 MASE = 0.8374506
## p = 1 q = 3 AIC = 271.662 BIC = 280.9874 MASE = 0.8526614
## p = 1 q = 4 AIC = 264.9657 BIC = 275.3324 MASE = 0.8562152
## p = 1 q = 5 AIC = 258.6201 BIC = 269.943 MASE = 0.8645458
## p = 2 q = 1 AIC = 277.8754 BIC = 286.0792 MASE = 0.873467
## p = 2 q = 2 AIC = 279.2593 BIC = 288.8304 MASE = 0.8792838
## p = 2 q = 3 AIC = 272.4696 BIC = 283.1273 MASE = 0.8861359
## p = 2 q = 4 AIC = 265.8572 BIC = 277.5197 MASE = 0.8794839
## p = 2 q = 5 AIC = 259.3834 BIC = 271.9644 MASE = 0.879796
## p = 3 q = 1 AIC = 271.3617 BIC = 280.6871 MASE = 0.8776525
## p = 3 q = 2 AIC = 272.7623 BIC = 283.4199 MASE = 0.8843112
## p = 3 q = 3 AIC = 274.4652 BIC = 286.455 MASE = 0.8875795
## p = 3 q = 4 AIC = 267.8348 BIC = 280.7932 MASE = 0.878722
## p = 3 q = 5 AIC = 261.3622 BIC = 275.2013 MASE = 0.8763482
## p = 4 q = 1 AIC = 265.0053 BIC = 275.3719 MASE = 0.8620171
## p = 4 q = 2 AIC = 266.3633 BIC = 278.0259 MASE = 0.8601795
## p = 4 q = 3 AIC = 268.0405 BIC = 280.9989 MASE = 0.8626041
## p = 4 q = 4 AIC = 269.8042 BIC = 284.0584 MASE = 0.8729167
## p = 4 q = 5 AIC = 263.233 BIC = 278.3302 MASE = 0.8596
## p = 5 q = 1 AIC = 255.5008 BIC = 266.8237 MASE = 0.7769052
## p = 5 q = 2 AIC = 256.8777 BIC = 269.4586 MASE = 0.7623793
## p = 5 q = 3 AIC = 258.0904 BIC = 271.9295 MASE = 0.7356013
## p = 5 q = 4 AIC = 259.5186 BIC = 274.6158 MASE = 0.732152
## p = 5 q = 5 AIC = 261.187 BIC = 277.5423 MASE = 0.7198996
```

Based on the provided results, the ARDL(5,5) model stands out as the optimal choice, as evidenced by its AIC, BIC, and MASE metrics. The finiteDLMauto function was employed to evaluate other predictors and determine the best p & q values. The analysis revealed that the ARDL(5,5) configuration consistently delivered the best performance for all predictor variables. Hence, for all autoregressive DLMs, we will use the (5,5) values for p & q.

Fitting Autoregressive model with serveral possible combinations

```
model.temperatureardl = ardlDlm(x=as.vector(temperature), y=as.vector(FFD), p = 5, q = 5)
model.rainfallardl = ardlDlm(x=as.vector(rainfall), y=as.vector(FFD), p = 5, q = 5)
model.humidityardl = ardlDlm(x=as.vector(humidity), y=as.vector(FFD), p = 5, q = 5)
model.radiationardl = ardlDlm(x=as.vector(radiation), y=as.vector(FFD), p = 5, q = 5)
```

Evaluating Model based on AIC, BIC and MASE scores

```
# AIC scores
sort.score(AIC(model.temperatureardl$model,model.rainfallardl$model,model.humidityardl$model,
model.radiationardl$model), score = "aic")
```

```
##                              df      AIC
## model.rainfallardl$model    13 253.8860
## model.temperatureardl$model 13 261.1870
## model.radiationardl$model   13 265.1234
## model.humidityardl$model    13 266.0988
```

```
# BIC scores
sort.score(BIC(model.temperatureardl$model,model.rainfallardl$model,model.humidityardl$model,
model.radiationardl$model), score = "bic")
```

```
##                              df      BIC
## model.rainfallardl$model    13 270.2413
## model.temperatureardl$model 13 277.5423
## model.radiationardl$model   13 281.4786
## model.humidityardl$model    13 282.4541
```

```
# MASE scores
Maseardl <- MASE(model.temperatureardl,model.rainfallardl,model.humidityardl,model.radiationa
rdl)

arrange(Maseardl,MASE)
```

```
##                         n      MASE
## model.temperatureardl 26 0.7198996
## model.rainfallardl    26 0.7385166
## model.radiationardl   26 0.8283986
## model.humidityardl    26 0.9144586
```

Based on the AIC and BIC metrics, the model using only radiation as a predictor emerges as the top choice. However, when considering the MASE metric, the model with rainfall as the predictor takes precedence. We'll delve deeper into the latter model since it's grounded on the MASE evaluation.

Analyzing and summarising **model.rainfallardl**

```
summary(model.rainfallardl)
```

```
##
## Time series regression with "ts" data:
## Start = 6, End = 31
##
## Call:
## dynlm(formula = as.formula(model.text), data = data, start = 1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -42.268 -13.132  -4.447  12.986  40.148
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 410.082613 160.315445   2.558   0.0228 *
## X.t         -33.930608  15.558394  -2.181   0.0467 *
## X.1          15.501254  17.353599   0.893   0.3868
## X.2          -9.690104  18.755022  -0.517   0.6135
## X.3          -0.697562  19.149439  -0.036   0.9715
## X.4          16.623087  20.331258   0.818   0.4273
## X.5         -34.282583  16.469981  -2.082   0.0562 .
## Y.1           0.527250   0.255924   2.060   0.0585 .
## Y.2          -0.240139   0.293847  -0.817   0.4275
## Y.3           0.004557   0.321936   0.014   0.9889
## Y.4           0.103765   0.312075   0.332   0.7444
## Y.5          -0.378869   0.299419  -1.265   0.2264
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.39 on 14 degrees of freedom
## Multiple R-squared:  0.4637, Adjusted R-squared:  0.04225
## F-statistic:   1.1 on 11 and 14 DF,  p-value: 0.4256
```

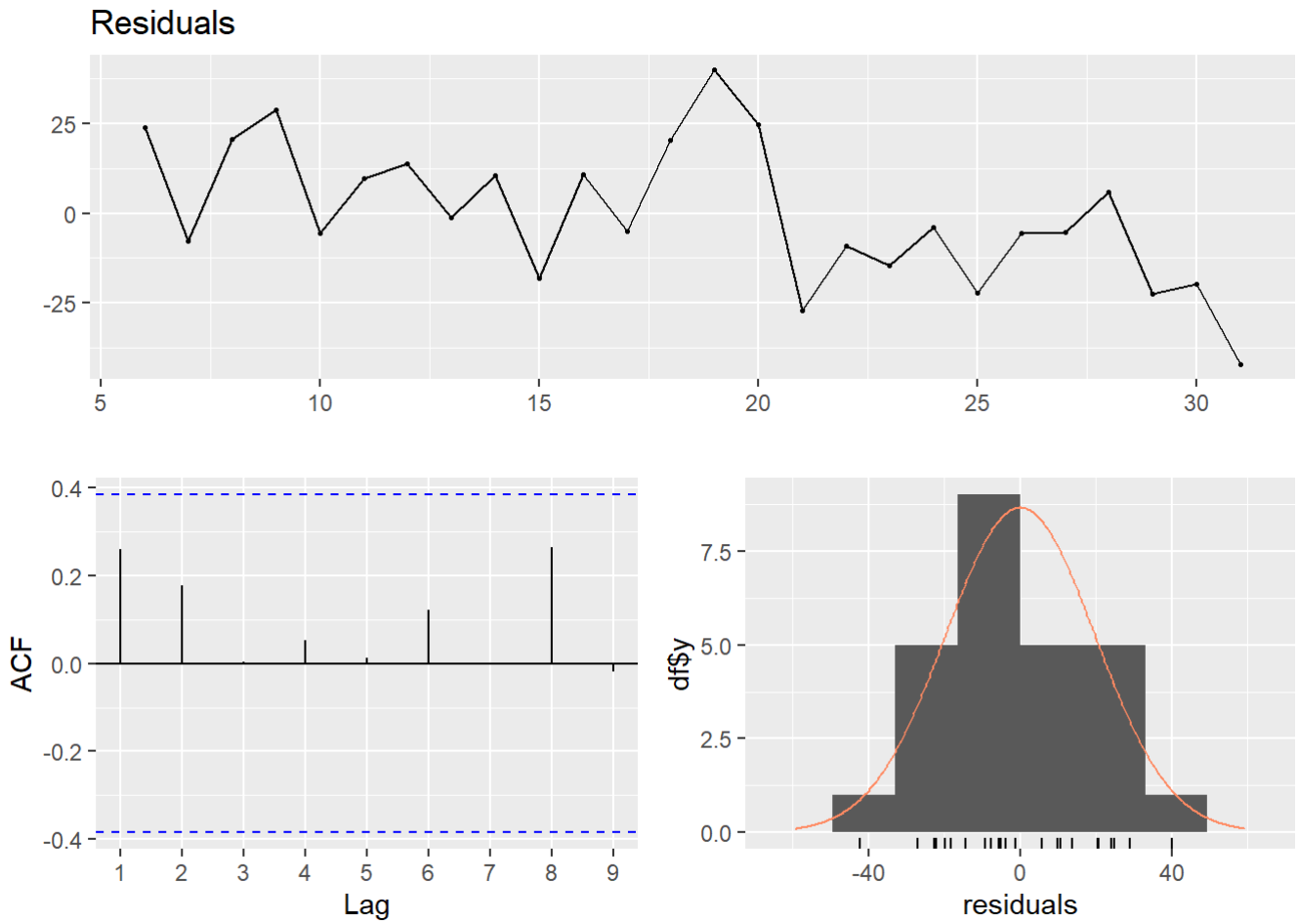
Here we have chosen **model.rainfallardl** instead of **model.temperatureardl** due to better significant coefficients and Adjusted R-squared value.

From the outputs of **model.rainfallardl** model and its summary :

- The p-values is reported at 0.301 which is much greater than 5% level of significance.
- The adjusted R-squared value reported was 0.03508 which is very poor.
- None of the coefficients are significant.
- The Wu-Hausman test is reported at 0.30 on 1 and 26 Degrees of Freedom.

Residual Analysis

```
checkresiduals(model.rainfallardl$model, test=FALSE)
```



From the outputs of Model "**model.rainfallardl**" residuals :

- The residuals appear to be distributed randomly with exhibiting a downward trend.
- The ACF plot reveals that there is seasonality and correlation exist in residual due to significant lag present in the ACF.
- The Histogram is unevenly distributed overall.
- The time-series plot of residual does seems to follow a downward trend overall.

So, we can conclude that the model "**model.rainfallardl**" is a decent model with a appropriate composition of residuals and a MASE score of 0.845.

Forecasting the model **model.rainfallardl**

```
# Producing point predictions utilizing the radiation values from task2_covariate
forecastFFDpoint_3 <- dLagM::forecast(model.rainfall,x = c(1.60,1.56,1.79,1.69) ,h = 4)

forecastFFDpoint_3 <- round(forecastFFDpoint_3$forecasts,2)

# Generating prediction intervals of forecast

#forecastFFDinterval_3 <- forecast(model = model.rainfall, x = c(14.60,14.56,14.79,14.69,14.
4,14.6,14.7,14.2,14.83) , h = 4 ,interval = TRUE)

#round(forecastFFDinterval_3$forecasts,2)
```

Plotting the Forecast

```
plot(ts(c(as.vector(FFD),forecastFFDpoint_3),start=1984),type="o",col="black", ylab="FFD_fore
casted",
main="Figure 39: FFD four year ahead predicted values from 2015-2018")
lines(ts(c(as.vector(FFD),forecastFFDpoint_3),start=1984),type="o",col="green")
legend("topleft", lty = 1,pch=1,text.width=11, col = c("black","green"), c("Data","model.rain
fallardl"))
```

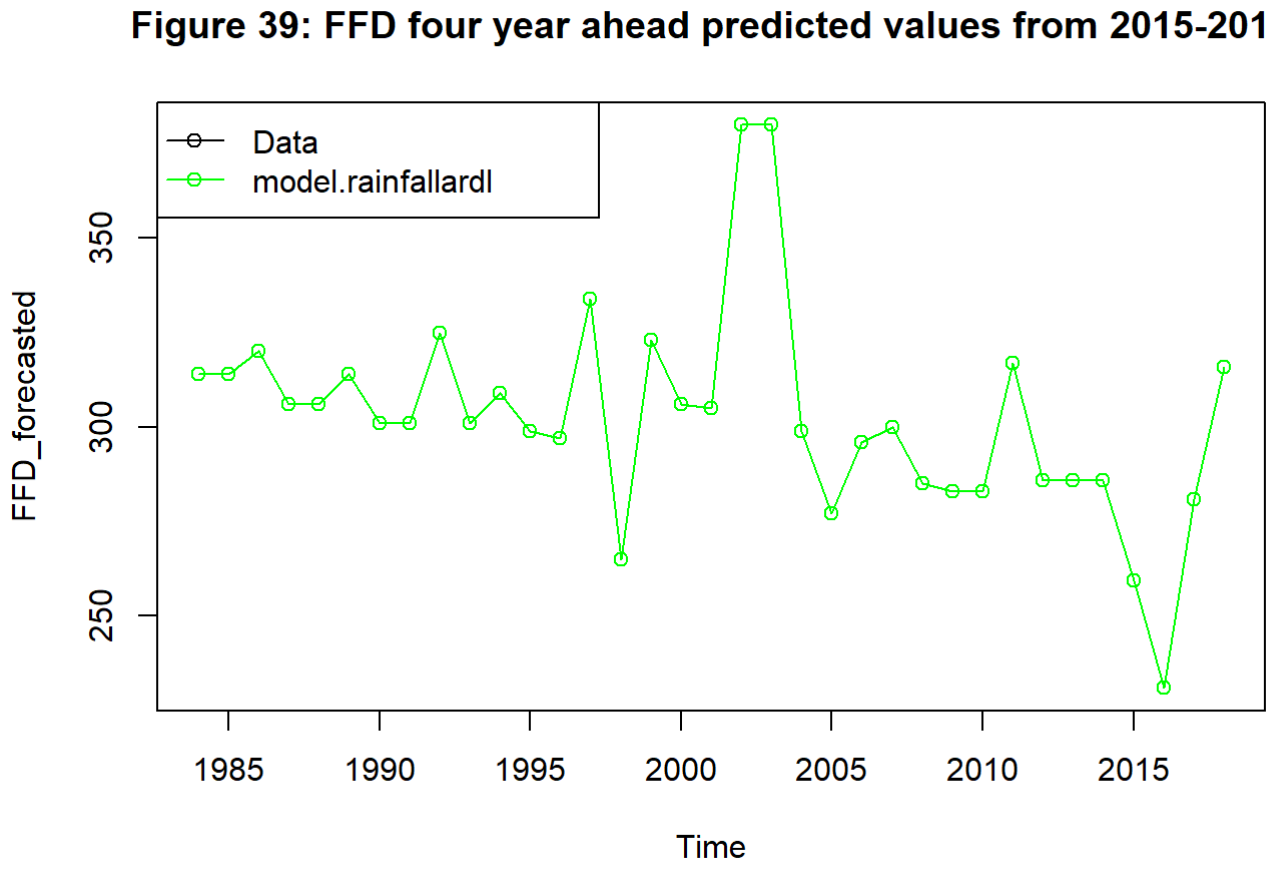


Figure 39

The model **model.rainfallardl** aligns well with the FFD values spanning from 1984 to 2014. As a result, forecasts produced by the ARDL model can be deemed trustworthy and precise, even if the projected high variations for the upcoming four years appear somewhat improbable.

Modelling using Dynamic Linear Models

Dynamic Linear Models are frequently favored for prediction tasks. Given that the FFD series doesn't exhibit any trend or seasonal patterns, our focus will be on fitting models with singular independent variables, both with and without an intercept. Let's proceed to construct several dynlm models.

```
# Fitting univariate models with FFD

dynamicmodel1 <-dynlm(FFD~ L(FFD , k = 1 ))
dynamicmodel2 <-dynlm(FFD~ L(FFD , k = 1 )+ L(FFD , k = 2 ))
dynamicmodel3 <-dynlm(FFD~ L(FFD , k = 1 )+ L(FFD , k = 2 )+L(FFD , k = 3 ))



# Fitting models with temperature as independent variable

temperaturedynlm <-dynlm(FFD ~ temperature + L(FFD, k=1))
temperaturedynlm1 <-dynlm(FFD ~ temperature + L(FFD, k=2))
temperaturedynlm2 <-dynlm(FFD ~ temperature)
temperaturedynlm3 <-dynlm(FFD ~ temperature - 1)
temperaturedynlm4 <-dynlm(FFD ~ temperature + L(temperature, k=1))
temperaturedynlm5 <-dynlm(FFD ~ temperature + L(temperature, k=2))
temperaturedynlm6 <-dynlm(FFD ~ temperature + L(FFD, k=3))
temperaturedynlm7 <-dynlm(FFD ~ temperature + L(temperature, k=3))

# Fitting models with rainfall as independent variable

rainfalldynlm <-dynlm(FFD ~ rainfall + L(FFD, k=1))
rainfalldynlm1 <-dynlm(FFD ~ rainfall + L(FFD, k=2))
rainfalldynlm2 <-dynlm(FFD ~ rainfall)
rainfalldynlm3 <-dynlm(FFD ~ rainfall - 1)
rainfalldynlm4 <-dynlm(FFD ~ rainfall + L(rainfall, k=1))
rainfalldynlm5 <-dynlm(FFD ~ rainfall + L(rainfall, k=2))
rainfalldynlm6 <-dynlm(FFD ~ rainfall + L(rainfall, k=3))
rainfalldynlm7 <-dynlm(FFD ~ rainfall + L(FFD, k=3))


# Fitting models with humidity as independent variable

humiditydynlm <- dynlm(FFD ~ humidity + L(FFD, k=1))
humiditydynlm1 <- dynlm(FFD ~ humidity + L(FFD, k=2))
humiditydynlm2 <- dynlm(FFD ~ humidity)
humiditydynlm3 <- dynlm(FFD ~ humidity - 1)
humiditydynlm4 <- dynlm(FFD ~ humidity + L(humidity, k=1))
humiditydynlm5 <- dynlm(FFD ~ humidity + L(humidity, k=2))
humiditydynlm6 <- dynlm(FFD ~ humidity + L(humidity, k=3))
humiditydynlm7 <- dynlm(FFD ~ humidity + L(FFD, k=3))

# Fitting models with radiation as independent variable

radiationdynlm <- dynlm(FFD ~ radiation + L(FFD, k=1))
radiationdynlm1 <- dynlm(FFD ~ radiation + L(FFD, k=2))
radiationdynlm2 <- dynlm(FFD ~ radiation)
radiationdynlm3 <- dynlm(FFD ~ radiation - 1)
radiationdynlm4 <- dynlm(FFD ~ radiation + L(radiation, k=1))
radiationdynlm5 <- dynlm(FFD ~ radiation + L(radiation, k=2))
radiationdynlm6 <- dynlm(FFD ~ radiation + L(radiation, k=3))
radiationdynlm7 <- dynlm(FFD ~ radiation + L(FFD, k=3))
```

Evaluating models based on MASE scores

```
dynlm_MASE2 <- MASE(lm(temperaturedynlm), lm(temperaturedynlm1), lm(temperaturedynlm2), lm(te
mperaturedynlm3), lm(temperaturedynlm4),lm(temperaturedynlm5),lm(temperaturedynlm6),lm(temper
aturedynlm7),lm(rainfalldynlm),lm(rainfalldynlm1),lm(rainfalldynlm2),lm(rainfalldynlm3),
lm(rainfalldynlm4),lm(rainfalldynlm5),lm(rainfalldynlm6),lm(rainfalldynlm7),lm(humiditydynl
m),lm(humiditydynlm1),lm(humiditydynlm2),lm(humiditydynlm3),lm(humiditydynlm4),lm(humiditydyn
lm5),lm(humiditydynlm6),lm(humiditydynlm7),lm(radiationdynlm),lm(radiationdynlm1),lm(radiatio
ndynlm2),lm(radiationdynlm3),lm(radiationdynlm4),lm(radiationdynlm5),lm(radiationdynlm6),lm(r
adiationdynlm7),lm(dynamicmodel1),lm(dynamicmodel2),lm(dynamicmodel3))

arrange(dynlm_MASE2,MASE)
```

```
##                         n      MASE
## lm(temperaturedynlm1) 29 0.8189541
## lm(rainfalldynlm6)    28 0.8200430
## lm(temperaturedynlm4) 30 0.8230056
## lm(rainfalldynlm7)    28 0.8240797
## lm(rainfalldynlm1)    29 0.8258225
## lm(humiditydynlm4)    30 0.8304564
## lm(humiditydynlm7)    28 0.8334884
## lm(humiditydynlm6)    28 0.8337672
## lm(temperaturedynlm6) 28 0.8368777
## lm(radiationdynlm7)   28 0.8373167
## lm(humiditydynlm3)    31 0.8374536
## lm(rainfalldynlm5)    29 0.8375900
## lm(radiationdynlm6)   28 0.8377941
## lm(humiditydynlm5)    29 0.8378424
## lm(temperaturedynlm2) 31 0.8383751
## lm(temperaturedynlm)  30 0.8389872
## lm(rainfalldynlm)     30 0.8413683
## lm(rainfalldynlm4)    30 0.8416367
## lm(humiditydynlm1)    29 0.8423076
## lm(radiationdynlm5)   29 0.8466401
## lm(radiationdynlm1)   29 0.8502380
## lm(temperaturedynlm7) 28 0.8516146
## lm(radiationdynlm4)   30 0.8537868
## lm(temperaturedynlm5) 29 0.8541180
## lm(radiationdynlm)    30 0.8597834
## lm(humiditydynlm2)    31 0.8602236
## lm(dynamicmodel1)     30 0.8624381
## lm(humiditydynlm)     30 0.8624882
## lm(rainfalldynlm2)    31 0.8633473
## lm(dynamicmodel2)     29 0.8664225
## lm(dynamicmodel3)     28 0.8726962
## lm(radiationdynlm2)   31 0.8762647
## lm(radiationdynlm3)   31 0.9715703
## lm(temperaturedynlm3) 31 1.1300789
## lm(rainfalldynlm3)    31 2.2013932
```

Analysing and summarising best model

dynamicmodel3

```
summary(dynamicmodel3)
```

```
##
## Time series regression with "ts" data:
## Start = 1987, End = 2014
##
## Call:
## dynlm(formula = FFD ~ L(FFD, k = 1) + L(FFD, k = 2) + L(FFD,
##     k = 3))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.022 -16.279  -6.006   7.874  73.949
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   233.62078   95.29633   2.452   0.0219 *
## L(FFD, k = 1)   0.29138    0.20438   1.426   0.1669
## L(FFD, k = 2)  -0.12374    0.21171  -0.584   0.5644
## L(FFD, k = 3)   0.06633    0.20843   0.318   0.7531
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.4 on 24 degrees of freedom
## Multiple R-squared:  0.08013,    Adjusted R-squared:  -0.03486
## F-statistic: 0.6968 on 3 and 24 DF,  p-value: 0.5631
```

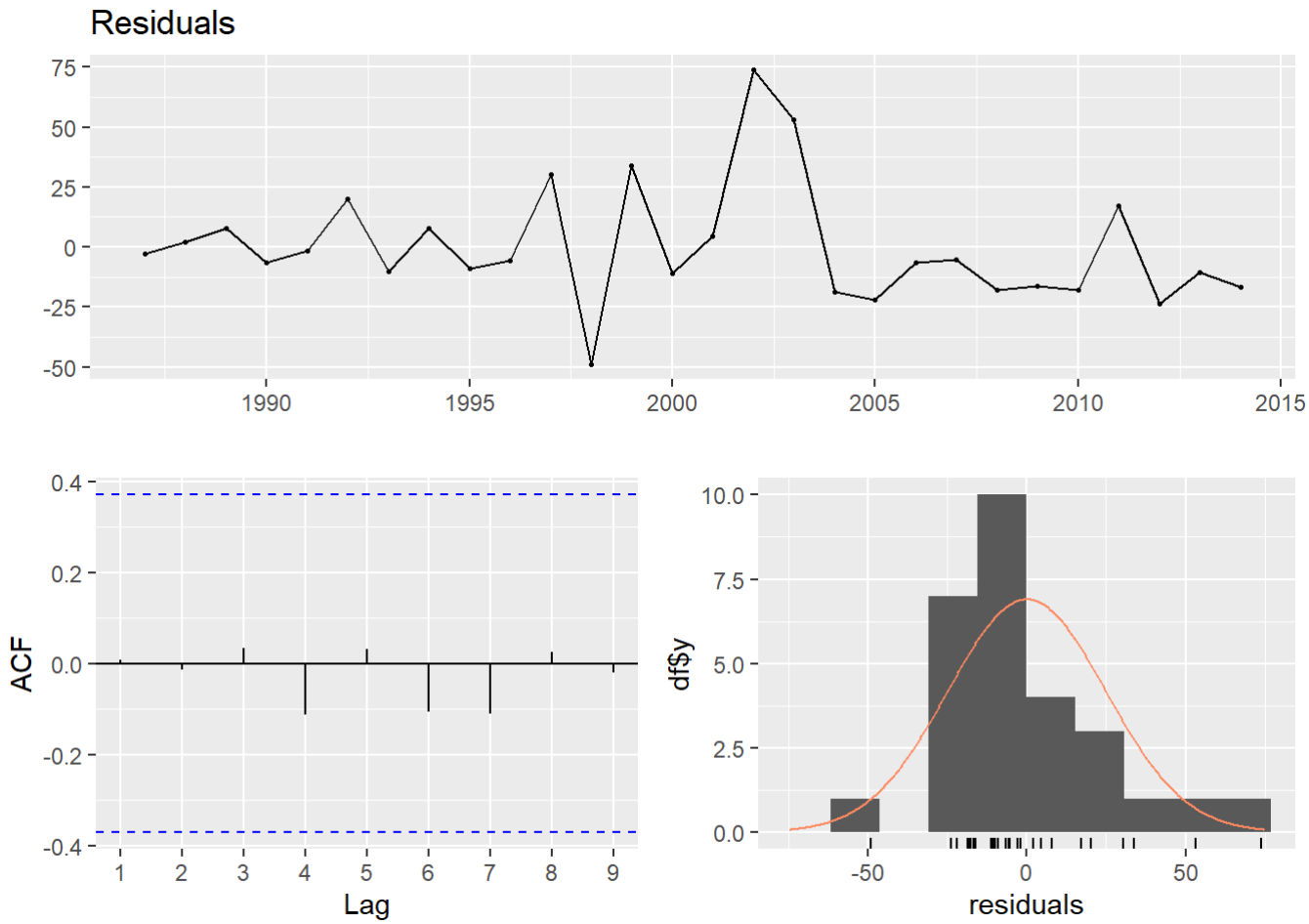
Here we have chosen **dynamicmodel3** instead of **temperaturedynlm1** due to better significant coefficients and adjusted R-squared value.

From the outputs of **model.rainfallardl** model and its summary :

- The p-values is reported at 0.301 which is much greater than 5% level of significance.
- The adjusted R-squared value reported was 0.03508 which is very poor.
- None of the coefficients are significant.
- The Wu-Hausman test is reported at 0.30 on 1 and 26 Degrees of Freedom.

Residual Analysis

```
checkresiduals(dynamicmodel3)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 7
##
## data:  Residuals
## LM test = 1.3154, df = 7, p-value = 0.988
```

From the outputs of Model "**dynamicmodel3**" residuals :

- The residuals appear to be distributed randomly with exhibiting a downward trend.
- The ACF plot reveals that there is seasonality and correlation exist in residual due to significant lag present in the ACF.
- The Histogram is unevenly distributed overall.
- The time-series plot of residual does seems to follow a downward trend overall.

So, we can conclude that the model "**dynamicmodel3**" is a decent model with a appropriate composition of residuals and a MASE score of 0.818.

Fitting the model **dynamicmodel3**

```
par(mfrow=c(1,1))
plot(FFD,ylab='FFD',xlab='Year',main = "Figure 40: Fit with original FFD", col="blue")
lines(dynamicmodel3$fitted.values,col="black")
legend("topleft", lty = 1,pch=1,text.width=11, col = c("black","blue"), c("Data","dynamicmode
l3"))
```

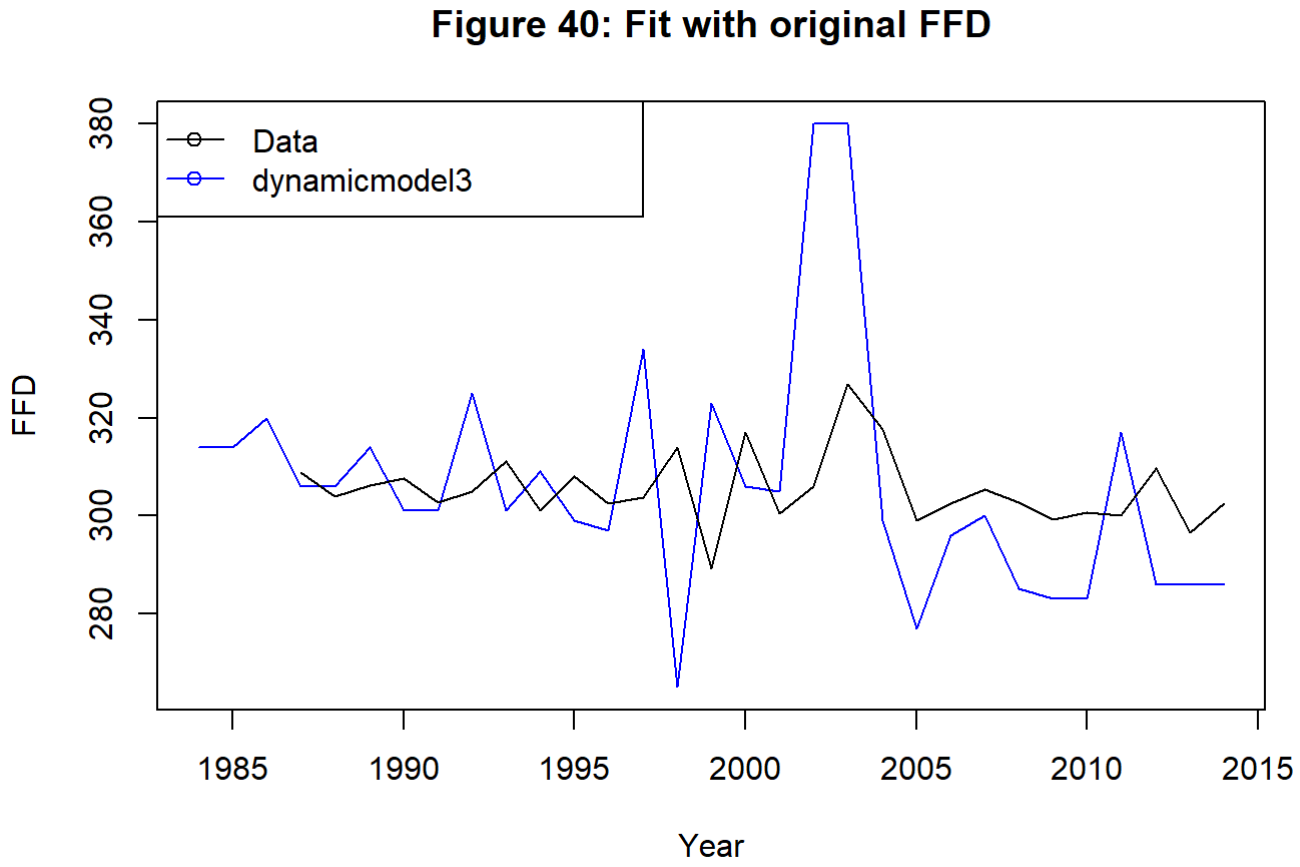


Figure 40

Forecasting the model **dynamicmodel3**

```
q = 4
n = nrow(dynamicmodel3$model)
FFD.frc = array(NA , (n + q))
FFD.frc[1:n] = FFD[4:length(FFD)]

for(i in 1:q){
data.new =c(1,FFD.frc[n-1+i],FFD.frc[n-2+i],FFD.frc[n-3+i])
FFD.frc[n+i] = as.vector(dynamicmodel3$coefficients) %*% data.new
}

plot(FFD,xlim=c(1984,2018),
ylab='RBO',xlab='Year',
main = "Figure-41: Predicted FFD values using dynamicmodel3")
lines(ts(FFD.frc[(n+1):(n+q)],start=c(2015)),col="blue")
```

Figure-41: Predicted FFD values using dynamicmodel3

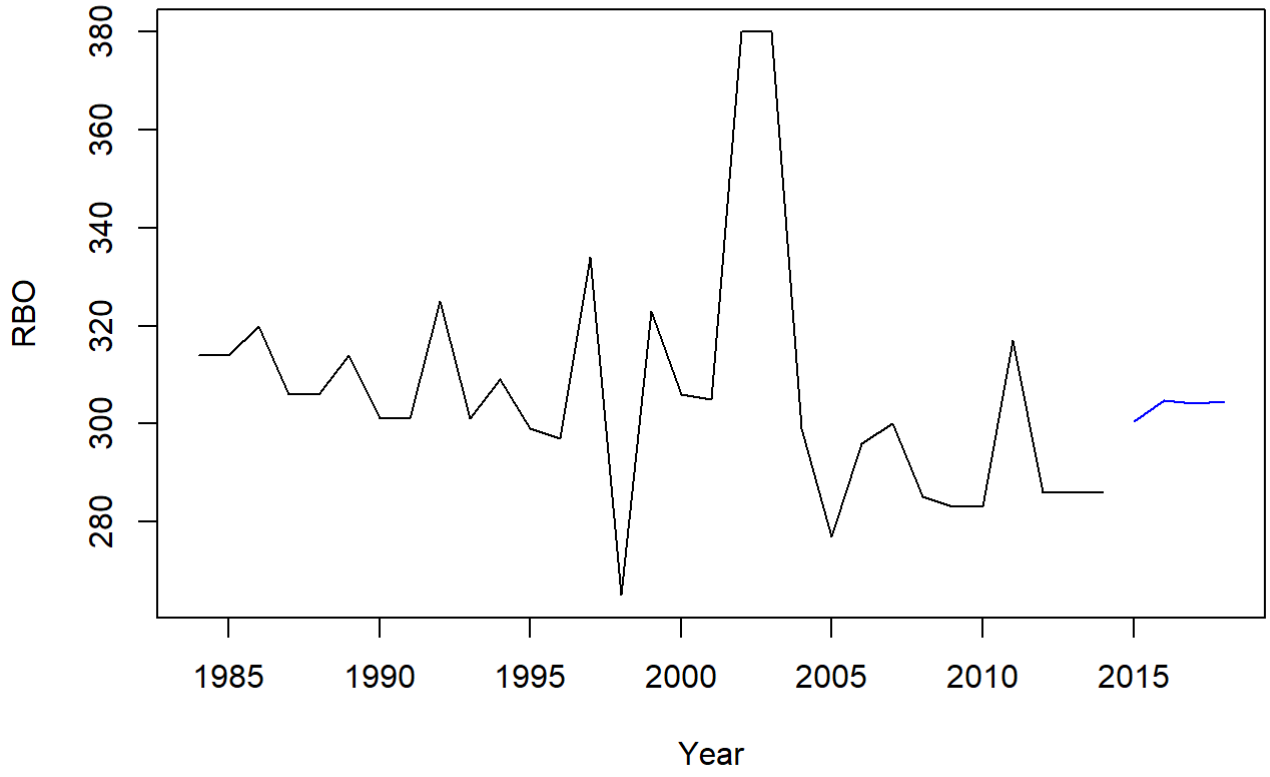


Figure 41

```
# Analusing point forecast

FFD.frc[(n+1):(n+q)]
```

```
## [1] 300.5352 304.7704 304.2059 304.4814
```

Based on the plotted predictions and our calculations, it's evident that the forecasted values for the upcoming four years will remain relatively stable, ranging between 300-305. However, given that the dynamic model wasn't a good fit for the original data series, we should approach these predicted values with caution, as they may not be trustworthy.

Modelling using Exponential Smoothing

The FFD series lacks both trend and seasonality, limiting our options to specific models when employing exponential smoothing techniques.

Fitting basic ETS model

```
# Computing values of alpha and beta

model.FFDses <-ses(FFD, initial="simple", h=4)
```

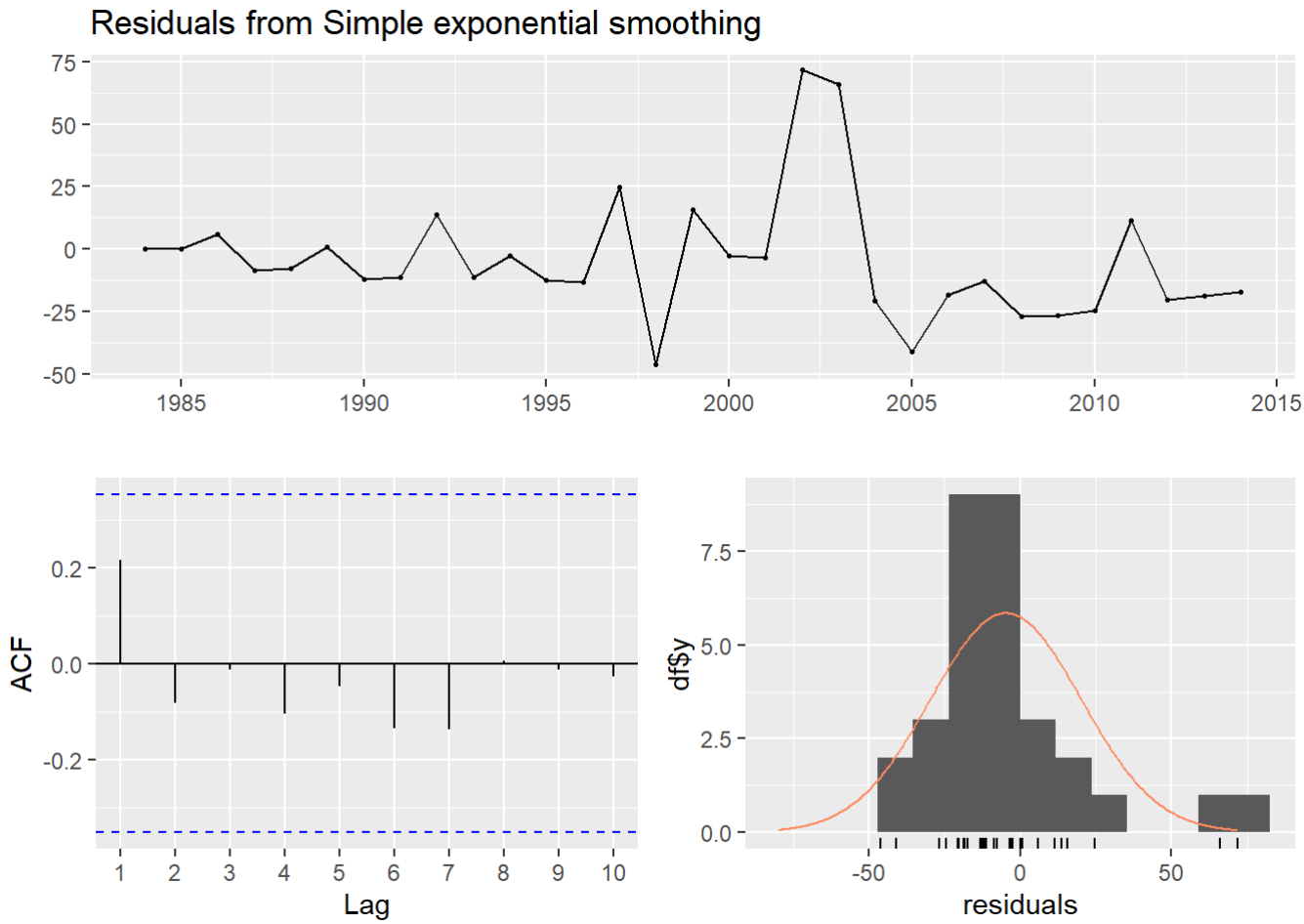
Analyzing and Summarising model

```
summary(model.FFDses)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = FFD, h = 4, initial = "simple")
##
##   Smoothing parameters:
##     alpha = 0.0821
##
##   Initial states:
##     l = 314
##
##   sigma:  25.0406
## Error measures:
##                     ME     RMSE      MAE       MPE    MAPE      MASE     ACF1
## Training set -4.795374 25.04059 18.35388 -2.150391 5.92343 0.9592619 0.215416
##
## Forecasts:
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2015       301.7998 269.7090 333.8906 252.7211 350.8785
## 2016       301.7998 269.6011 333.9985 252.5561 351.0435
## 2017       301.7998 269.4936 334.1060 252.3917 351.2079
## 2018       301.7998 269.3864 334.2132 252.2278 351.3718
```

Residual analysis of model

```
checkresiduals(model.FFDses)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Simple exponential smoothing
## Q* = 3.069, df = 6, p-value = 0.8001
##
## Model df: 0.   Total lags used: 6
```

Based on the summary and residual visualizations of the model.FFDses model, we can derive the following observations:

- The model has a MASE value of 0.954609. Other accuracy metrics include an RMSE of 25.0236 and a MAPE of 5.96345.
- The model's smoothing parameter, alpha, is set to 0.
- The Ljung-Box test yields a p-value above the 5% significance level, suggesting that there's no autocorrelation within the residuals.
- A single significant lag observed in the ACF plot suggests some degree of autocorrelation in the residuals.
- The residuals appear to be scattered without any discernible trend or pattern.
- There appear to be issues with normality in the model, as the residuals don't seem to adhere to a normal distribution.
- The model's forecasts are provided, accompanied by both 80% and 95% confidence intervals.

Forecasting the model

Plotting the model with forecast

```
plot(FFD, fcol = "black", main = "Figure 42: FFD series with four years ahead forecasts", yla
b = "FFD")
lines(fitted(model.FFDses), col = "green")
lines(model.FFDses$mean, col = "green", lwd = 2)
legend("topleft", lty = 1, col = c("black", "green"), c("Data", "FFDses"))
```

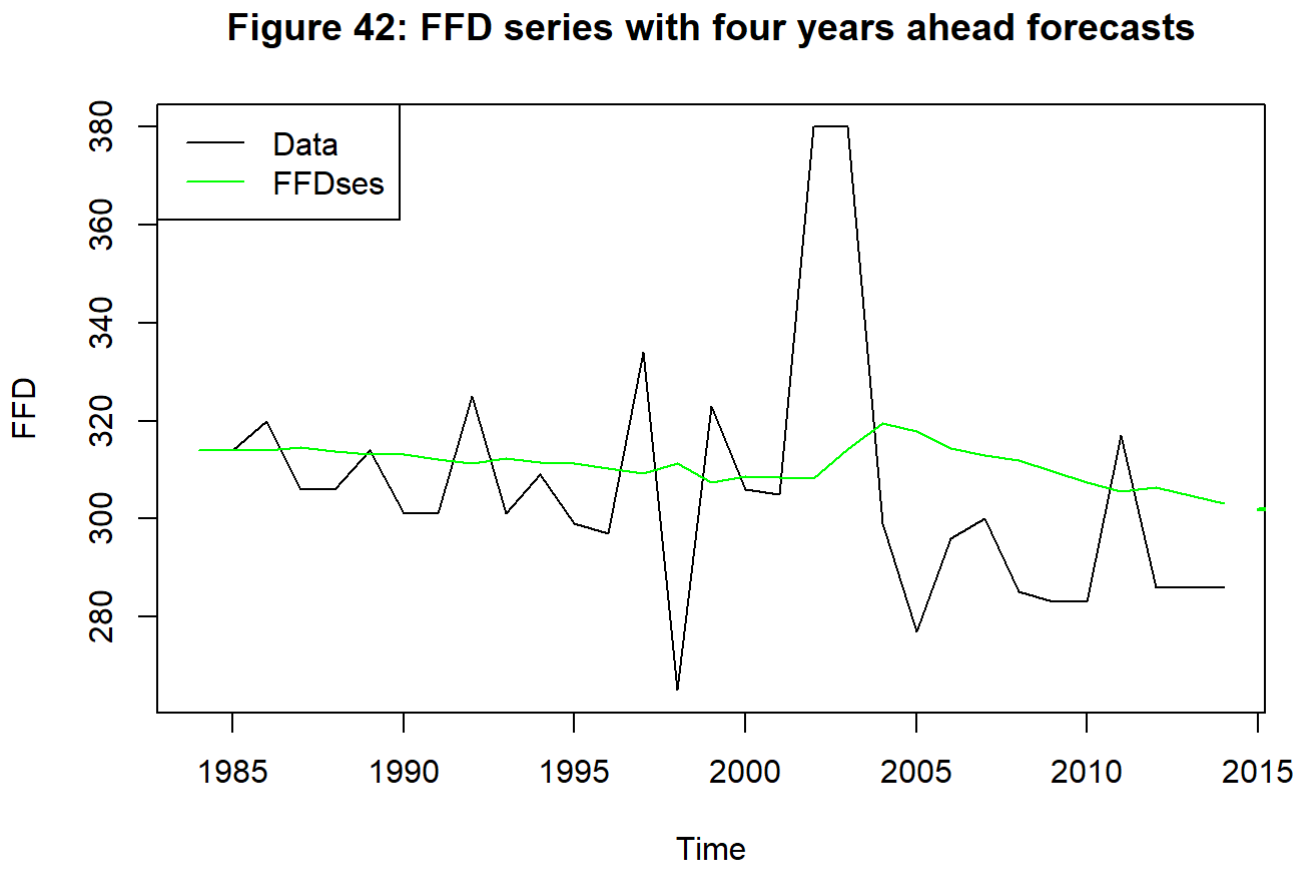


Figure 42

We will now fit the forecasted values to the model.

```
# Value adding
frc <- model.FFDses$mean
ub <- model.FFDses$upper[,2]
lb <- model.FFDses$lower[,2]
forecasts <- ts.intersect(ts(lb, start = c(2015,1), frequency = 1), ts(frc,start = c(2015,1),
frequency = 1), ts(ub,start = c(2015,1), frequency = 1))
colnames(forecasts) <- c("Lower bound", "Point forecast", "Upper bound")

forecasts
```

```
## Time Series:
## Start = 2015
## End = 2018
## Frequency = 1
##      Lower bound Point forecast Upper bound
## 2015    252.7211       301.7998    350.8785
## 2016    252.5561       301.7998    351.0435
## 2017    252.3917       301.7998    351.2079
## 2018    252.2278       301.7998    351.3718
```

From the above result, we can interpret that the basic Exponential Smoothing model doesn't align well with the observed and projected values. It produces a flat trend and offers nearly uniform predictions for the upcoming four years. Therefore, this model isn't suitable for forecasting FFD values over the next four years.

Modelling with State Space Models

While the model doesn't exhibit any trend or seasonality, we can explore models with both multiplicative and additive errors. Let's proceed to fit these models and evaluate their performance.

Fitting ETS(M,N,N) with multiplicative type

```
FFDets1 <- ets(FFD, model="MNN")
```

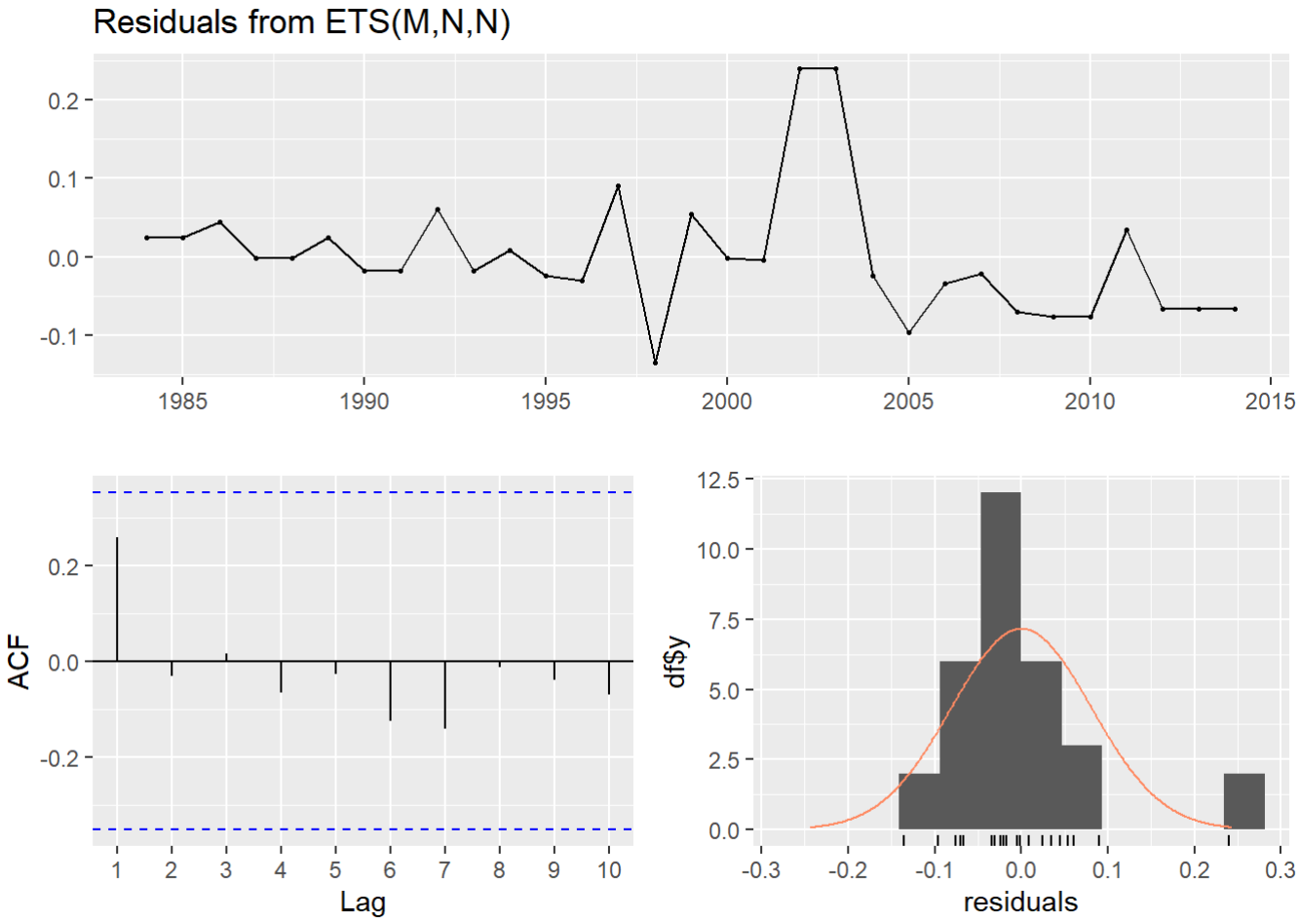
Analysing and Summarising the model

```
summary(FFDets1)
```

```
## ETS(M,N,N)
##
## Call:
##  ets(y = FFD, model = "MNN")
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 306.3826
##
##   sigma:  0.0825
##
##      AIC     AICc      BIC
## 310.6132 311.5021 314.9152
##
## Training set error measures:
##                        ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.0009438502 24.43768 16.75958 -0.5805143 5.329407 0.8759362
##                   ACF1
## Training set 0.2589561
```

Residual analysis

```
checkresiduals(FFDets1)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,N)
## Q* = 3.1718, df = 6, p-value = 0.787
##
## Model df: 0.   Total lags used: 6
```

Based on the summary and residual visualizations of FFDets1, we can infer the following:

- The model's MASE score is 0.875936, while other accuracy indicators like AIC, AICc, and BIC hover between 262 and 266. These metrics suggest that the model's performance is comparable to prior models, making it an adequate but not exceptional fit.
- The model uses a smoothing parameter with alpha set to 1e-04.
- The Ljung-Box test yields a p-value above the 5% significance threshold, implying that the residuals does not exhibit serial correlation.
- Residual distribution appears to be random, but the ACF plot highlights one significant lag, reinforcing the findings from the Ljung-Box test.
- The model displays issues with normality, as the residuals don't align well with a normal distribution.

Fitting ETS(A,N,N) with additive type

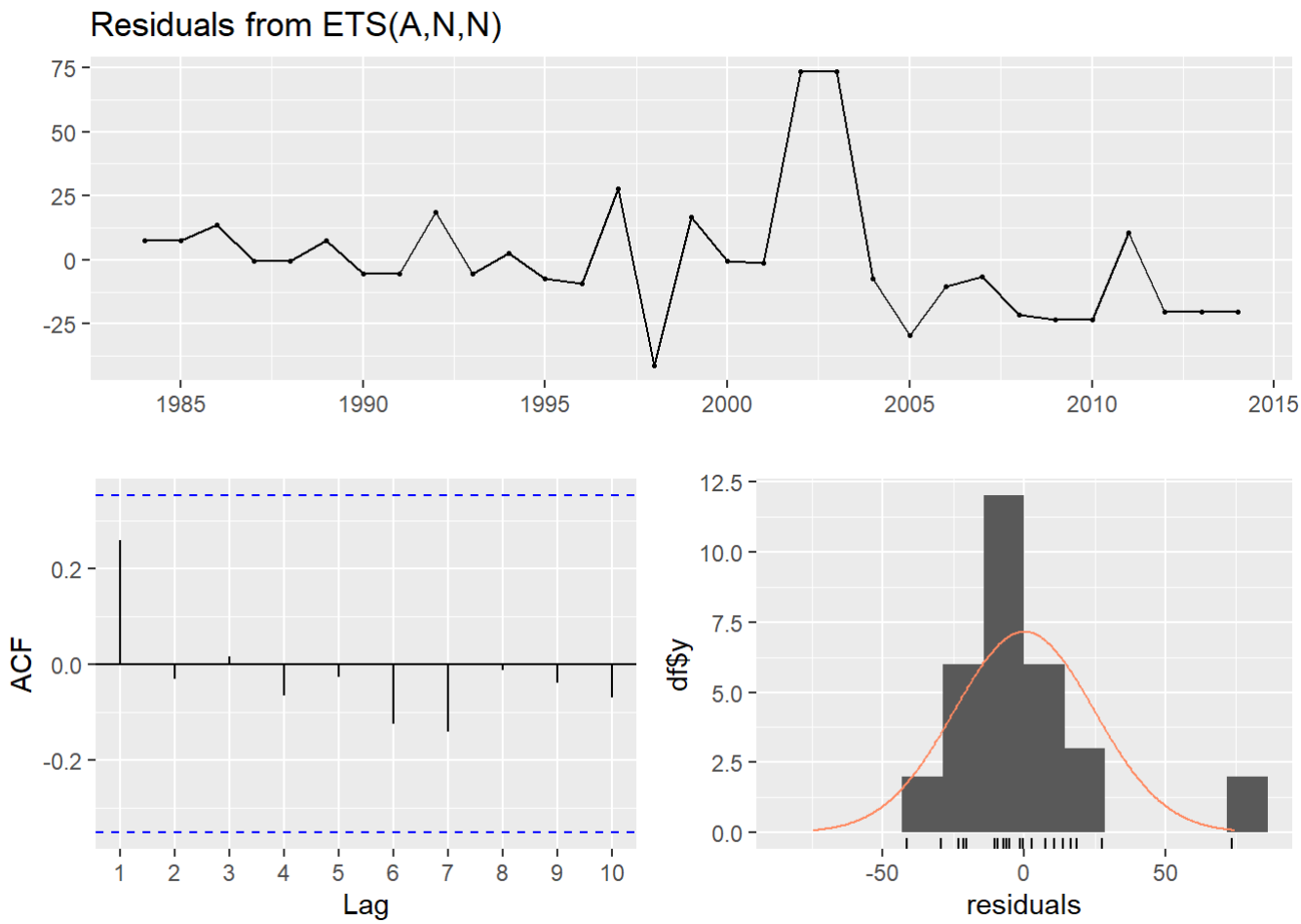
Analysing and Summarising the model

```
FFDets2 <- ets(FFD, model="ANN")
summary(FFDets2)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = FFD, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 306.3843
##
##   sigma:  25.2663
##
##      AIC     AICc      BIC
## 310.6134 311.5023 314.9154
##
## Training set error measures:
##                       ME     RMSE      MAE         MPE     MAPE      MASE
## Training set -0.002622643 24.43768 16.76007 -0.5810654 5.329595 0.8759617
##                    ACF1
## Training set 0.2589561
```

Residual Analysis

```
checkresiduals(FFDets2)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,N)
## Q* = 3.1719, df = 6, p-value = 0.787
##
## Model df: 0.   Total lags used: 6
```

Based on the summary and residual analysis of FFDets2, we can deduce the following:

- The model's MASE score is 0.875961, and its other accuracy metrics like AIC, AICc, and BIC fall between 310 and 314. These values mirror those from the previously fitted model, suggesting that while this model is reasonable, it is not optimal.
- The model utilizes a smoothing parameter with alpha = 1e-04.
- The Ljung-Box test indicates a p-value above the 5% threshold, suggesting the absence of serial correlation in the residuals.
- Residuals seem to be dispersed randomly, but the ACF plot displays a significant lag, aligning with the Ljung-Box test findings.
- There appears to be an issue with normality as the residuals don't follow a normal distribution.

In conclusion, both ETS models have an identical MASE value of 0.875961. However, when compared to other models, **this value isn't the lowest**, implying that the model may not be the best choice for predicting FFD.

Forecasting with respect to FFDets1 & FFDets2

Plotting the forecast model.

```
plot(FFD, fcol = "black", main = "Figure 43: FFD series with four years ahead forecasts", yla
b = "FFD")
lines(fitted(FFDets1), col = "green")
lines(FFDets1$mean, col = "green", lwd = 2)
lines(fitted(FFDets2), col = "darkmagenta")
lines(FFDets2$mean, col = "darkmagenta", lwd = 2)
legend("topleft", lty = 1, col = c("black", "green","darkmagenta"), c("Data", "FFDets1","FFDe
ts2"))
```

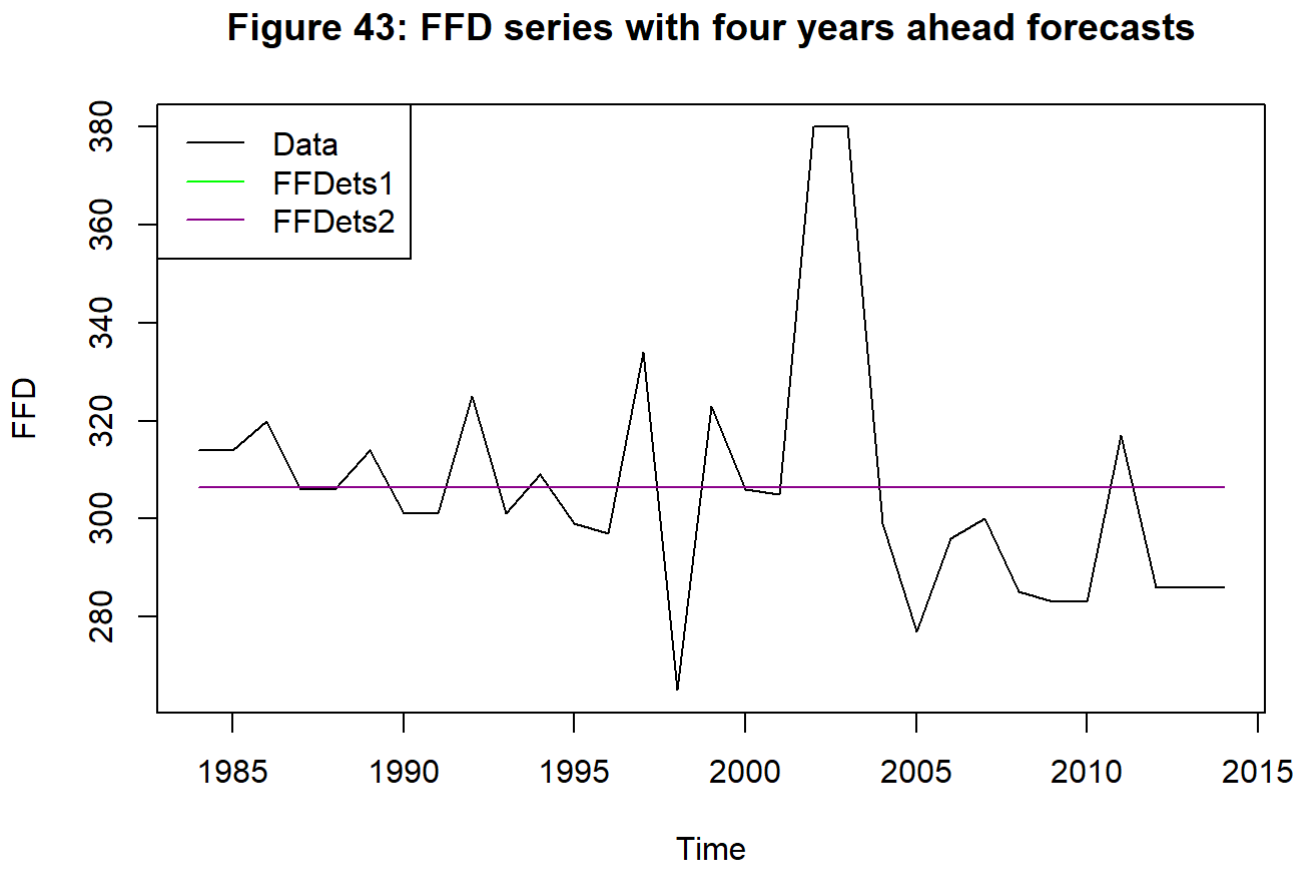


Figure 43

```
# Applied values to the models
frc <- model.FFDses$mean
ub <- model.FFDses$upper[,2]
lb <- model.FFDses$lower[,2]
forecasts <- ts.intersect(ts(lb, start = c(2015,1), frequency = 1), ts(frc,start = c(2015,1),
frequency = 1), ts(ub,start = c(2015,1), frequency = 1))
colnames(forecasts) <- c("Lower bound", "Point forecast", "Upper bound")

forecasts
```

```
## Time Series:
## Start = 2015
## End = 2018
## Frequency = 1
##      Lower bound Point forecast Upper bound
## 2015    252.7211       301.7998    350.8785
## 2016    252.5561       301.7998    351.0435
## 2017    252.3917       301.7998    351.2079
## 2018    252.2278       301.7998    351.3718
```

Hence we can interpret that, while the MASE value for the two ETS models is identical at 0.875961, it doesn't fare the best when compared to other models. Furthermore, the predictions made by the ETS models are off the mark, suggesting that they don't align well with the data. As a result, these models are unsuitable for accurately forecasting FFD.

Overview of Task 2 Summary Table

Let's evaluate all the previously constructed models and determine the optimal one using the MASE score.

```
# Summarising the data models
overview_table2 <- data.frame(Model = c("model.finite2","model.radiationpoly","model.radiatio
nkoyck","model.rainfallardl","dynamicmodel3","model.FFDses","model.FFDets1","model.FFDets2"),
MASE= c(0.2319041,0.6355363,0.7540611,0.4143799,0.6549642,0.7446097,0.7584965,0.7584965))


# Comparing the models wrt MASE score

arrange(overview_table2,MASE)
```

```
##                  Model      MASE
## 1        model.finite2 0.2319041
## 2   model.rainfallardl 0.4143799
## 3  model.radiationpoly 0.6355363
## 4        dynamicmodel3 0.6549642
## 5         model.FFDses 0.7446097
## 6 model.radiationkoyck 0.7540611
## 7        model.FFDets1 0.7584965
## 8        model.FFDets2 0.7584965
```

Based on the provided summary, the **model.finite2**, which uses **radiation** as its predictor, stands out as the optimal model for predicting FFD values over the next four years, as indicated by its MASE.

Conclusion

The analysis for Task 2 was completed successfully on the five series contained in the FFD.csv file. Given that these series have a yearly frequency (less than 2), they couldn't be broken down for a more detailed analysis. These series were scrutinized, and various modeling techniques, including **DLM, koyck, polyDLM, dynlm, ES, and state space models**, were employed to establish univariate models, each considering a single climate indicator. Using accuracy metrics such as **R2, MASE, AIC, and BIC**, the most effective model from each method was identified for forecasting purposes. **Radiation** , **rainfall** and **temperature** emerged as significant climate drivers in the top-performing models.

The **finite, poly, koyck, ARDL, and dynlm** methods yielded reliable predictions, whereas the optimal models determined via Exponential smoothing and ETS were deemed unsuitable for the FFD series. Out of all the evaluated models, model.finite2, which uses radiation as its predictor, stood out in terms of **MASE score, R2 value, F-test**, and the **forecasts** produced for the upcoming four years.

Task 3

Part(a)- Time series analysis & forecast of RBO series

Data Description

Between 1983 and 2014, Hudson & Keatley (2021) analyzed the influence of prolonged climate variations in Victoria on the blooming sequence resemblance of 81 flora species. The yearly blooming sequence was determined using the Rank-based Order similarity metric. They gauged shifts in this sequence by comparing each year's flowering sequence to that of 1983, using the Relative Blooming Order (RBO) method. In this system, the species that bloom the earliest in a given year receive a ranking of 1, while the last to bloom is ranked 81.

The dataset 'RBO.csv' contains five time series, rooted in 1983 being the benchmark year for flowering order evaluations. These series encapsulate the RBO evaluations of the 81 plant species highlighted in Hudson & Keatley's study, alongside averaged annual climatic data spanning from 1984 to 2014. Each of these series has 31 data points. Besides the RBO metric, the other variables in the dataset include temperature, rainfall, radiation, and relative humidity.

Objectives and Methodology

The primary goal of Task 3 is to delve into the analysis of RBO and provide forecasts for the next three years. In sub-section (b), we will employ dynamic models to adjust the series to account for Australia's drought phase and subsequently forecast for the subsequent three years. The procedure will unfold as follows:

- Initially, we will undergo essential data preprocessing for all five series. The '**RBO.csv**' file will be scrutinized for any anomalies, missing or unique values, which will then be appropriately addressed. Every series within '**RBO.csv**' will be transformed into a time-series format, showcasing annual data for all series, setting the stage for a comprehensive time series analysis.
- We will also introduce the **covariate** file, which will play a role in the forecasting process.
- Subsequently, to grasp the inherent patterns and attributes of each time series, we will visualize them through individual and collective time series plots. Stationarity checks will be conducted on all five series using ACF, PACF plots, and Dicker-Fuller Unit tests.
- A correlation matrix will be employed to ascertain the inter-series relationships.
- Additionally, we will fit various distributed lag models to these series.
- Based on insights from prior analyses, different **DLMs** and **dynlms** will be adapted to the series.
- Evaluation metrics like R squared, AIC, BIC, and MASE will guide the selection of the most apt model or set of models to forecast the RBO values for the upcoming **three** years.

- For sub-section (b), to capture the impact of the drought phase on RBO, **dynamic linear models (dynlm)** will be employed. This involves transforming the series to reflect the effects of that period before fitting the *dynlm* models.

In conclusion, we will forecast the RBO values for the subsequent **three** years.

Importing the dataset RBO

```
RBO_series <- read_csv("C:/Users/Rakshit Chandna/OneDrive/Desktop/DataMain/Forecasting/RBO.cs
v",show_col_types = FALSE)
head(RBO_series)
```

```
## # A tibble: 6 × 6
##    Year   RBO Temperature Rainfall Radiation RelHumidity
##   <dbl> <dbl>       <dbl>    <dbl>     <dbl>       <dbl>
## 1  1984 0.755        18.7     2.49      14.9        93.9
## 2  1985 0.741        19.3     2.48      14.7        94.9
## 3  1986 0.842        18.6     2.42      14.5        94.1
## 4  1987 0.748        19.1     2.32      14.7        94.5
## 5  1988 0.798        20.4     2.47      14.7        94.1
## 6  1989 0.794        19.6     2.74      14.8        96.1
```

Importing Covariate File

```
task3_covariate <- read_csv("C:/Users/Rakshit Chandna/OneDrive/Desktop/DataMain/Forecasting/C
ovariate x-values for Task 3.csv")
```

```
## Rows: 6 Columns: 5
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (5): Year, Temperature, Rainfall, Radiation, RelHumidity
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(task3_covariate)
```

```
## # A tibble: 6 × 5
##    Year Temperature Rainfall Radiation RelHumidity
##   <dbl>       <dbl>    <dbl>     <dbl>       <dbl>
## 1  2015        20.7     2.27      14.6        94.4
## 2  2016        20.5     2.38      14.6        94.0
## 3  2017        20.5     2.26      14.8        95.0
## 4  2018        20.6     2.27      14.8        95.1
## 5    NA          NA       NA        NA          NA
## 6    NA          NA       NA        NA          NA
```

Checking class for both data series

```
class(RBO_series)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

```
class(task3_covariate)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

Removing extra columns from RBO_series

```
RBO_series <- RBO_series[,1:6]

head(RBO_series)
```

```
## # A tibble: 6 × 6
##    Year   RBO Temperature Rainfall Radiation RelHumidity
##   <dbl> <dbl>       <dbl>    <dbl>     <dbl>       <dbl>
## 1  1984 0.755        18.7     2.49      14.9        93.9
## 2  1985 0.741        19.3     2.48      14.7        94.9
## 3  1986 0.842        18.6     2.42      14.5        94.1
## 4  1987 0.748        19.1     2.32      14.7        94.5
## 5  1988 0.798        20.4     2.47      14.7        94.1
## 6  1989 0.794        19.6     2.74      14.8        96.1
```

Checking for missing and special values

```
colSums(is.na(RBO_series))
```

```
##        Year         RBO Temperature    Rainfall   Radiation RelHumidity
##           0           0           0           0           0           0
```

```
colSums(is.na(task3_covariate))
```

```
##        Year Temperature    Rainfall   Radiation RelHumidity
##           2           2           2           2           2
```

We observe that there are few missing values in covariate series, hence we will deal with them.

Removing NA values in task3_covariate

```
task3_covariate <- na.omit(task3_covariate)
colSums(is.na(task3_covariate))
```

```
##        Year Temperature    Rainfall   Radiation RelHumidity
##           0           0           0           0           0
```

We have now removed all the missing values and our data is ready to be converted in to time series format.

Converting each variable to time series

```
# Converting 'RBO_series' to time series and storing in 'climate2'

climate2 <- ts(RBO_series[,2:6], start = c(1984,1), frequency = 1)

# Converting Temperature from 'RBO_series' to time series and storing in 'temptr'

temptr <- ts(RBO_series$Temperature, start = c(1984,1), frequency = 1)


# Converting Rainfall from 'RBO_series' to time series and storing in 'rain'

rain <- ts(RBO_series$Rainfall, start = c(1984,1), frequency = 1)

# Converting Radiation from 'RBO_series' to time series and storing in 'radtn'

radtn <- ts(RBO_series$Radiation, start = c(1984,1), frequency = 1)


# Converting RelHumidity from 'RBO_series' to time series and storing in 'humid'

humid <- ts(RBO_series$RelHumidity, start = c(1984,1), frequency = 1)


# Converting RBO from 'RBO_series' to time series and storing in 'RBO'

RBO <- ts(RBO_series$RBO, start = c(1984,1), frequency = 1)


# Checking conversion has been successfully done for each series

class(climate2)
```

```
## [1] "mts"    "ts"     "matrix" "array"
```

```
cbind(c(class(temptr),class(rain),class(humid),class(radtn),class(RBO)))
```

```
##      [,1]
## [1,] "ts"
## [2,] "ts"
## [3,] "ts"
## [4,] "ts"
## [5,] "ts"
```

Data Exploration and Visualisation

We will now plot the converted Time Series Data for each of the five series variables and interpret their important characteristics using the **5 Bullet Points**:

- **Trend**
- **Seasonality**
- **Changing Variance**
- **Behavior**
- **Intervention Point**

Time-series plot of RBO series

```
plot(RBO, ylab='RBO', xlab='Time period', type='o', col='blue', main = 'Figure 44: Time-serie
s plot of change in yearly  values of RBO')
```

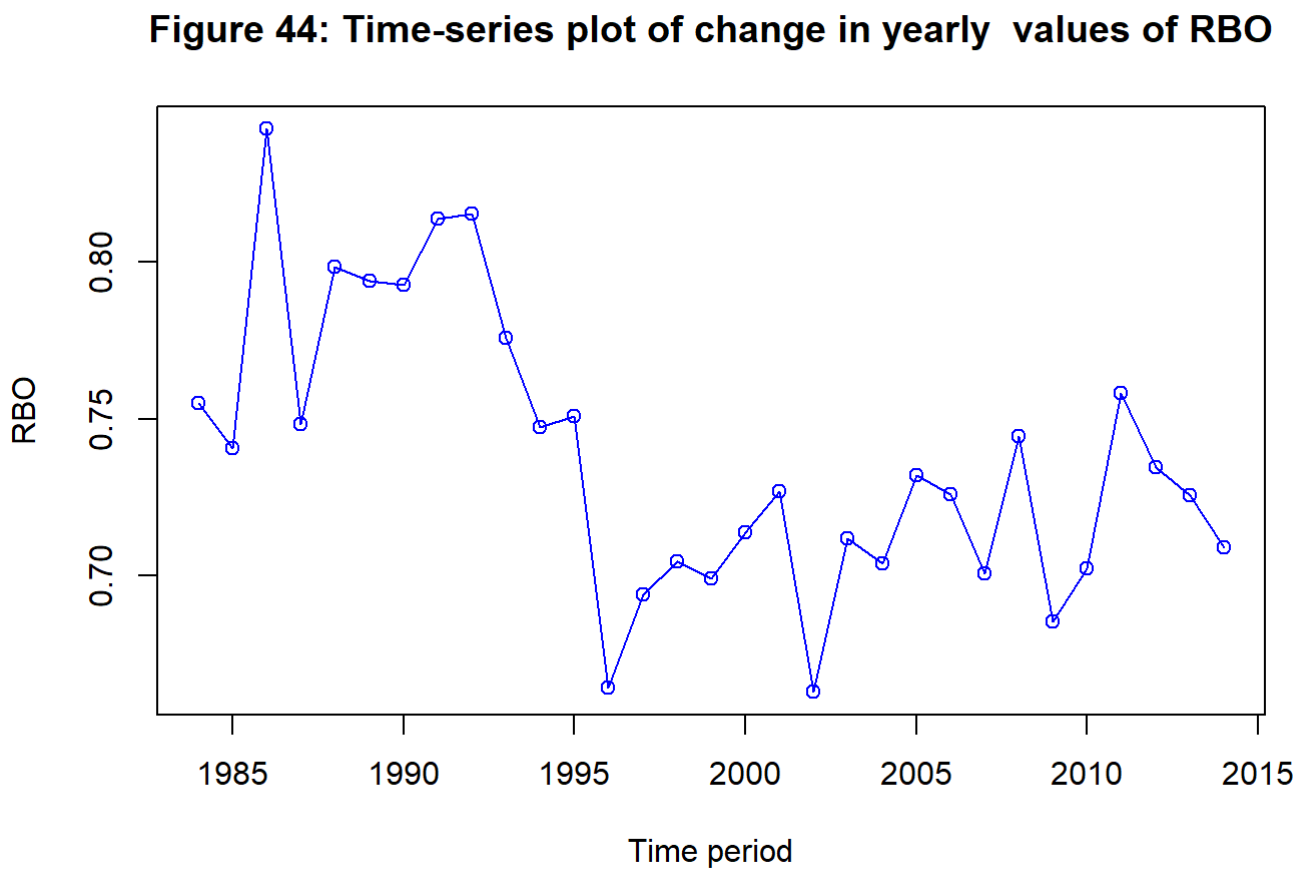


Figure 44

From the above plot, the Time Series plot shows us moderate level of **Seasonality** and **Downward Trend** with successive Auto regressive points and moving average overall from the year 1984 to 2014 respectively.

Checking **5 bullet points**:

- **Trend** - There is a Downward Trend present in the series.
- **Seasonality** - There is very moderate level of repeating patterns (seasonality) can be seen in the graph from year 1960-2014
- **Changing Variance** - There is a significant level of changing variance which is also in repeating patterns, can be seen if we look at the years 1995 and 2005.
- **Behavior** - A mixed Auto regressive and Moving average behavior can be seen in the graph.

- **Intervention point** - There is a sudden change point (drop) in year 1996 observed in the series.

Time-series plot of Temperature(temptr)

```
plot(temptr, ylab='Temperature(temptr)', xlab='Time period', type='o', col='red', main = 'Fig
ure 45: Time-series plot of change in yearly values of temperature')
```

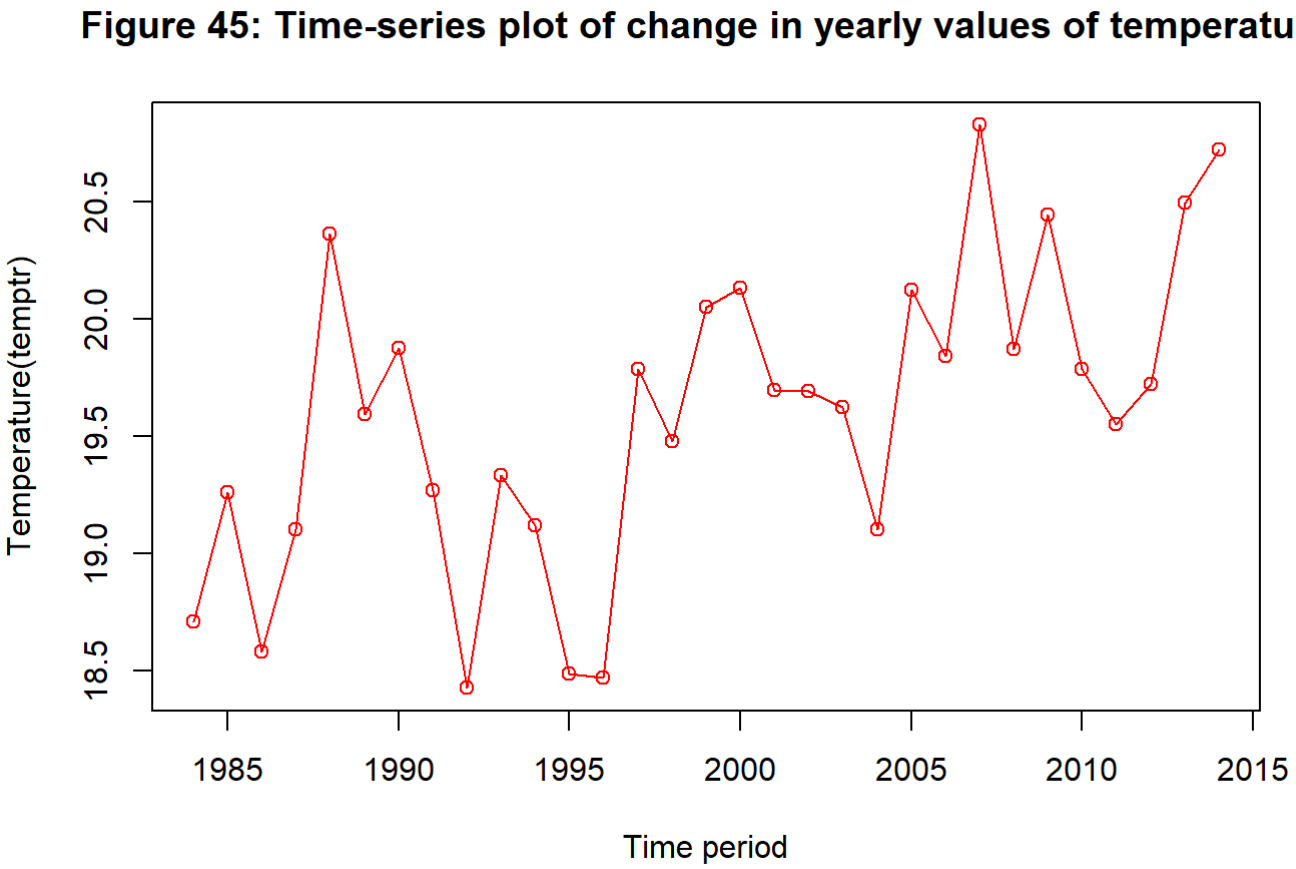


Figure 45

From the above plot, the Time Series plot shows us moderate level of **Seasonality** and **Upward Trend** with successive Auto regressive points and moving average overall from the year 1984 to 2014 respectively.

Checking **5 bullet points**:

- **Trend** - There is a Upward Trend present in the series.
- **Seasonality** - There is very moderate level of repeating patterns (seasonality) can be seen in the graph from year 1984-2014
- **Changing Variance** - There is a significant level of changing variance which is also in repeating patterns, can be seen if we look at the years 1995 and 2005.
- **Behavior** - A mixed Auto regressive and Moving average behavior can be seen in the graph.
- **Intervention point** - There is a sudden change point (drop) in year 1992 observed in the series.

Time-series plot of Rainfall(rain)

```
plot(rain, ylab='Rainfall(rain)', xlab='Time period', type='o', col='green', main = 'Figure 4
6: Time-series plot of change in yearly values of rainfall')
```

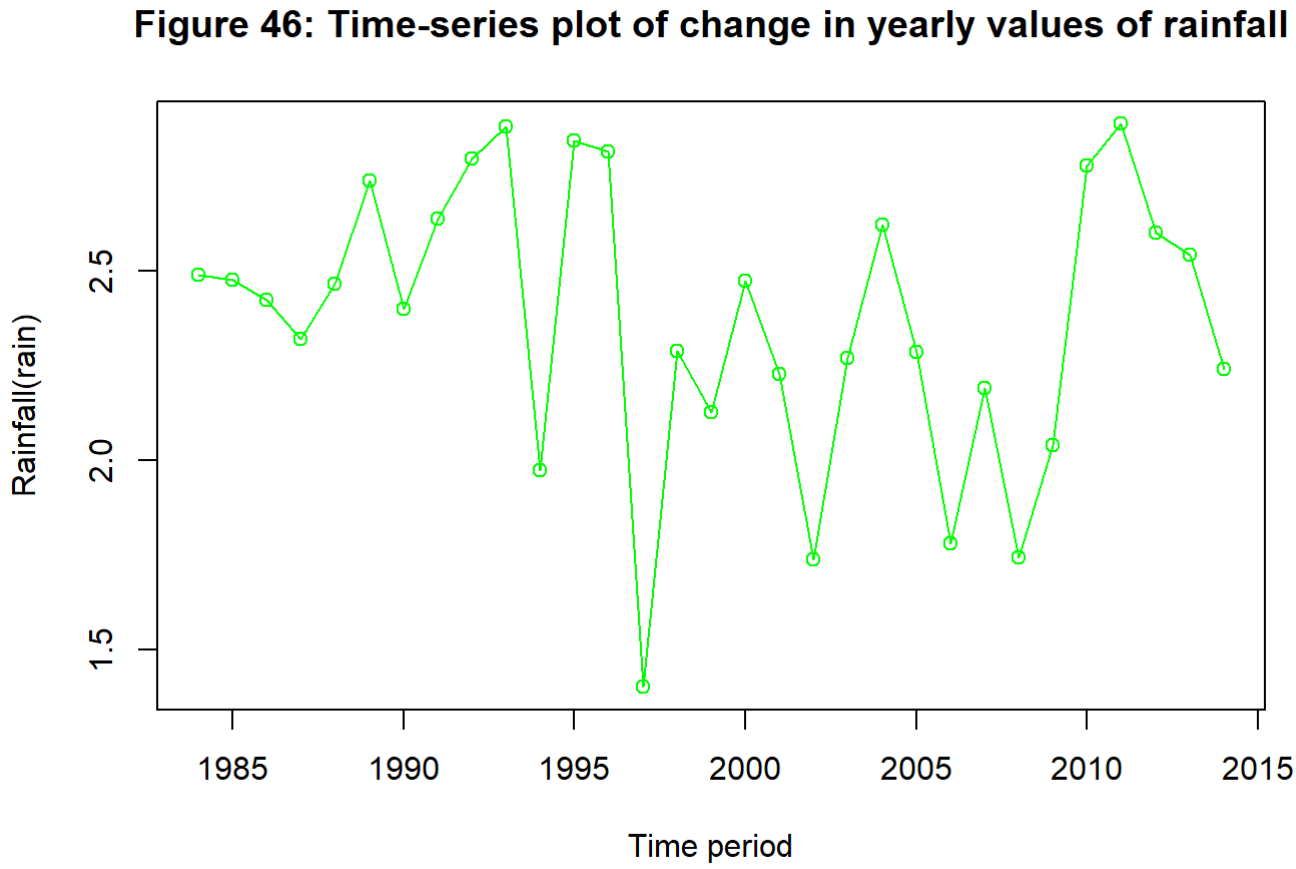


Figure 46

From the above plot, the Time Series plot shows us moderate level of **Seasonality** and **Nearly No Trend** with successive Auto regressive points and moving average overall from the year 1984 to 2014 respectively.

Checking **5 bullet points**:

- **Trend** - There is nearly No Trend present in the series.
- **Seasonality** - There is very high level of repeating patterns (seasonality) can be seen in the graph from year 1984-2014
- **Changing Variance** - There is a significant level of changing variance which is also in repeating patterns, can be seen if we look at the years 1995 and 2005.
- **Behavior** - A mixed Auto regressive and Moving average behavior can be seen in the graph.
- **Intervention point** - There is a sudden change point (drop) in year 1996 observed in the series.

Time-series plot of Radiation(radtn)

```
plot(radtn, ylab='Radiation(radtn)', xlab='Time period', type='o', col='darkmagenta', main =
'Figure 47: Time-series plot of change in yearly values of radiation')
```

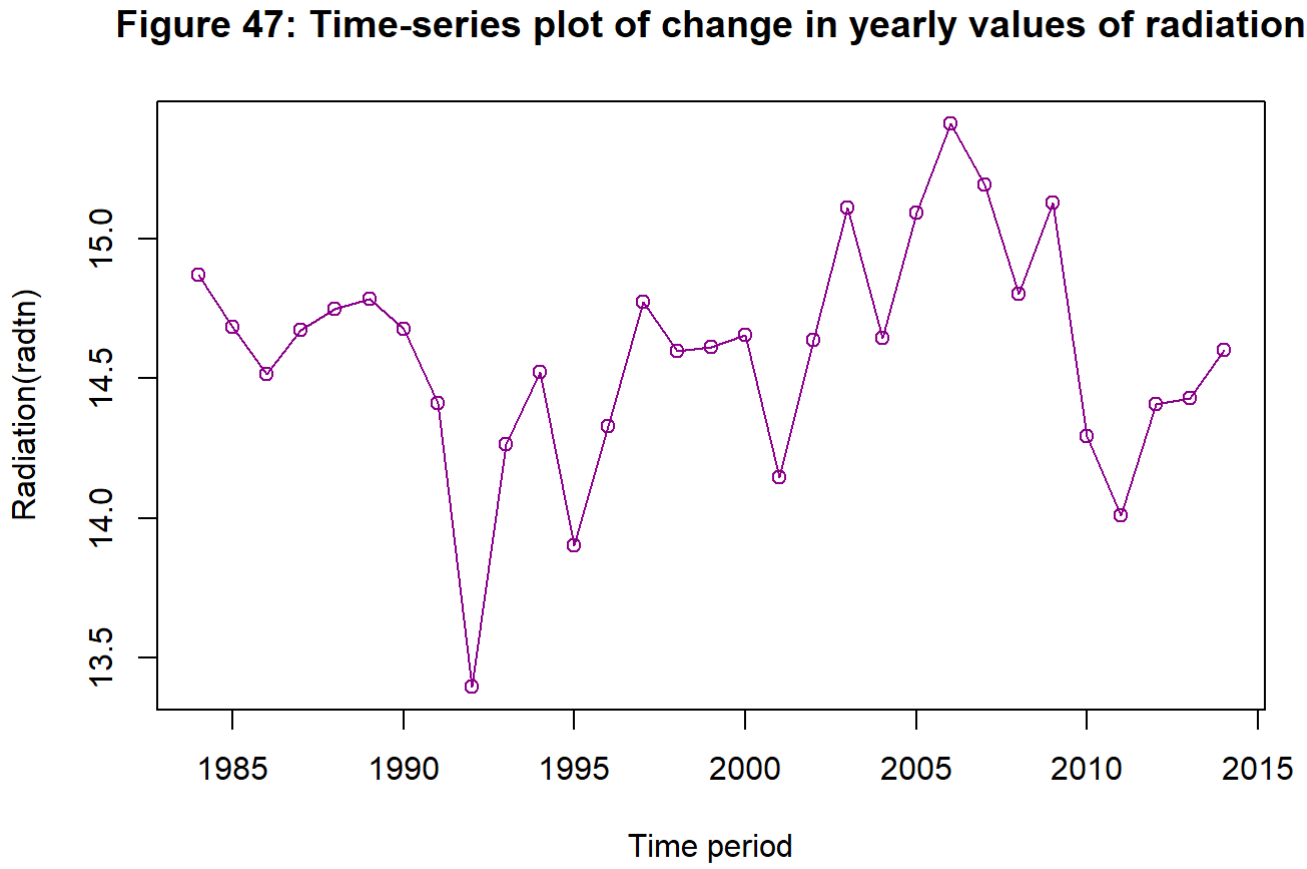


Figure 47

From the above plot, the Time Series plot shows us moderate level of **Seasonality** and **Nearly No Trend** with successive Auto regressive points and moving average overall from the year 1984 to 2014 respectively.

Checking **5 bullet points**:

- **Trend** - There is nearly No Trend present in the series.
- **Seasonality** - There is very high level of repeating patterns (seasonality) can be seen in the graph from year 1984-2014
- **Changing Variance** - There is a significant level of changing variance which is also in repeating patterns, can be seen if we look at the years 1992 and 2005.
- **Behavior** - A mixed Auto regressive and Moving average behavior can be seen in the graph.
- **Intervention point** - There is a sudden change point (drop) in year 1992 observed in the series.

Time-series plot of Humidity(humid)

```
plot(humid, ylab='Humidity(humid)', xlab='Time period', type='o', col='black', main = 'Figure 48: Time-series plot of change in yearly values of humidity')
```

Figure 48: Time-series plot of change in yearly values of humidity

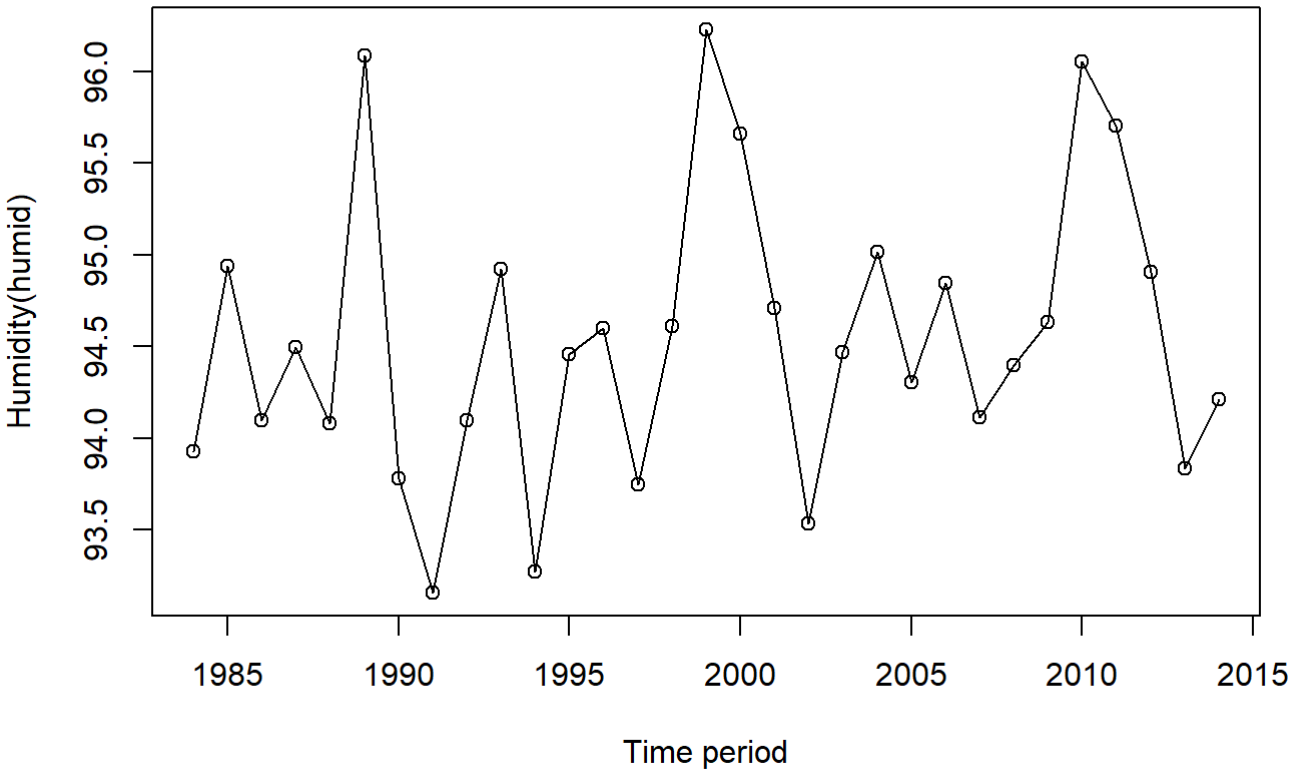


Figure 48

From the above plot, the Time Series plot shows us high level of **Seasonality** and **Nearly No Trend** with successive Auto regressive points and moving average overall from the year 1984 to 2014 respectively.

Checking **5 bullet points**:

- **Trend** - There is nearly No Trend present in the series.
- **Seasonality** - There is very high level of repeating patterns (seasonality) can be seen in the graph from year 1984-2014
- **Changing Variance** - There is a significant level of changing variance which is also in repeating patterns, can be seen if we look at the years 1992 and 2005.
- **Behavior** - A mixed Auto regressive and Moving average behavior can be seen in the graph.
- **Intervention point** - There is a sudden change point (drop) in year 1992 observed in the series.

Time-series plot of data set climate2 (scaled)

```
# Scaling the data climate2

scaled3  = scale(climate2)

# Plotting time-series plot containing all the five series together

plot(scaled3, plot.type = "s", col = c("blue", "red","green","darkmagenta","black"), main =
"Figure 49: Time Series plot of  scaled climate2",xlab="Time period")
legend("bottomright",lty=1,cex=0.55, text.width = 5, col=c("blue","red", "green", "darkmagent
a","black"), c("RBO", "Temperature", "Rainfall", "Radiation","Humidity"))
```

Figure 49: Time Series plot of scaled climate2

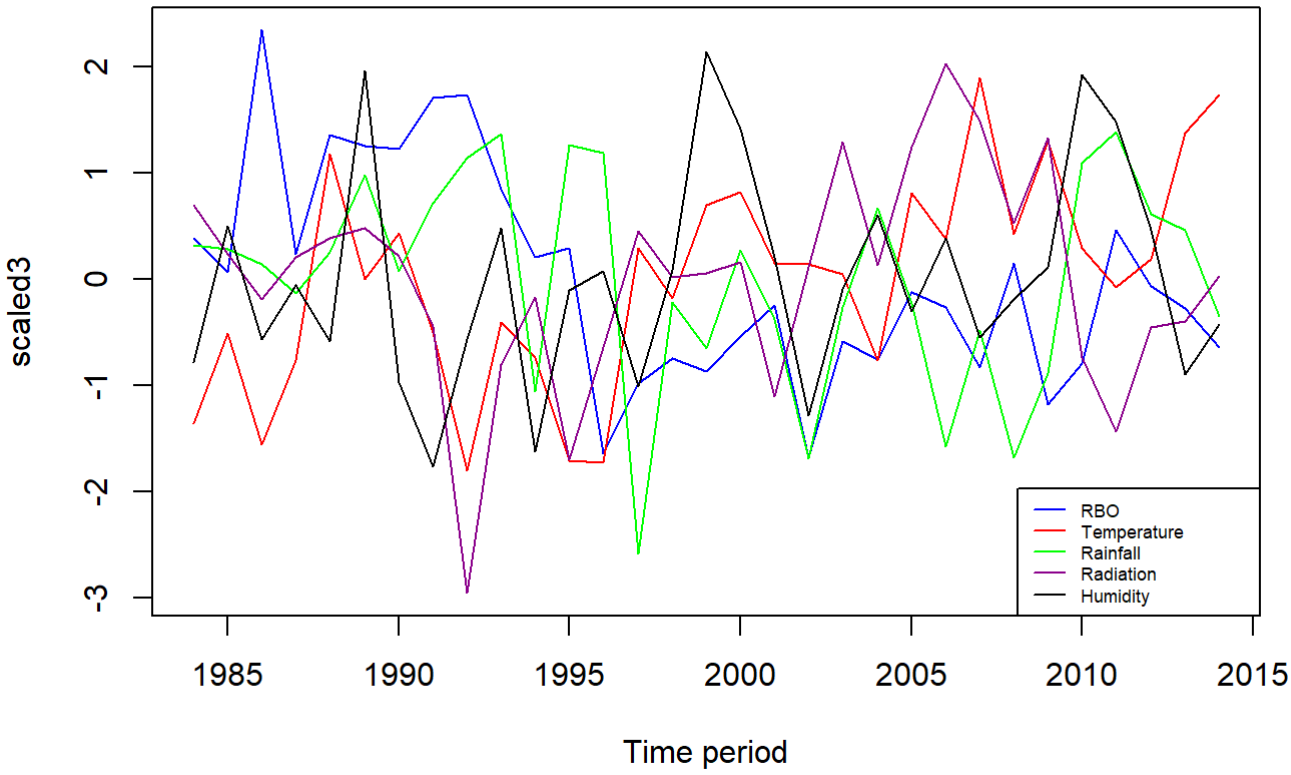


Figure 49

The above five series appear to be interconnected and do not exhibit **seasonality**, though some display a discernible trend. Following this, we will quantitatively assess these visual observations by computing the correlation among the five series.

Estimating correlation for climate2

```
# Calculating correlation
corr2<-rcorr(as.matrix(climate2))
flattenCorrMatrix(corr2$r, corr2$P)
```

```
##            row      column         cor           p
## 1          RBO Temperature -0.34520526 0.057174667
## 2          RBO    Rainfall  0.39322821 0.028636852
## 3  Temperature    Rainfall -0.39150719 0.029404244
## 4          RBO   Radiation -0.31736018 0.081917645
## 5  Temperature   Radiation  0.51935760 0.002753070
## 6     Rainfall   Radiation -0.58131610 0.000604604
## 7          RBO RelHumidity -0.17763486 0.339063146
## 8  Temperature RelHumidity  0.09350562 0.616848779
## 9     Rainfall RelHumidity  0.33846101 0.062543181
## 10   Radiation RelHumidity -0.05520965 0.768003663
```

Based on the provided results:

1. There is a strong positive correlation between Temperature and Radiation humidity (0.51), whereas radiation and rainfall humidity exhibit a notable negative correlation (-0.581).

2. Importantly, most of these correlations are statistically significant, with p-values below the 5% significance level, ensuring that the observed correlations are genuine and not due to random chance.
3. The dependent variable, RBO, does not have a pronounced correlation with any of the four climate indicators. However, there are slight significant correlations with each of them.
4. Specifically, RBO displays minor negative correlations with temperature , humidity and radiation, while it shows minor positive correlations with rainfall.

Check for non-stationarity

Plotting ACF and PACF for RBO

```
par(mfrow=c(1,2))
acf(RBO, main = "Figure 50.1: RBO  ACF ",lag.max = 48)
pacf(RBO, main = "Figure 50.2: RBO  PACF",lag.max = 48)
```

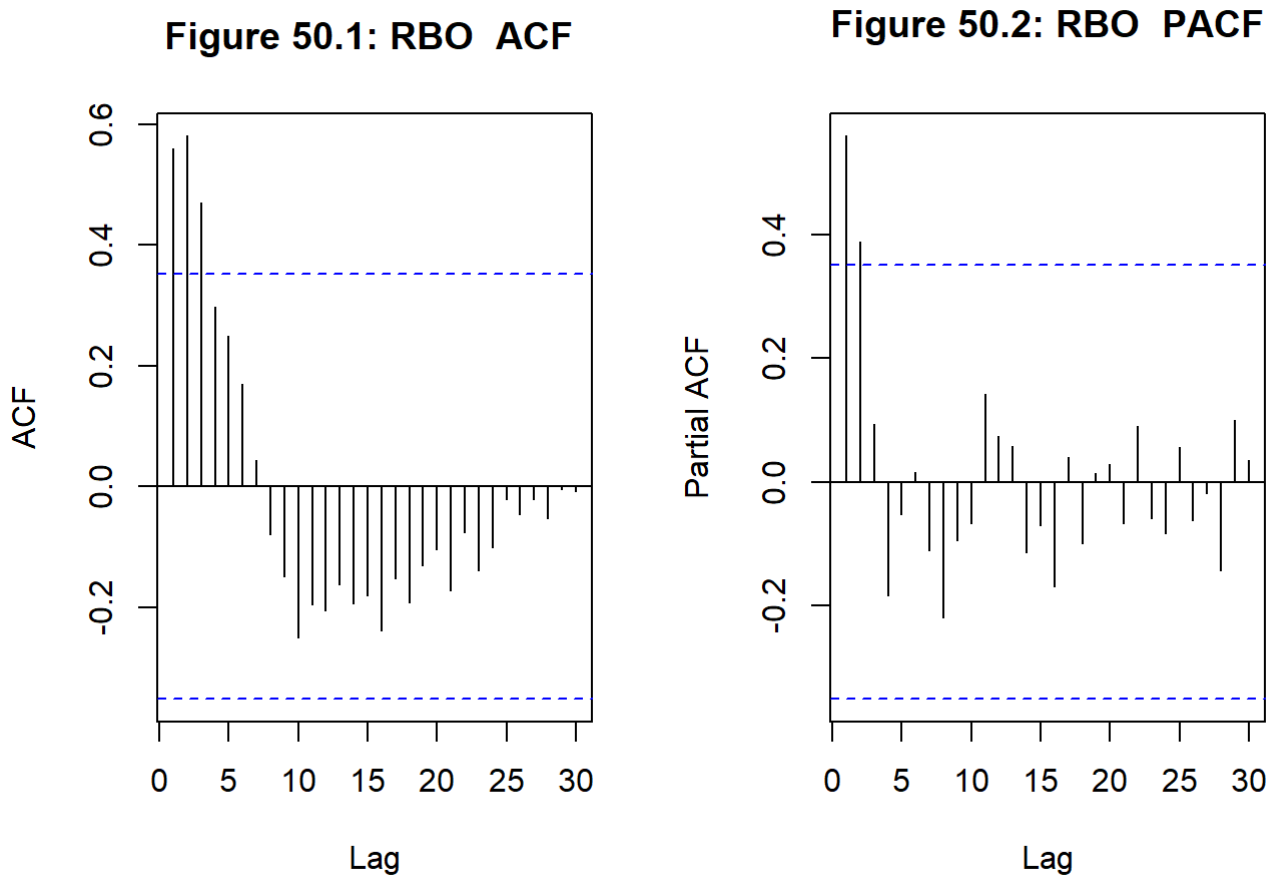


Figure 50

From the above ACF plot, we can observe a slowly decaying pattern and we also have a first large lag in PACF which indicates the RBO series is Non-stationary.

Plotting ACF and PACF for temperature(temptr)

```
par(mfrow=c(1,2))
acf(temptr, main = "Figure 51.1: Temperature ACF ",lag.max = 48)
pacf(temptr, main = "Figure 51.2: Temperature PACF",lag.max = 48)
```

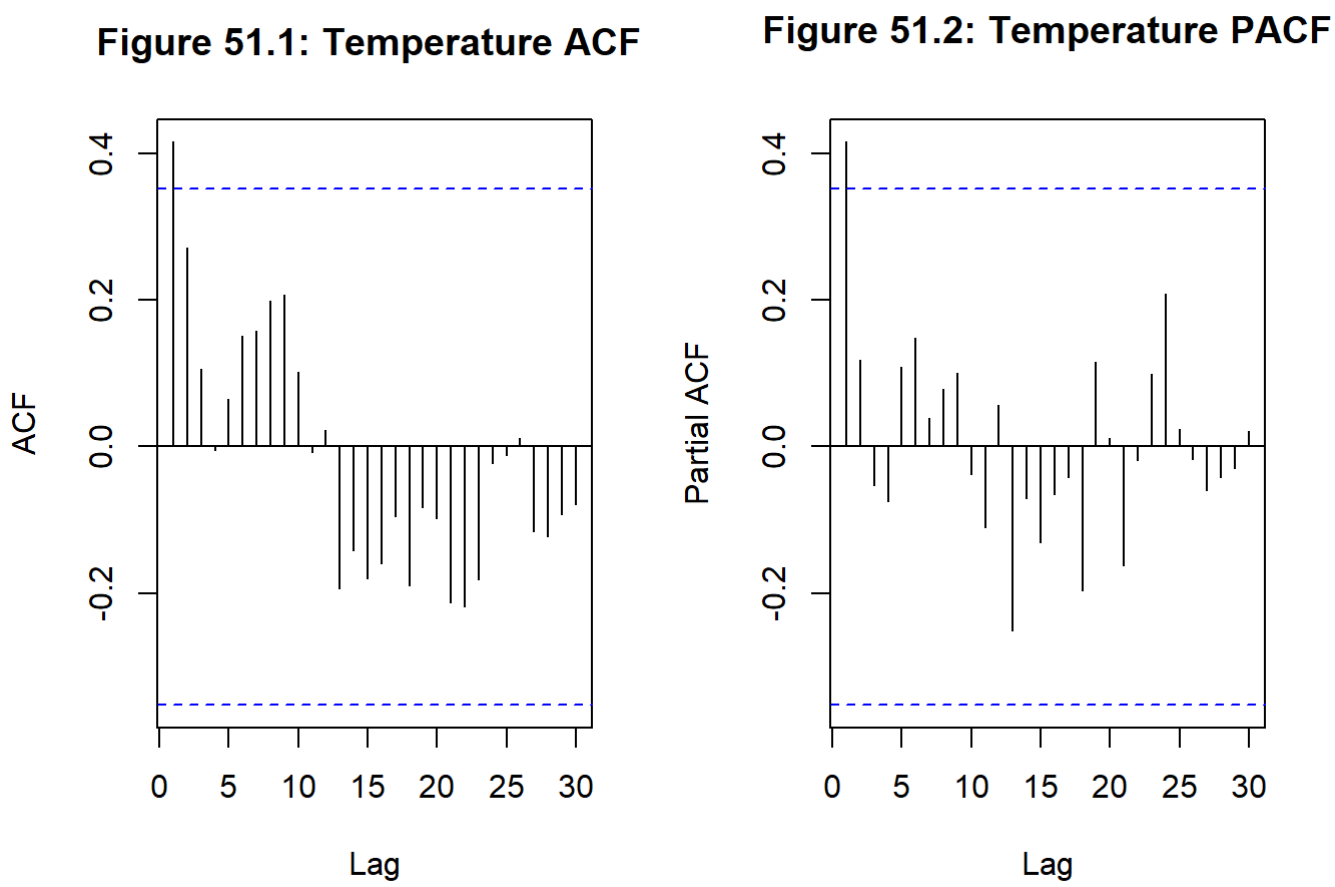
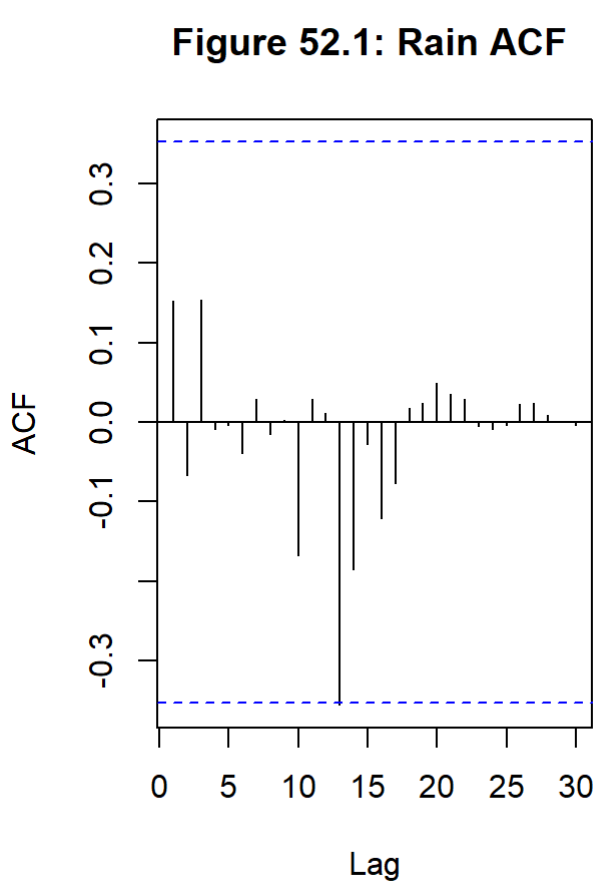


Figure 51

From the above ACF plot, we can observe a slowly decaying pattern and we also have a first large lag in PACF which indicates the Temperature series is Non-stationary.

Plotting ACF and PACF for rainfall(rain)

```
par(mfrow=c(1,2))
acf(rain, main = "Figure 52.1: Rain ACF ",lag.max = 48)
pacf(rain, main = "Figure 52.2: Rain PACF",lag.max = 48)
```



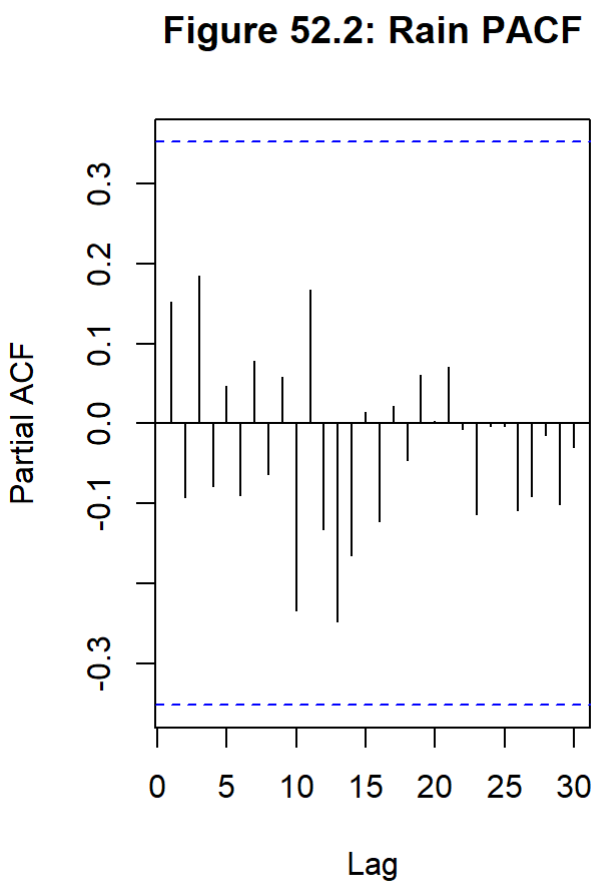


Figure 52

From the above ACF plot, we do not see a slowly decaying pattern and we also do not have a first large lag in PACF which indicates the rainfall series is stationary.

Plotting ACF and PACF for humidity(humid)

```
par(mfrow=c(1,2))
acf(humid, main = "Figure 53.1: Humidity ACF ",lag.max = 48)
pacf(humid, main = "Figure 53.2: Humidity PACF",lag.max = 48)
```

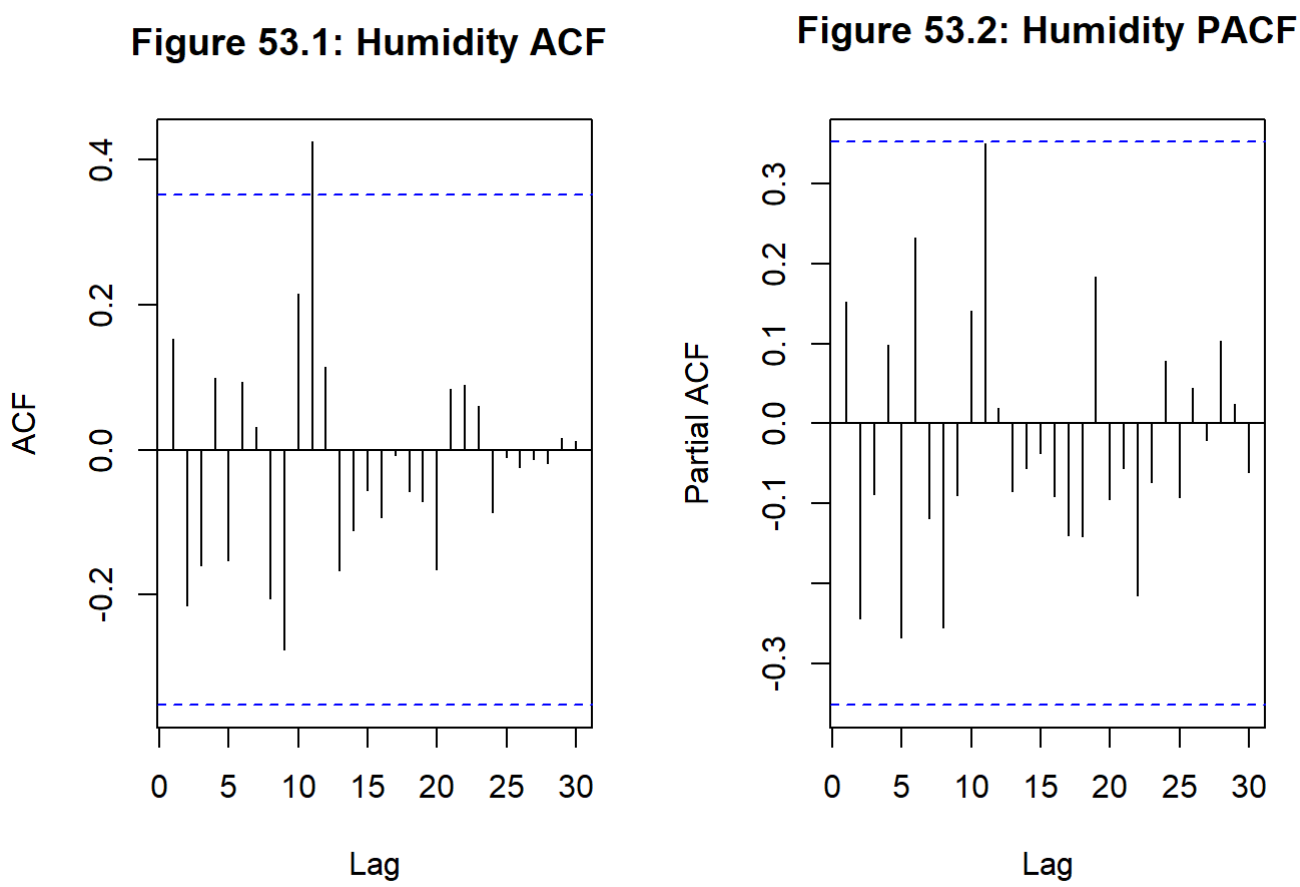


Figure 53

From the above ACF plot, we do not see a slowly decaying pattern and we also do not have a first large lag in PACF which indicates the humidity series is stationary.

Plotting ACF and PACF for radiation(radtn)

```
par(mfrow=c(1,2))
acf(radtn, main = "Figure 54.1: Radiation ACF ",lag.max = 48)
pacf(radtn, main = "Figure 54.2: Radiation PACF",lag.max = 48)
```

Figure 54.1: Radiation ACF

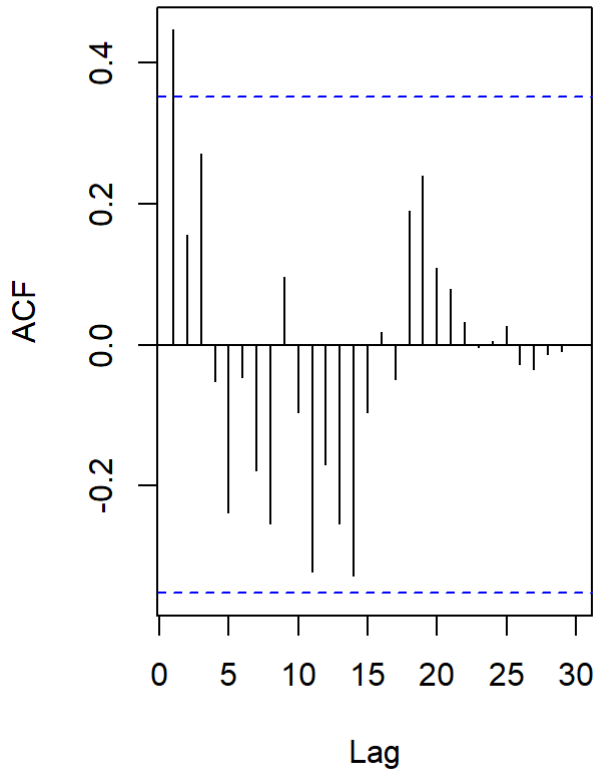


Figure 54.2: Radiation PACF

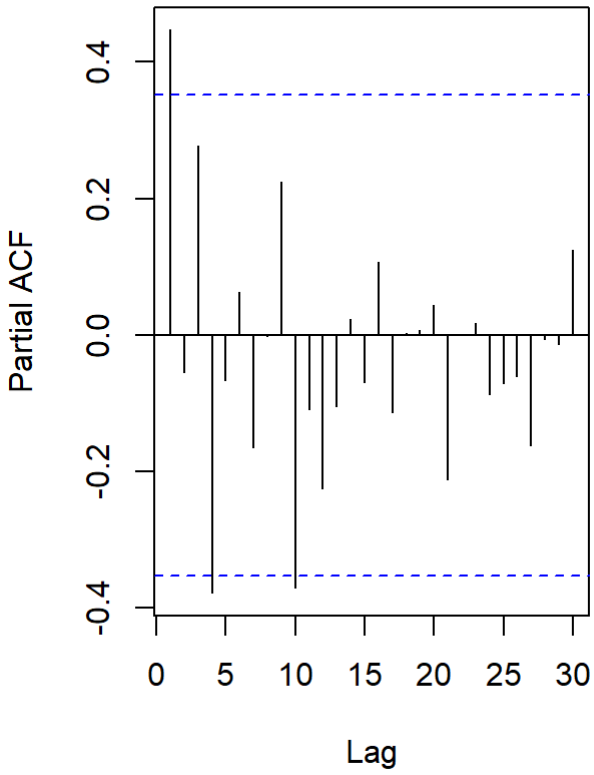


Figure 54

From the above ACF plot, we can observe a slowly decaying pattern and we also have a first large lag in PACF which indicates the Radiation series is Non-stationary.

Performing Stationarity Check

From the above correlation plots , we have seen the few variables series were stationary while others were not. Hence we will perform stationarity check on these and identify the same.

Performing ADF test on RBO series

```
adf.test(RBO, k=ar(RBO)$order)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  RBO
## Dickey-Fuller = -1.4542, Lag order = 2, p-value = 0.7829
## alternative hypothesis: stationary
```

From the above ADF test, we can observe that the p-value is greater than 5% level of significance, hence we say that the series is Non-stationary.

Performing ADF test on Temperature(temptr) series

```
adf.test(temptr, k=ar(temptr)$order)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  temptr
## Dickey-Fuller = -2.9938, Lag order = 1, p-value = 0.1902
## alternative hypothesis: stationary
```

From the above ADF test, we can observe that the p-value is greater than 5% level of significance, hence we say that the series is Non-stationary.

Performing ADF test on rainfall(rain) serie

```
adf.test(rain, k=ar(rain)$order)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  rain
## Dickey-Fuller = -4.5622, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

From the above ADF test, we can observe that the p-value is less than 5% level of significance, hence we say that the series is Stationary.

Performing ADF test on humidity(humid) series

```
adf.test(humid, k=ar(humid)$order)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  humid
## Dickey-Fuller = -4.5749, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

From the above ADF test, we can observe that the p-value is less than 5% level of significance, hence we say that the series is Stationary.

Performing ADF test on radiation series

```
adf.test(radtn, k=ar(radtn)$order)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  radtn
## Dickey-Fuller = -2.7317, Lag order = 4, p-value = 0.2911
## alternative hypothesis: stationary
```

From the above ADF test, we can observe that the p-value is greater than 5% level of significance, hence we say that the series is Non-stationary.

Transformation for series

We will now transform the above **Non-stationary** series by performing orders of **differencing** upon them.

Performing Differencing

Differencing of RBO series

```
RBO_diff = diff(RBO)
plot(RBO_diff,ylab='RBO',xlab='Time period', col="blue", main = "Figure 55: Time series plot
of first differenced RBO")
```

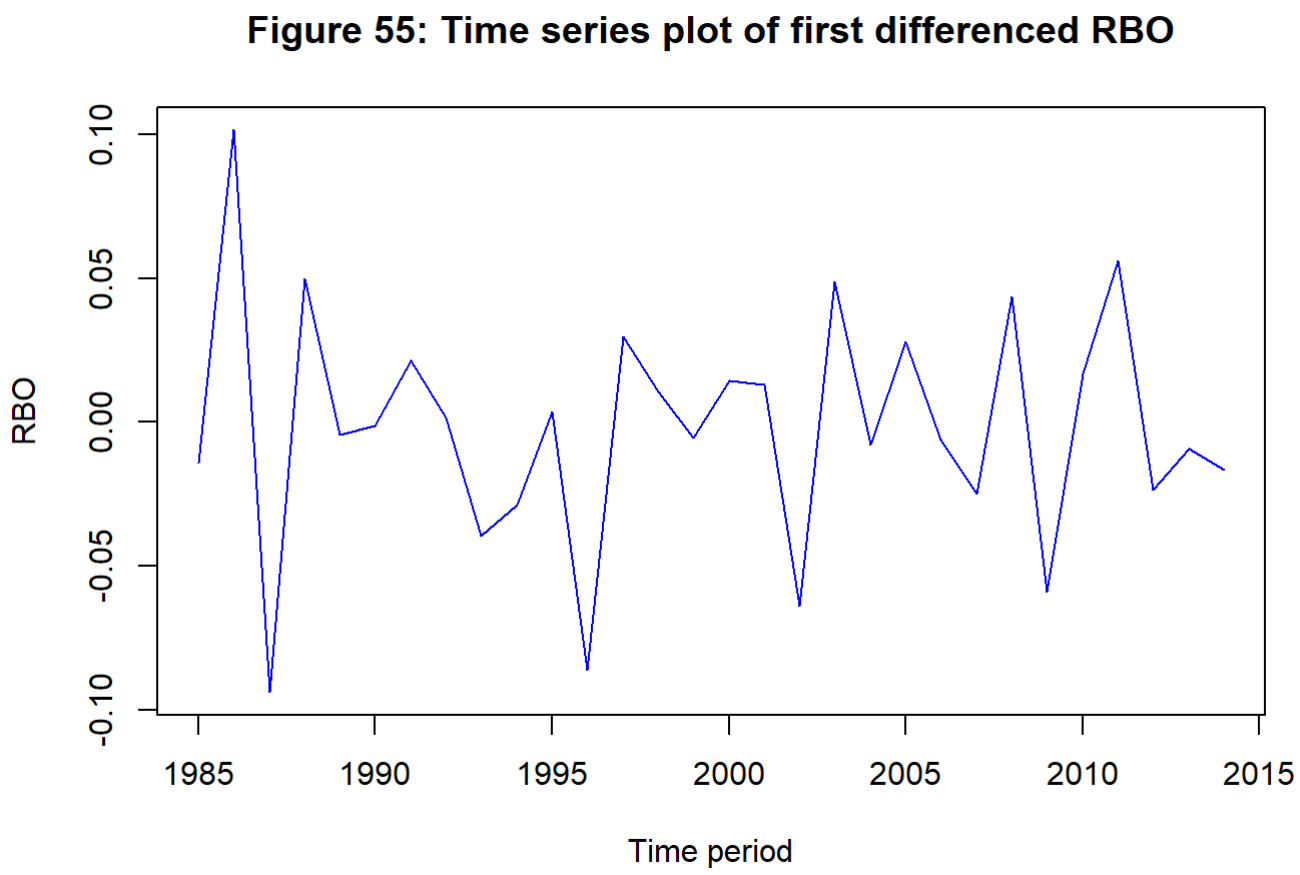


Figure 55

```
adf.test(RBO)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  RBO
## Dickey-Fuller = -2.0545, Lag order = 3, p-value = 0.5518
## alternative hypothesis: stationary
```

From the result of first differencing of series, we can still observe that the p-value is greater than 5% level of significance. Hence we will perform next order of differencing then.

Second Differencing of RBO series

```
RBO_diff2 = diff(RBO_diff)
plot(RBO_diff2,ylab='RBO',xlab='Time period', col="blue", main = "Figure 56: Time series plot
of second differenced RBO")
```

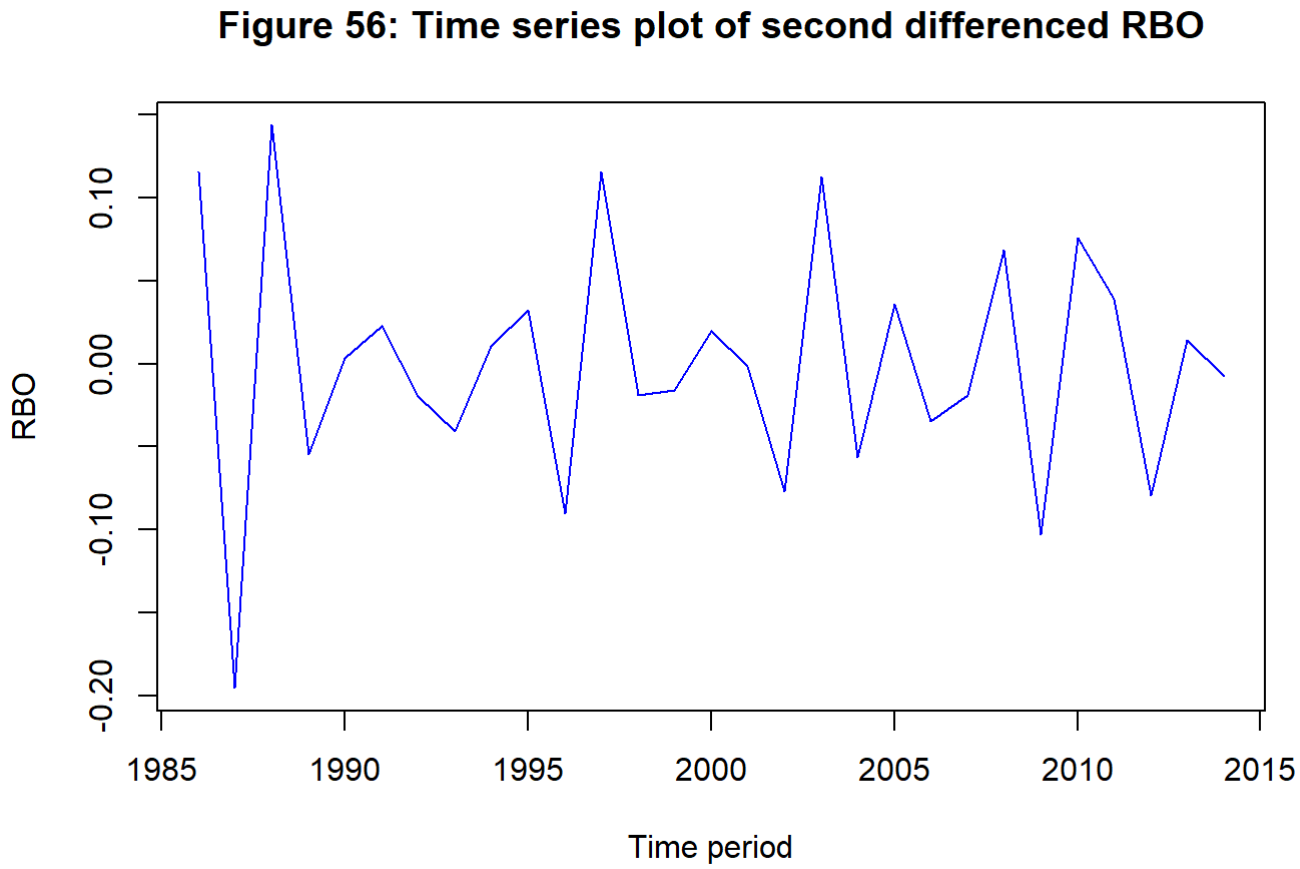


Figure 56

```
adf.test(RBO_diff2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  RBO_diff2
## Dickey-Fuller = -3.1875, Lag order = 3, p-value = 0.1174
## alternative hypothesis: stationary
```

From the result of second differencing of series, we can still observe that the p-value is greater than 5% level of significance. Hence we will perform next order of differencing then.

Third Differencing of RBO series

```
RBO_diff3 = diff(RBO_diff2)
plot(RBO_diff3,ylab='RBO',xlab='Time period', col="blue", main = "Figure 57: Time series plot
of third differenced RBO")
```

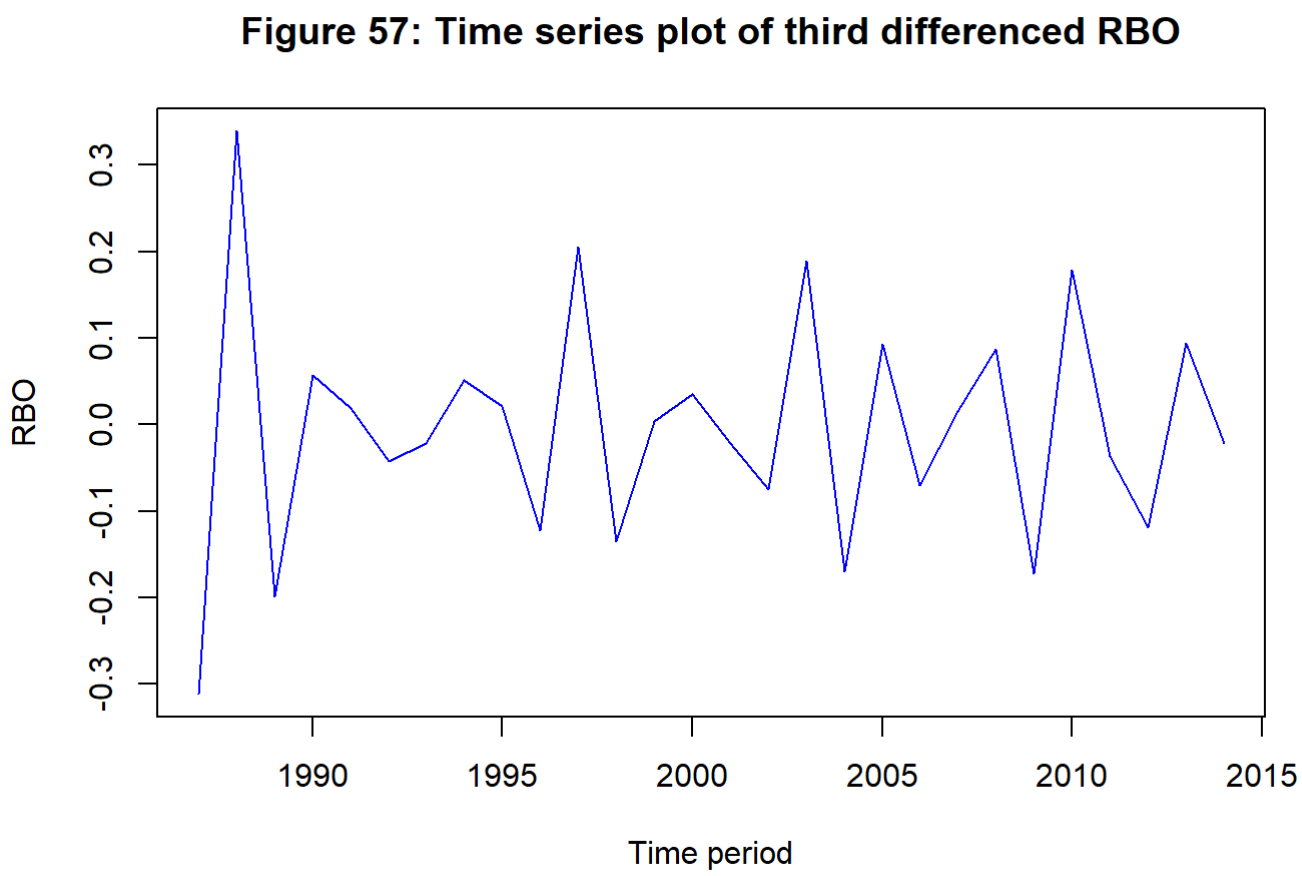


Figure 57

```
adf.test(RBO_diff3)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  RBO_diff3
## Dickey-Fuller = -4.0249, Lag order = 3, p-value = 0.02192
## alternative hypothesis: stationary
```

After performing third differencing and ADF test on it, it is now clear that the series is Stationary as the p-value is now less than 5% level of significance.

Differencing of temperature(temptr) series

```
temptrdiff = diff(temptr)
plot(temptrdiff,ylab='Temperature',xlab='Time period', col="red", main = "Figure 58: Time ser
ies plot of first differenced temptr series")
```

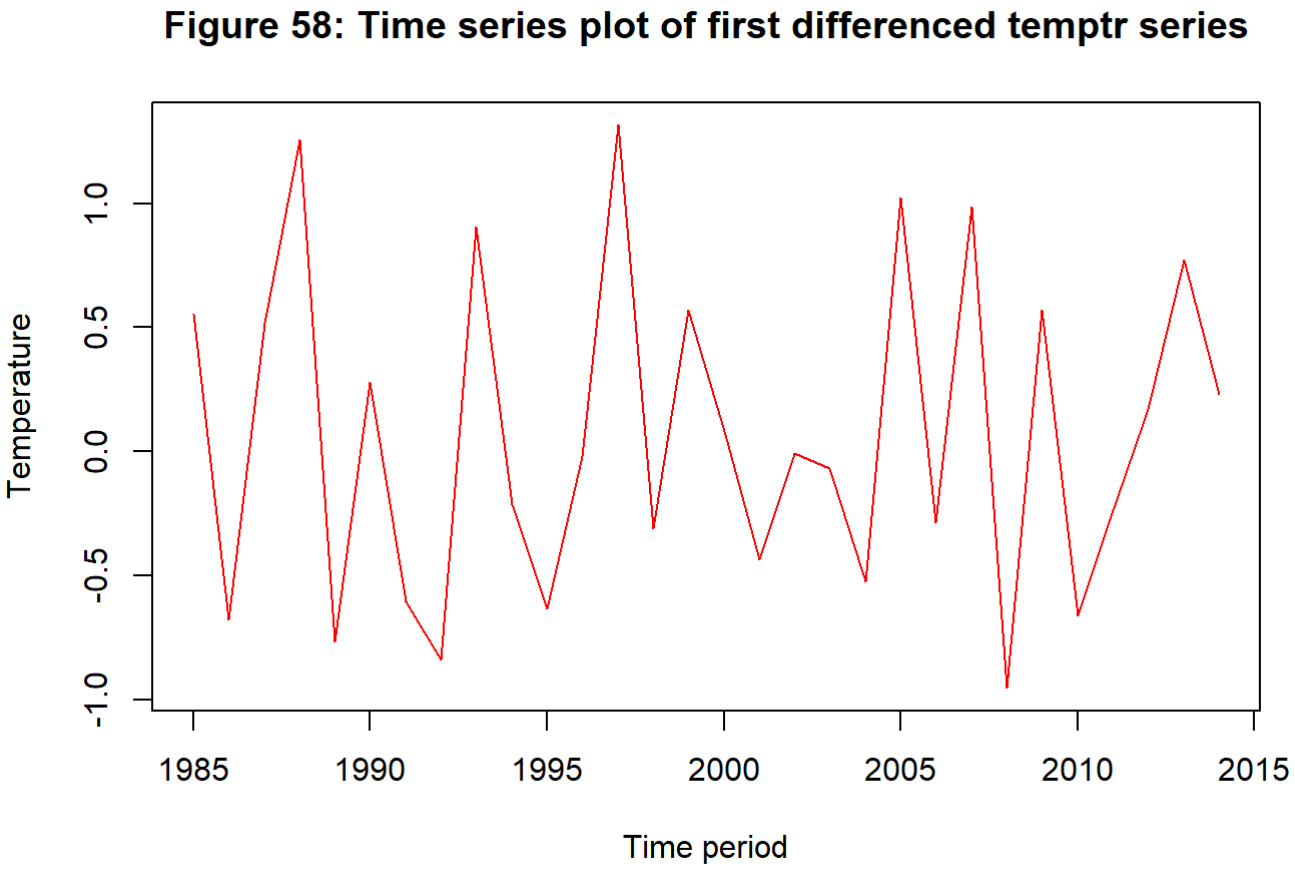


Figure 58

```
adf.test(temptrdiff)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  temptrdiff
## Dickey-Fuller = -3.4245, Lag order = 3, p-value = 0.07255
## alternative hypothesis: stationary
```

From the result of first differencing of series, we can still observe that the p-value is greater than 5% level of significance. Hence we will perform next order of differencing then.

Second Differencing of temperature(temptr) series

```
temptrdiff2 = diff(temptrdiff)
plot(temptrdiff2,ylab='Temperature',xlab='Time period', col="red", main = "Figure 59: Time se
ries plot of second differenced temptr series")
```

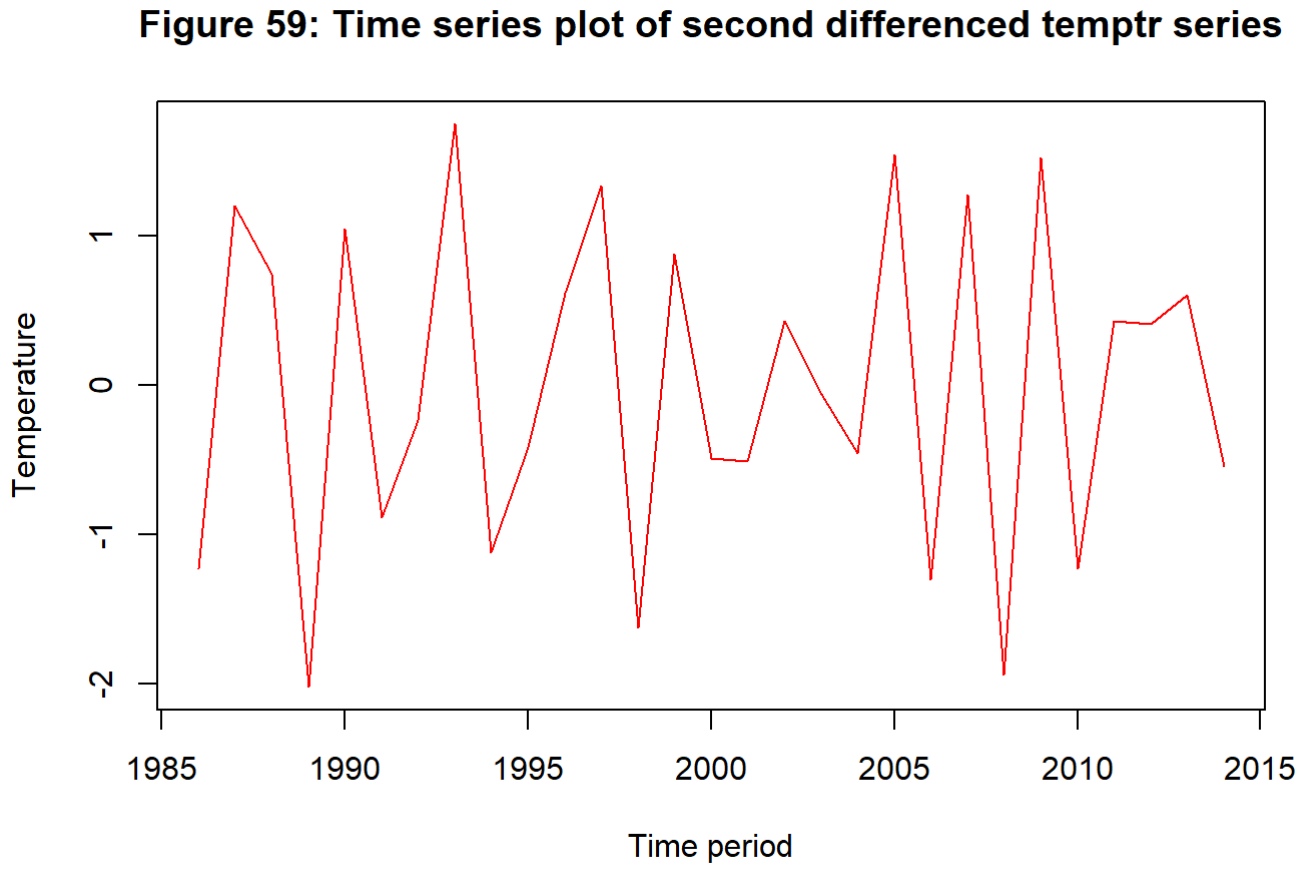


Figure 59

```
adf.test(temptrdiff2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  temptrdiff2
## Dickey-Fuller = -3.2642, Lag order = 3, p-value = 0.09558
## alternative hypothesis: stationary
```

From the result of second differencing of series, we can still observe that the p-value is greater than 5% level of significance. Hence we will perform next order of differencing then.

Third Differencing of temperature(temptr) series

```
temptrdiff3 = diff(temptrdiff2)
plot(temptrdiff3,ylab='Temperature',xlab='Time period', col="red", main = "Figure 60: Time se
ries plot of third differenced temptr series")
```

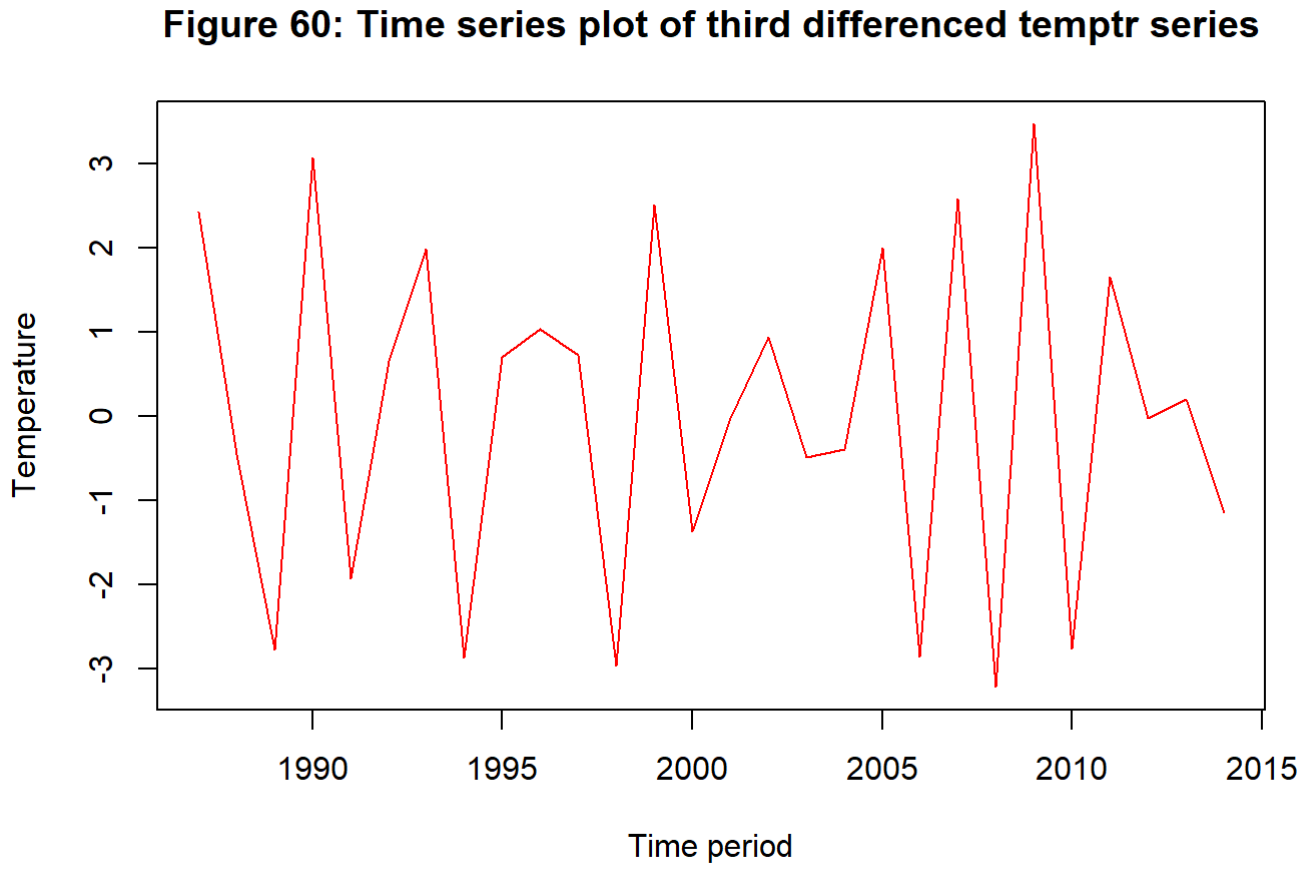


Figure 60

```
adf.test(temptrdiff3)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  temptrdiff3
## Dickey-Fuller = -4.0851, Lag order = 3, p-value = 0.01979
## alternative hypothesis: stationary
```

After performing third differencing and ADF test on it, it is now clear that the series is Stationary as the p-value is now less than 5% level of significance.

Differencing of radiation(radtn) series

```
radtndiff = diff(radtn)
plot(radtndiff,ylab='Radiation',xlab='Time period', col="darkmagenta", main = "Figure 61: Tim
e series plot of first differenced radtn series")
```

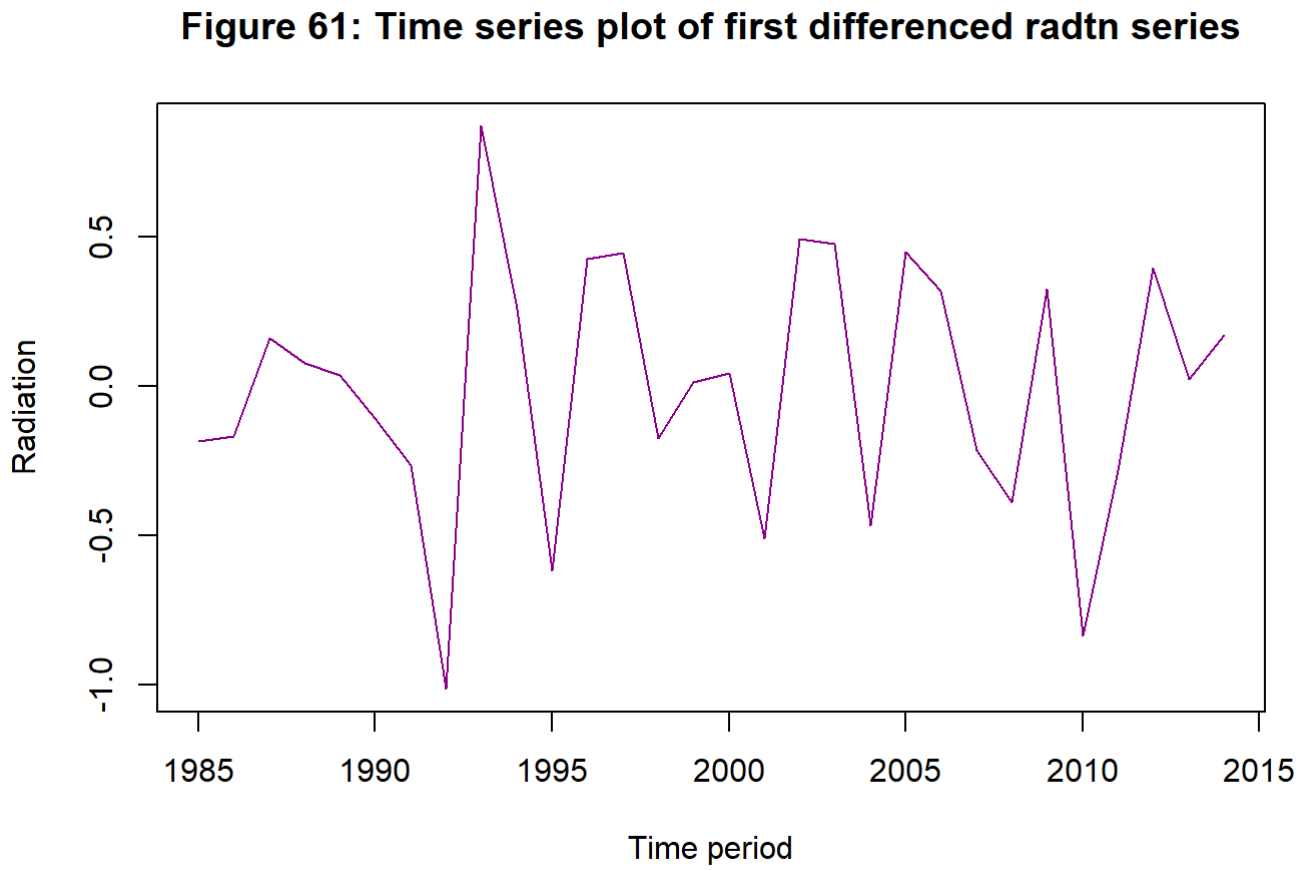


Figure 61

```
adf.test(radtndiff)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  radtndiff
## Dickey-Fuller = -2.6911, Lag order = 3, p-value = 0.3072
## alternative hypothesis: stationary
```

From the result of first differencing of series, we can still observe that the p-value is greater than 5% level of significance. Hence we will perform next order of differencing then.

Second Differencing of radiation(radtn) series

```
radtndiff2 = diff(radtndiff)
plot(radtndiff2,ylab='Radiation',xlab='Time period', col="darkmagenta", main = "Figure 62: Ti
me series plot of second differenced radtn series")
```

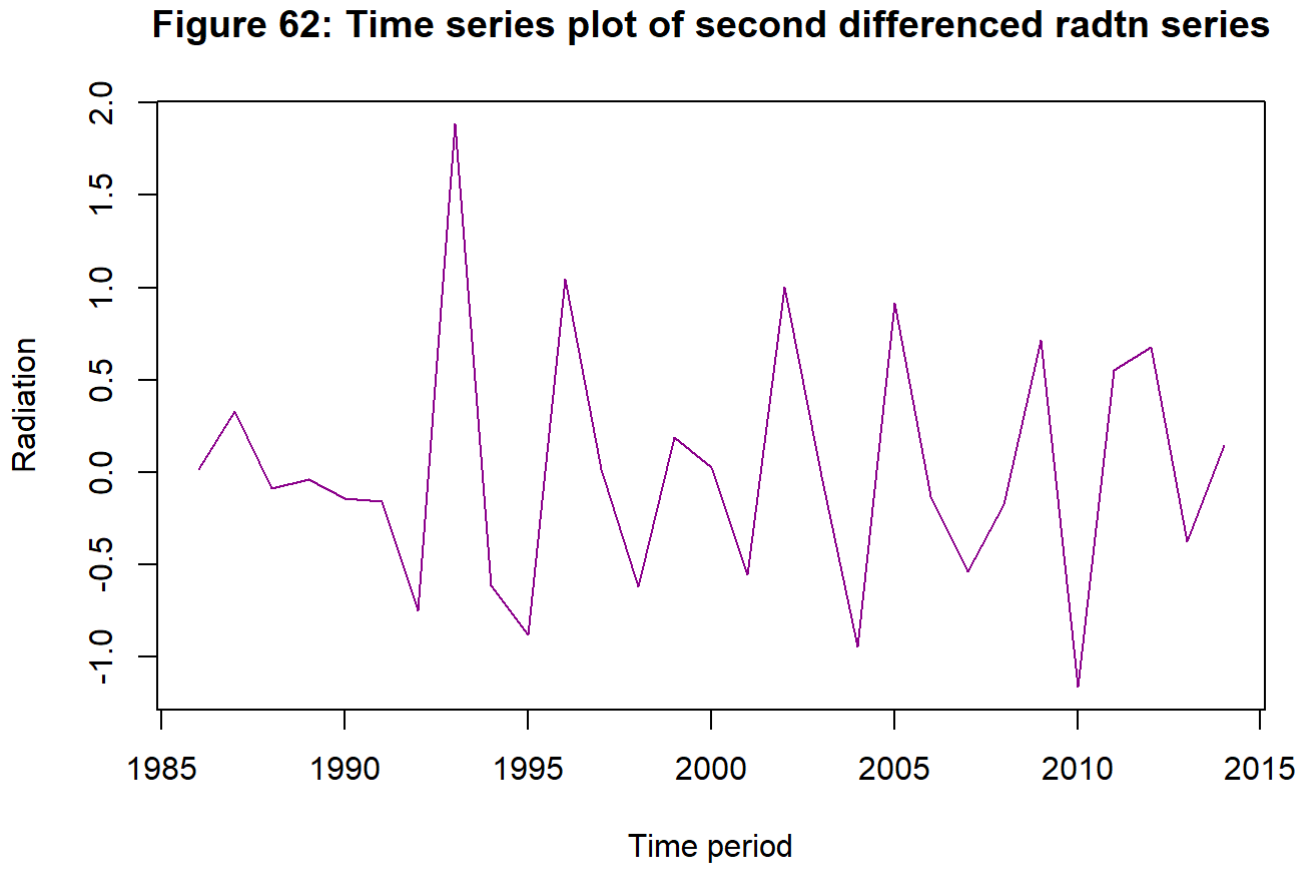


Figure 62

```
adf.test(radtndiff2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  radtndiff2
## Dickey-Fuller = -3.0559, Lag order = 3, p-value = 0.1678
## alternative hypothesis: stationary
```

From the result of second differencing of series, we can still observe that the p-value is greater than 5% level of significance. Hence we will perform next order of differencing then.

Third Differencing of radiation(radtn) series

```
radtndiff3 = diff(radtndiff2)
plot(radtndiff3,ylab='Radiation',xlab='Time period', col="darkmagenta", main = "Figure 63: Ti
me series plot of third differenced radtn series")
```

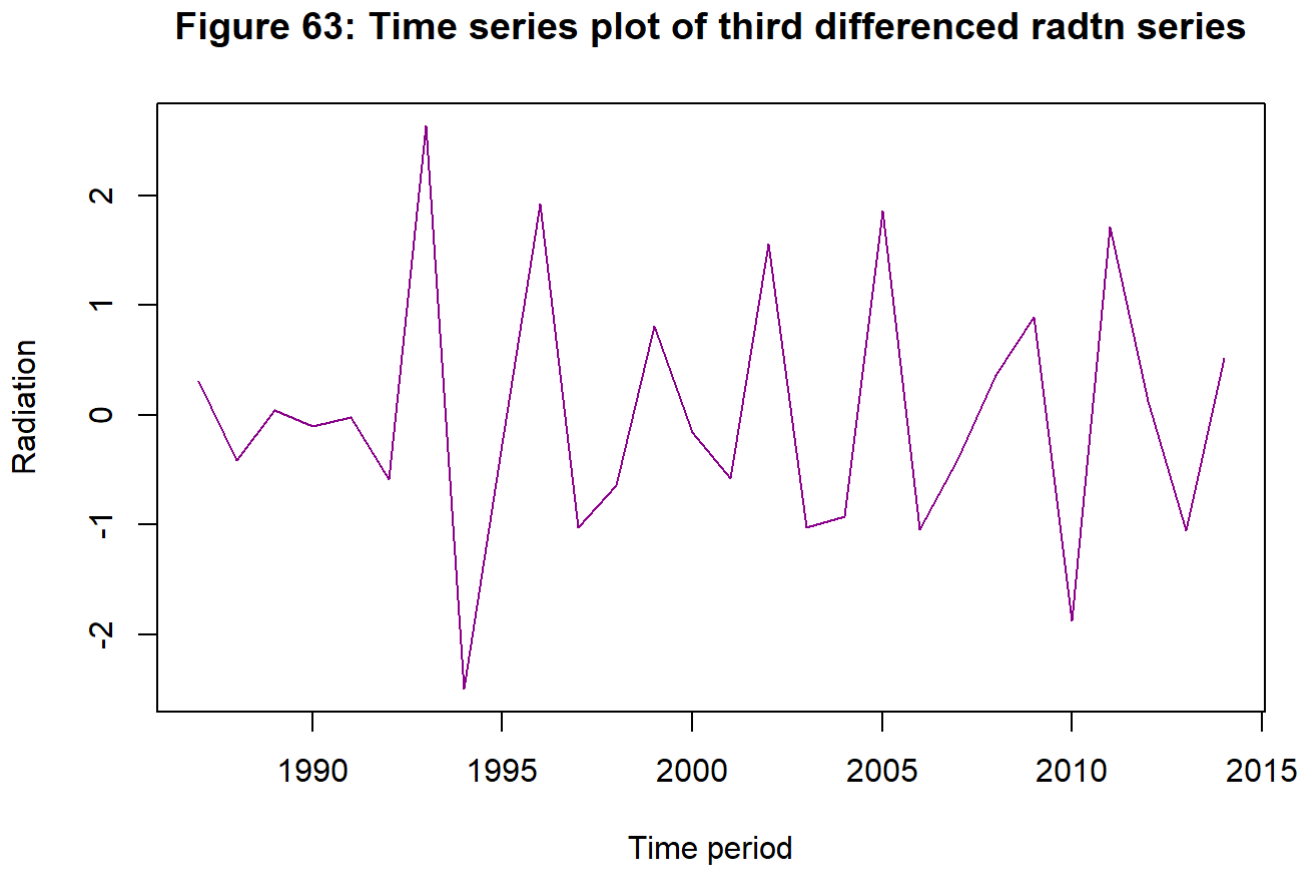


Figure 63

```
adf.test(radtndiff3)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  radtndiff3
## Dickey-Fuller = -4.1613, Lag order = 3, p-value = 0.01709
## alternative hypothesis: stationary
```

After performing third differencing and ADF test on it, it is now clear that the series is Stationary as the p-value is now less than 5% level of significance.

We have now successfully converted all the series into Stationary, we will now fit the models accordingly.

Time Series Regression Models

We will try fitting distributed lag models, which incorporate an independent explanatory series and its lags to assist explain the general variance and correlation structure in our dependent series, in an effort to identify a good model for predicting Climate series.

To determine the model's finite lag length, we create a procedure that computes measurements of accuracy like AIC/BIC and MASE for models with various lag lengths, then select the model with the lowest values.

Distributed Lag Model

The Distributed Lag Model from the Regression models describes that the effect of an independent variable on the dependent variables occurs over the time. Therefore we require to build distributed lag models to reduce the multi-collinearity and dependency for each variable. Some of the most important used methods are:

- **Finite Distributed Lag Model**
- **Polynomial Distributed Lag Model**
- **Koyck Distributed Lag Model**
- **Autoregressive Distributed Lag Model**

Finite Distributed Lag Model

To identify the optimal model using finite DLM, we'll examine each of the five series individually as predictors and implement the finite DLM based on the suitable lag duration.

We will determine the appropriate lag length for the model, where temperature serves as the predictor and RBO as the outcome variable. This is done using the **finiteDLMauto** function, which compares based on AIC, BIC, and MASE scores for lags between 1 to 10. The model with the smallest AIC, BIC, and MASE values will be selected.

```
# Changing names of column in data climate2 for ease of writing code

colnames(climate2) <- c("RBO","temptr","rain","radtn","humid")

# Applying finiteDLMauto to calculate best model based on AIC, BIC and MASE values

finiteDLMauto( x=as.vector(temptr), y= as.vector(RBO), q.min = 1, q.max = 10, model.type="dl
m", error.type = "AIC",trace=TRUE)
```

```
##    q - k    MASE       AIC       BIC   GMRAE     MBRAE R.Adj.Sq    Ljung-Box
## 1      1 1.00562 -97.52442 -91.91963 0.97718  -1.50100  0.05726 0.0022526947
## 2      2 1.14742 -91.57332 -84.73684 1.18433   2.00053  0.03520 0.0019973822
## 3      3 1.18155 -88.84678 -80.85356 1.19082   1.78688 -0.09575 0.0003753525
## 10    10 0.52353 -85.18937 -71.61058 0.65204 -10.06486 -0.01070 0.1806431756
## 4      4 1.24918 -82.35556 -73.28471 1.17170   0.61711 -0.15735 0.0006014497
## 5      5 1.09444 -81.62553 -71.56076 1.03411   0.44496 -0.08477 0.0016802983
## 9      9 0.62982 -79.75895 -66.66644 0.52185   2.67533 -0.28008 0.7633752533
## 8      8 0.71270 -78.35040 -65.85997 0.55285  -3.45059 -0.09835 0.2689392429
## 6      6 0.98697 -77.12304 -66.15316 0.75879   1.05760 -0.17893 0.0077737837
## 7      7 0.91375 -76.88467 -65.10413 0.79294   0.21697 -0.09764 0.0661515342
```

Modelling with several possible combinations

We will now model several possible combinations of model and decide to go for the best one.

```
model.rain = dlm(x=as.vector(rain), y=as.vector(RBO), q=1)
model.temptr = dlm(x=as.vector(temptr), y=as.vector(RBO), q=1)
model.humid = dlm(x=as.vector(humid), y=as.vector(RBO), q=1)
model.radtn = dlm(x=as.vector(radtn), y=as.vector(RBO), q=1)
model.nointercepttem = dlm(formula = RBO ~ temptr - 1, data=data.frame(climate2), q=1)
model.nointerceptrain = dlm(formula = RBO ~ rain - 1, data=data.frame(climate2), q=1)
model.nointercepthum = dlm(formula = RBO ~ humid - 1, data=data.frame(climate2), q=1)
model.nointerceptrad = dlm(formula = RBO ~ radtn - 1, data=data.frame(climate2), q=1)
```

In order to go for the bes model we will evaluate them based on different scores such as AIC, BIC and MASE.

Evaluating based on AIC, BIC and MASE

```
#  AIC scores
sort.score(AIC(model.temptr$model,model.rain$model,model.humid$model,model.radtn$model,model.
nointercepttem$model,model.nointerceptrain$model,model.nointercepthum$model,model.nointercept
rad$model), score = "aic")
```

```
##                              df        AIC
## model.rain$model              4 -100.89798
## model.radtn$model             4  -97.71113
## model.temptr$model            4  -97.52442
## model.humid$model             4  -94.56619
## model.nointercepthum$model    3  -94.44965
## model.nointerceptrad$model    3  -89.33439
## model.nointercepttem$model    3  -86.53783
## model.nointerceptrain$model   3  -62.07890
```

```
#  BIC scores

sort.score(BIC(model.temptr$model,model.rain$model,model.humid$model,model.radtn$model,model.
nointercepttem$model,model.nointerceptrain$model,model.nointercepthum$model,model.nointercept
rad$model), score = "bic")
```

```
##                              df       BIC
## model.rain$model              4 -95.29319
## model.radtn$model             4 -92.10634
## model.temptr$model            4 -91.91963
## model.nointercepthum$model    3 -90.24605
## model.humid$model             4 -88.96140
## model.nointerceptrad$model    3 -85.13079
## model.nointercepttem$model    3 -82.33424
## model.nointerceptrain$model   3 -57.87530
```

```
# MASE scores

sort.score(MASE(model.temptr$model,model.rain$model,model.humid$model,model.radtn$model,mode
l.nointercepttem$model,model.nointerceptrain$model,model.nointercepthum$model,model.nointerce
ptrad$model), score = "mase")
```

```
##                               n      MASE
## model.rain$model             30 0.9417954
## model.temptr$model           30 1.0056219
## model.radtn$model            30 1.0630293
## model.humid$model            30 1.1010995
## model.nointercepthum$model   30 1.1765863
## model.nointerceptrad$model   30 1.3296554
## model.nointercepttem$model   30 1.4121567
## model.nointerceptrain$model 30 1.9110310
```

From the above results of MASE scores, we can observe that the best model turns out to be **model.rain** with respect to the MASE. We will next analyse the model and do residual analysis.

Analyzing and Summarising Model **model.rain**

```
summary(model.rain)
```

```
##
## Call:
## lm(formula = model.formula, data = design)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.105903 -0.024178 -0.006166  0.014773  0.099699
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.56594    0.06397   8.847 1.84e-09 ***
## x.t          0.04199    0.02058   2.040   0.0512 .
## x.1          0.03032    0.02059   1.473   0.1524
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04153 on 27 degrees of freedom
## Multiple R-squared:  0.2156, Adjusted R-squared:  0.1575
## F-statistic: 3.711 on 2 and 27 DF,  p-value: 0.03767
##
## AIC and BIC values for the model:
##        AIC       BIC
## 1 -100.898 -95.29319
```

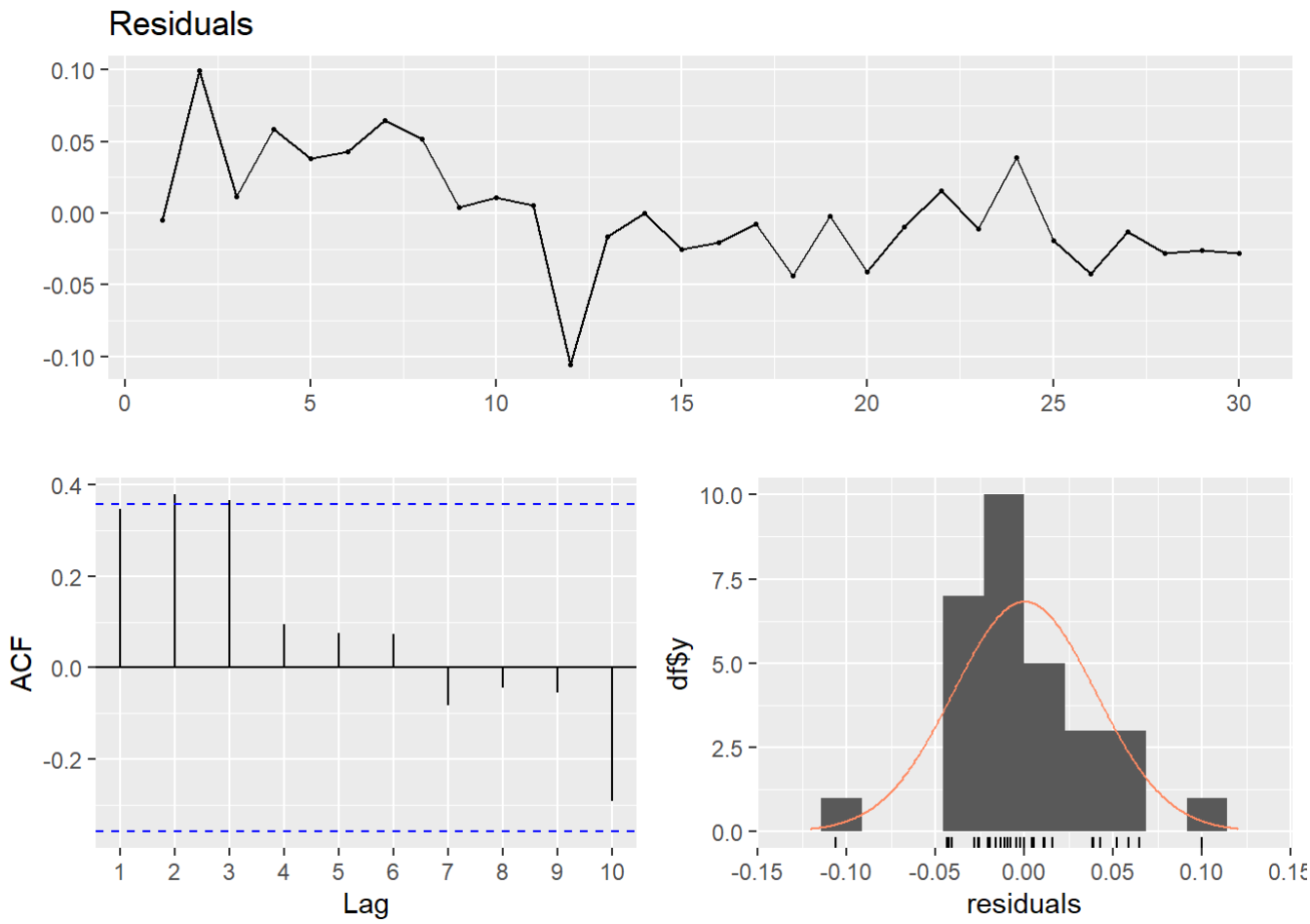
From the outputs of **model.rain** * model and its summary :

- The values of AIC and BIC of the model are -100.898 -95.293.
- The adjusted R-squared value reported was $0.1575 $ which is very poor.
- Two out of three of the coefficients are significant.
- The F-statistic test is reported at 3.71 on 2 and 27 Degrees of Freedom.

Residual Analysis of model

The **model.rain** will undergo a residual analysis using the checkresiduals function. Besides creating plots of the residuals, the Breusch-Godfrey test will also be executed. In this test, H0 suggests that the residuals don’t exhibit serial correlation, whereas Ha suggests the presence of serial correlation in the residuals.

```
checkresiduals(model.rain$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 6
##
## data:  Residuals
## LM test = 9.6735, df = 6, p-value = 0.1391
```

From the outputs of Model **model.rain** residuals :

- The p-value from Breusch-Godfrey test was greater than significance level of 0.05.
- The ACF plot reveals that there is seasonality and correlation exist in residual due to significant lags present in the ACF.
- The time-series plot of residual seems to follow a downward trend overall.
- The histogram does seems to follow normal distribution with outlier on both ends.

So, we can conclude that the model **model.rain** is a decent model with an appropriate composition of residuals.

Polynomial Distributed Lag Model

The finite DLM model did not adequately fit the data, so we're considering a polynomial model instead. Before proceeding with the polynomial model, we'll determine the optimal lag length using `finiteDLMauto` , basing our decision on the AIC, BIC, and MASE values of the fitted models.

Identifying lag length based on AIC, BIC & MASE

```
finiteDLMauto(x = as.vector(temptr), y = as.vector(RBO), q.min = 1, q.max = 10, k.order = 2,
              model.type = "poly", error.type ="AIC", trace = TRUE)
```

```
##     q - k    MASE       AIC       BIC   GMRAE    MBRAE R.Adj.Sq    Ljung-Box
## 1   1 - 2 1.00562 -97.52442 -91.91963 0.97718 -1.50100  0.05726 0.0022526947
## 2   2 - 2 1.14742 -91.57332 -84.73684 1.18433  2.00053  0.03520 0.0019973822
## 3   3 - 2 1.17995 -90.36706 -83.70604 1.28905  6.54348 -0.06824 0.0004711456
## 9   9 - 2 0.66946 -89.78677 -84.33156 0.51496  1.86757  0.06293 0.9646666769
## 10 10 - 2 0.67149 -87.29374 -82.07113 0.61193 -0.89172 -0.03702 0.4275315970
## 8   8 - 2 0.77095 -86.93895 -81.26148 0.65368 -0.23247  0.12834 0.4128617986
## 5   5 - 2 1.16375 -85.59406 -79.30358 1.28421  2.08989 -0.01298 0.0021542325
## 4   4 - 2 1.24638 -85.57725 -79.09807 1.06398 -0.97930 -0.08762 0.0006720648
## 6   6 - 2 1.02179 -84.11435 -78.01997 0.86427  0.00783  0.00633 0.0132272227
## 7   7 - 2 0.91185 -83.72257 -77.83230 0.82528 -0.02209  0.06084 0.0895446547
```

According to the given results, the optimal lag-length is 1. However, a polynomial model cannot be constructed with a lag-length of 1. Consequently, we'll opt for the next best lag-length, which is 2, as guided by the AIC, BIC, and MASE metrics. By utilizing the finiteDLMauto function, we assessed alternate predictors to ascertain the appropriate lag length. For every predictor variable within the dataset, the ideal lag length was determined to be 1. Therefore, for all the polynomial DLMs, we'll adopt a lag length of 2 and an order of 2.

Fitting all possible combination of polynomial models

```
# Model Fitting

model.temptr_poly = polyDlm(x = as.vector(temptr), y = as.vector(RBO), q = 2, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##        Estimate Std. Error t value P(>|t|)
## beta.0  -0.0254     0.0150  -1.690   0.102
## beta.1   0.0104     0.0163   0.636   0.530
## beta.2  -0.0107     0.0155  -0.691   0.495
```

```
model.rain_poly = polyDlm(x = as.vector(rain), y = as.vector(RBO), q = 2, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##        Estimate Std. Error t value P(>|t|)
## beta.0   0.0449     0.0206    2.18  0.0383
## beta.1   0.0252     0.0209    1.21  0.2380
## beta.2   0.0305     0.0207    1.47  0.1530
```

```
model.humid_poly = polyDlm(x = as.vector(humid), y = as.vector(RBO), q = 2, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##        Estimate Std. Error t value P(>|t|)
## beta.0 -0.01060     0.0118  -0.900   0.376
## beta.1 -0.00180     0.0116  -0.155   0.878
## beta.2 -0.00492     0.0118  -0.417   0.680
```

```
model.radtn_poly = polyDlm(x = as.vector(radtn), y = as.vector(RBO), q = 2, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##        Estimate Std. Error t value P(>|t|)
## beta.0  -0.0436     0.0231  -1.890  0.0701
## beta.1   0.0111     0.0256   0.433  0.6690
## beta.2   0.0117     0.0230   0.507  0.6160
```

We will now evaluate the above models based on AIC, BIC and MASE scores.

Evaluating based on AIC, BIC and MASE

```
# AIC scores
sort.score(AIC(model.temptr_poly$model,model.rain_poly$model,model.humid_poly$model,model.rad
tn_poly$model), score = "aic")
```

```
##                          df       AIC
## model.rain_poly$model     5 -96.70956
## model.temptr_poly$model   5 -91.57332
## model.radtn_poly$model    5 -91.50708
## model.humid_poly$model    5 -88.37920
```

```
# BIC scores
sort.score(BIC(model.temptr_poly$model,model.rain_poly$model,model.humid_poly$model,model.rad
tn_poly$model), score = "bic")
```

```
##                          df       BIC
## model.rain_poly$model     5 -89.87308
## model.temptr_poly$model   5 -84.73684
## model.radtn_poly$model    5 -84.67061
## model.humid_poly$model    5 -81.54272
```

```
# MASE scores
sort.score(MASE(model.temptr_poly$model,model.rain_poly$model,model.humid_poly$model,model.ra
dtn_poly$model), score = "mase")
```

```
##                           n      MASE
## model.rain_poly$model    29 0.9993747
## model.temptr_poly$model  29 1.1474219
## model.radtn_poly$model   29 1.1747093
## model.humid_poly$model   29 1.2358844
```

Using AIC, BIC, and MASE criteria, the model.rainpoly stands out as the optimal polynomial DLM. We should delve deeper into this model's analysis.

Analyzing and summarising **model.rain_poly**

```
summary(model.rain_poly$model)
```

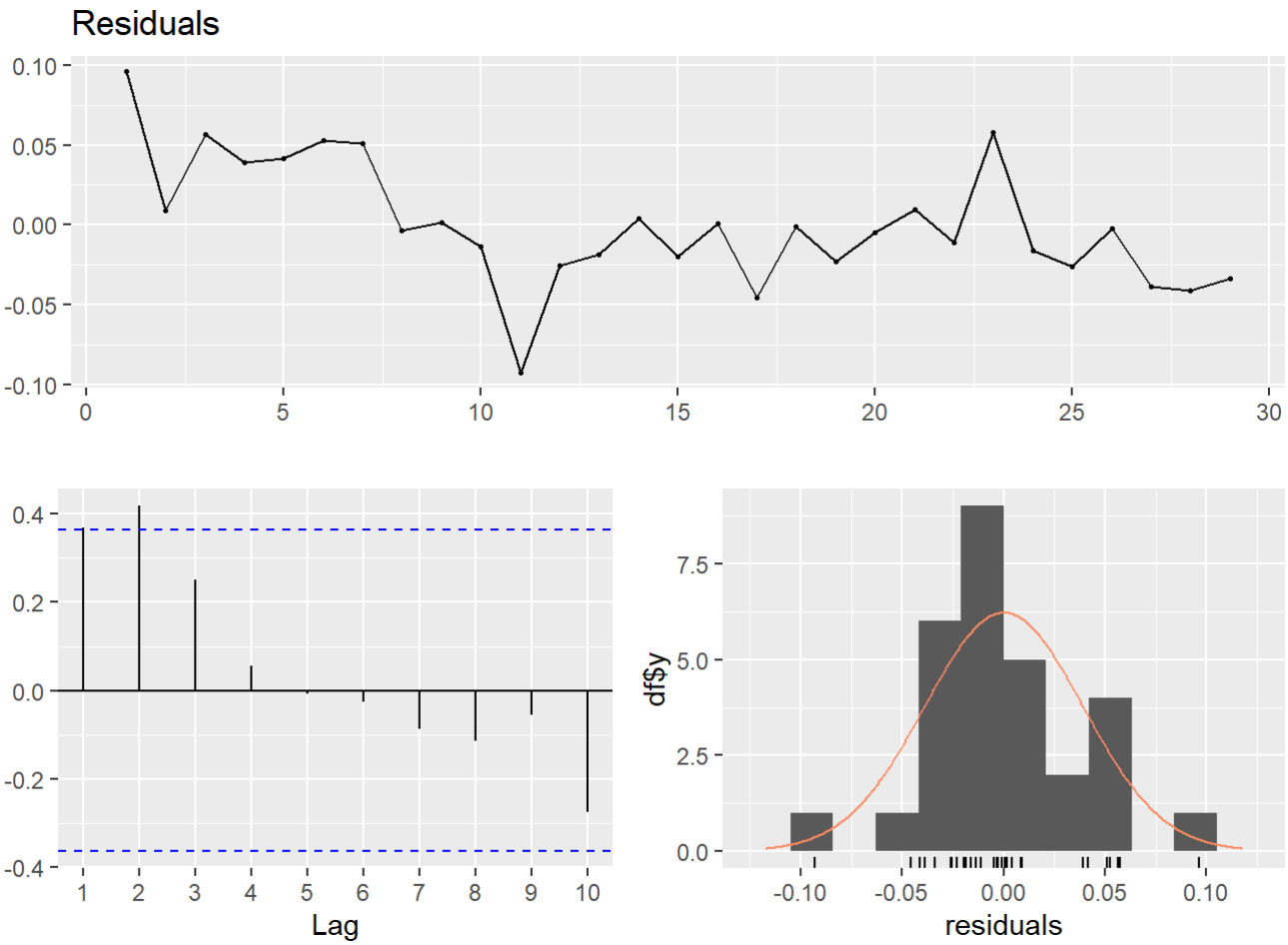
```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.092950 -0.023092 -0.003408  0.009246  0.096239
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.49915    0.07818   6.385  1.1e-06 ***
## z.t0         0.04494    0.02063   2.179    0.039 *
## z.t1        -0.03229    0.05860  -0.551    0.586
## z.t2         0.01253    0.02851   0.440    0.664
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0414 on 25 degrees of freedom
## Multiple R-squared:  0.2784, Adjusted R-squared:  0.1918
## F-statistic: 3.215 on 3 and 25 DF,  p-value: 0.03997
```

From the outputs of **model.rain_poly** model and its summary :

- The p-value is reported at 0.0399 which is less than 5% level of significance.
- The adjusted R-squared value reported was 0.1918 which is very poor.
- Two out of four of the coefficients are significant.
- The F-statistic test is reported at 3.21 on 3 and 25 Degrees of Freedom.

Residual Analysis of **model.rain_poly**

```
checkresiduals(model.rain_poly$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 7
##
## data:  Residuals
## LM test = 9.5736, df = 7, p-value = 0.214
```

From the outputs of Model **model.rain_poly** residuals :

- The p-value from Breusch-Godfrey test was greater than significance level of 0.05.
- The ACF plot reveals that there is seasonality and correlation exist in residual due to significant lags present in the ACF.
- The time-series plot of residual seems to follow a downward trend overall.
- The histogram does seems to follow normal distribution with outlier on both ends.

So, we can conclude that the model **model.rain_poly** is a decent model with an appropriate composition of residuals.

Koyck Distributed Lag Model

The polynomial DLMs did not produce a satisfactory model based on the R2 value. Let's explore fitting various koyck models that aren't reliant on any specific lag length.

Modelling several possible combinations

```
# Koyck modelling
model.temptr_koyck = koyckDlm(x = as.vector(temptr), y = as.vector(RBO))
model.rain_koyck = koyckDlm(x = as.vector(rain), y = as.vector(RBO))
model.humid_koyck = koyckDlm(x = as.vector(humid), y = as.vector(RBO))
model.radtn_koyck = koyckDlm(x = as.vector(radtn), y = as.vector(RBO))
```

We now have done modelling for various possible models using koyck method.

Evaluating based on MASE score

```
# MASE scores
MASE_koyck2 <- MASE(model.temptr_koyck,model.rain_koyck,model.humid_koyck,model.radtn_koyck)

# Arranging in ascending order

arrange(MASE_koyck2,MASE)
```

```
##                      n       MASE
## model.temptr_koyck 30  0.9535116
## model.humid_koyck  30  0.9559618
## model.radtn_koyck  30  1.0314227
## model.rain_koyck   30 19.1057647
```

Analyzing **model.temptr_koyck**

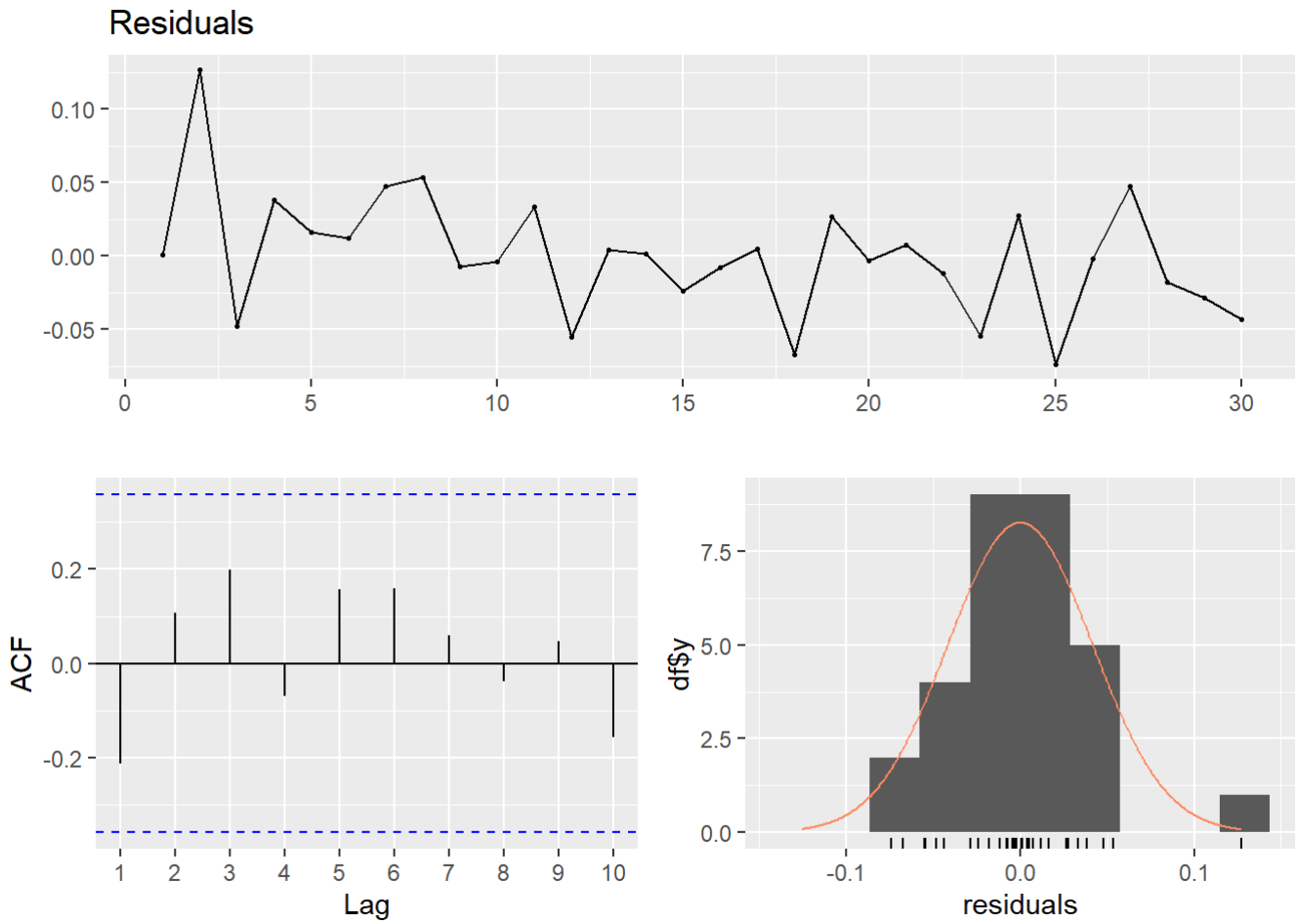
```
summary(model.temptr_koyck,diagnostics = TRUE)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0741656 -0.0225173 -0.0006794  0.0240622  0.1270971
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.20775    0.83741  -0.248   0.8059
## Y.1          0.68547    0.25559   2.682   0.0123 *
## X.t          0.02235    0.03523   0.634   0.5312
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04319 on 27 degrees of freedom
## Multiple R-Squared: 0.1517,  Adjusted R-squared: 0.08891
## Wald test: 5.309 on 2 and 27 DF,  p-value: 0.01136
##
## Diagnostic tests:
##                  df1 df2 statistic    p-value
## Weak instruments   1  27  4.634891 0.04042345
## Wu-Hausman         1  26  1.347013 0.25634707
##
##                               alpha       beta       phi
## Geometric coefficients:  -0.6605142 0.02234675 0.6854665
```

From the outputs of **model.temptr_koyck** model and its summary :

- The p-value is reported at 0.01136 which is less than 5% level of significance.
- The adjusted R-squared value reported was 0.08891 which is very poor.
- One out of four of the coefficients is significant.
- The Wu-Hausman test is reported at 4.63489 on 1 and 26 Degrees of Freedom.

```
checkresiduals(model.temptr_koyck$model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 5.4514, df = 6, p-value = 0.4874
##
## Model df: 0.   Total lags used: 6
```

From the outputs of Model **model.temptr_koyck** residuals :

- The p-value from Breusch-Godfrey test was greater than significance level of 0.05.
- The ACF plot reveals that there is seasonality and correlation exist in residual due to significant lags present in the ACF.
- The time-series plot of residual seems to follow a downward trend overall.
- The histogram does seems to follow normal distribution with outlier on positive end.

So, we can conclude that the model **model.temptr_koyck** is a decent model with an appropriate composition of residuals.

Autoregressive Distributed Lag Model

Calculating autoregressive DLMs across various lag durations and AR process orders through iteration, and selecting the model with the minimum information criterion for fitting.

```
for (i in 1:5){
  for(j in 1:5){
    model.autoreg = ardlDlm(x= as.vector(temptr),y=as.vector(RBO), p = i , q = j )
    cat("p =", i, "q =", j, "AIC =", AIC(model.autoreg$model), "BIC =", BIC(model.autoreg$mod
el), "MASE =", MASE(model.autoreg)$MASE, "\n")
  }
}
```

```
## p = 1 q = 1 AIC = -105.6323 BIC = -98.62635 MASE = 0.8183474
## p = 1 q = 2 AIC = -106.7338 BIC = -98.53007 MASE = 0.7421788
## p = 1 q = 3 AIC = -109.2081 BIC = -99.88269 MASE = 0.7734247
## p = 1 q = 4 AIC = -102.1791 BIC = -91.81242 MASE = 0.8236063
## p = 1 q = 5 AIC = -97.33529 BIC = -86.01242 MASE = 0.7402468
## p = 2 q = 1 AIC = -100.7618 BIC = -92.55807 MASE = 0.9167429
## p = 2 q = 2 AIC = -105.1274 BIC = -95.55629 MASE = 0.7379592
## p = 2 q = 3 AIC = -107.2334 BIC = -96.57577 MASE = 0.7724694
## p = 2 q = 4 AIC = -100.1907 BIC = -88.52822 MASE = 0.8207833
## p = 2 q = 5 AIC = -95.74261 BIC = -83.16164 MASE = 0.7353386
## p = 3 q = 1 AIC = -101.7805 BIC = -92.45505 MASE = 0.9406755
## p = 3 q = 2 AIC = -107.3386 BIC = -96.681 MASE = 0.7740433
## p = 3 q = 3 AIC = -105.3553 BIC = -93.36549 MASE = 0.7735463
## p = 3 q = 4 AIC = -98.69016 BIC = -85.73179 MASE = 0.8250965
## p = 3 q = 5 AIC = -94.15321 BIC = -80.31415 MASE = 0.7628024
## p = 4 q = 1 AIC = -96.62629 BIC = -86.25959 MASE = 0.9476737
## p = 4 q = 2 AIC = -100.9245 BIC = -89.26198 MASE = 0.8489344
## p = 4 q = 3 AIC = -99.22742 BIC = -86.26905 MASE = 0.8460958
## p = 4 q = 4 AIC = -97.42765 BIC = -83.17345 MASE = 0.8471965
## p = 4 q = 5 AIC = -95.44763 BIC = -80.35047 MASE = 0.778824
## p = 5 q = 1 AIC = -96.37589 BIC = -85.05302 MASE = 0.7857653
## p = 5 q = 2 AIC = -97.48107 BIC = -84.90011 MASE = 0.756288
## p = 5 q = 3 AIC = -95.70698 BIC = -81.86792 MASE = 0.7620542
## p = 5 q = 4 AIC = -93.77188 BIC = -78.67472 MASE = 0.7576551
## p = 5 q = 5 AIC = -93.93583 BIC = -77.58057 MASE = 0.7335979
```

Based on the presented results, the ARDL(5,5) model emerges as the top performer according to the AIC, BIC, and MASE metrics. The **finiteDLMauto** function was employed to substitute other predictors to determine the best p & q values. It was observed that the ARDL(5,5) configuration consistently provided optimal results for every predictor variable. Therefore, for all the autoregressive DLMs, we will use p & q values of (5,5).

Creating Autoregressive Model with several possible combinations

```
model.temptr_ardl = ardlDlm(x=as.vector(temptr), y=as.vector(RBO), p = 5, q = 5)
model.rain_ardl = ardlDlm(x=as.vector(rain), y=as.vector(RBO), p = 5, q = 5)
model.humid_ardl = ardlDlm(x=as.vector(humid), y=as.vector(RBO), p = 5, q = 5)
model.radtn_ardl = ardlDlm(x=as.vector(radtn), y=as.vector(RBO), p = 5, q = 5)
```

Evaluating based on AIC,BIC and MASE scores

```
sort.score(AIC(model.temptr_ardl$model,model.rain_ardl$model,model.humid_ardl$model,model.rad
tn_ardl$model), score = "aic")
```

```
##                          df       AIC
## model.humid_ardl$model  13 -96.63220
## model.temptr_ardl$model 13 -93.93583
## model.radtn_ardl$model  13 -93.64810
## model.rain_ardl$model   13 -90.68282
```

```
sort.score(BIC(model.temptr_ardl$model,model.rain_ardl$model,model.humid_ardl$model,model.rad
tn_ardl$model), score = "bic")
```

```
##                          df       BIC
## model.humid_ardl$model  13 -80.27694
## model.temptr_ardl$model 13 -77.58057
## model.radtn_ardl$model  13 -77.29285
## model.rain_ardl$model   13 -74.32757
```

```
Maseardl2 <- MASE(model.temptr_ardl,model.rain_ardl,model.humid_ardl,model.radtn_ardl)

arrange(Maseardl2,MASE)
```

```
##                    n      MASE
## model.humid_ardl  26 0.7128379
## model.radtn_ardl  26 0.7145698
## model.temptr_ardl 26 0.7335979
## model.rain_ardl   26 0.7469974
```

Analyzing **model.humid_ardl**

```
summary(model.humid_ardl)
```

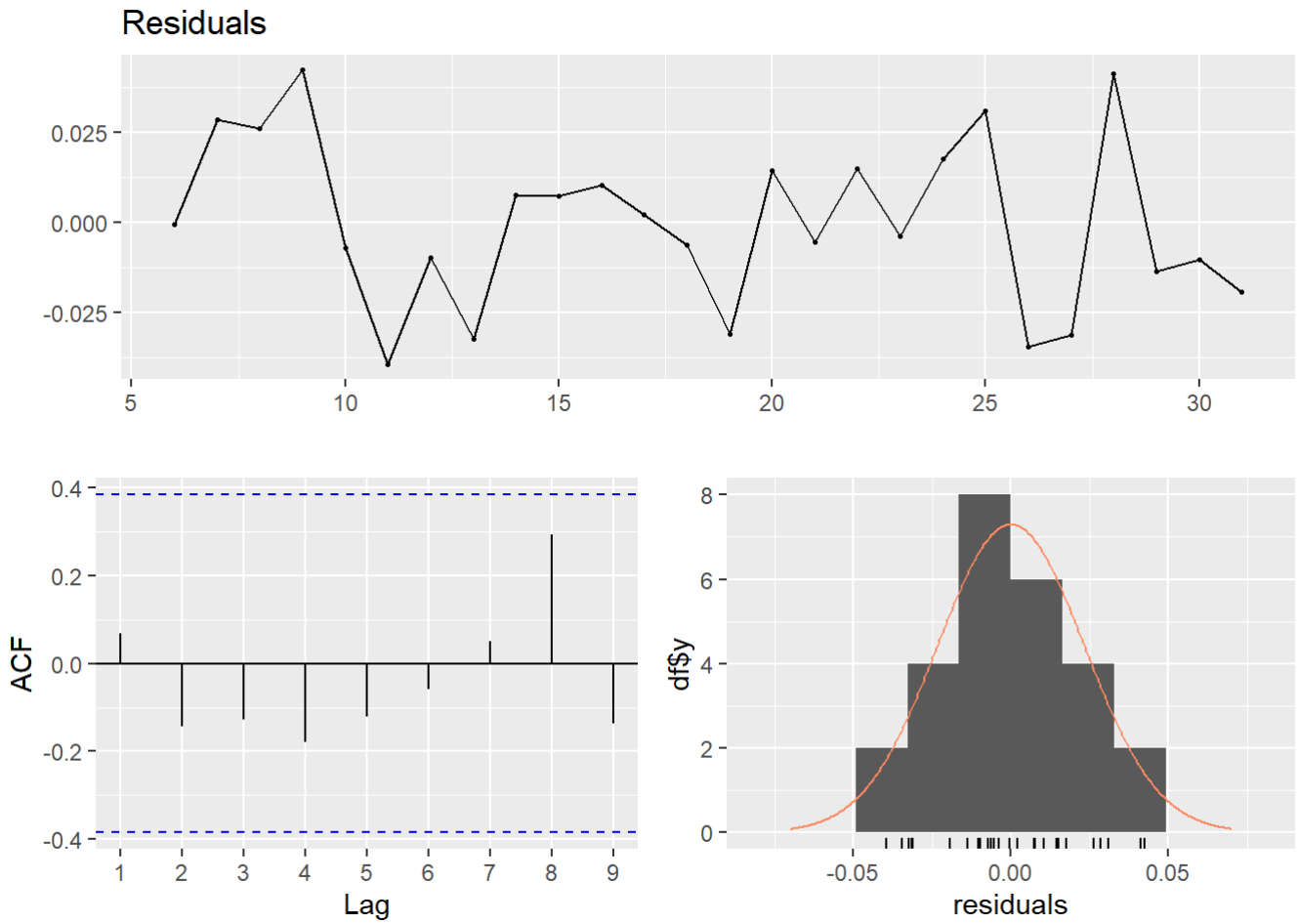
```
##
## Time series regression with "ts" data:
## Start = 6, End = 31
##
## Call:
## dynlm(formula = as.formula(model.text), data = data, start = 1)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.039514 -0.012819 -0.002017  0.014921  0.042583
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.189518   3.619113  -1.434   0.1736
## X.t          0.020748   0.011586   1.791   0.0950 .
## X.1         -0.005077   0.010123  -0.501   0.6238
## X.2          0.021221   0.011187   1.897   0.0787 .
## X.3         -0.002882   0.011434  -0.252   0.8047
## X.4          0.001211   0.009630   0.126   0.9017
## X.5          0.019552   0.011086   1.764   0.0996 .
## Y.1          0.449210   0.238192   1.886   0.0802 .
## Y.2          0.418296   0.283249   1.477   0.1619
## Y.3          0.059211   0.233143   0.254   0.8032
## Y.4          0.059871   0.237302   0.252   0.8045
## Y.5          0.021591   0.221332   0.098   0.9237
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03119 on 14 degrees of freedom
## Multiple R-squared:  0.6849, Adjusted R-squared:  0.4373
## F-statistic: 2.766 on 11 and 14 DF,  p-value: 0.03819
```

From the outputs of **model.humid_ardl** model and its summary :

- The p-value is reported at 0.0381 which is less than 5% level of significance.
- The adjusted R-squared value reported was 0.4373 which is descent.
- Some of the coefficients of model are significant.
- The F-statistic test is reported at 2.76 on 11 and 14 Degrees of Freedom.

Residual Analysis of model.humid_ardl

```
checkresiduals(model.humid_ardl$model,test = FALSE)
```



From the outputs of Model **model.humid_ardl** residuals :

- The ACF plot reveals that there is no seasonality and correlation exist in residual due to no significant lags present in the ACF.
- The time-series plot of residual seems to follow a downward trend overall.
- The histogram does seems to follow normal distribution overall.

So, we can conclude that the model **model.humid_ardl** is a decent model with an appropriate composition of residuals.

Analyzing **model.radtn_ardl**

```
summary(model.radtn_ardl)
```

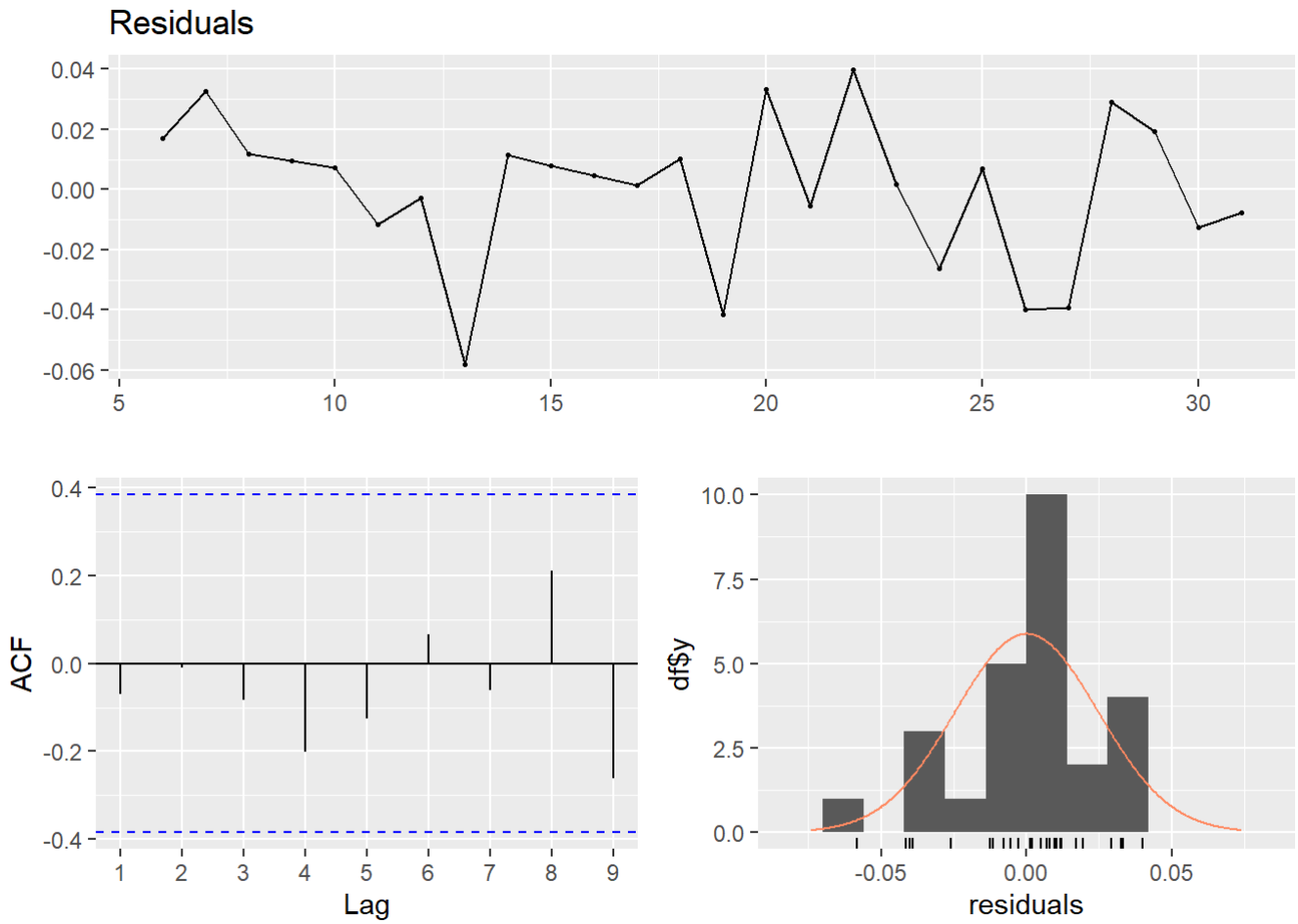
```
##
## Time series regression with "ts" data:
## Start = 6, End = 31
##
## Call:
## dynlm(formula = as.formula(model.text), data = data, start = 1)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.058181 -0.010561  0.005937  0.011888  0.039893
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.235092   0.635369  -0.370   0.7169
## X.t         -0.025859   0.020766  -1.245   0.2335
## X.1          0.026744   0.023598   1.133   0.2761
## X.2          0.008284   0.023664   0.350   0.7315
## X.3          0.006908   0.023343   0.296   0.7716
## X.4          0.004317   0.024792   0.174   0.8643
## X.5          0.006035   0.022186   0.272   0.7896
## Y.1          0.456956   0.259256   1.763   0.0998 .
## Y.2          0.235367   0.266174   0.884   0.3915
## Y.3          0.101679   0.243130   0.418   0.6821
## Y.4         -0.176699   0.222300  -0.795   0.4400
## Y.5          0.170775   0.240520   0.710   0.4893
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03303 on 14 degrees of freedom
## Multiple R-squared:  0.6466, Adjusted R-squared:  0.3689
## F-statistic: 2.328 on 11 and 14 DF,  p-value: 0.06941
```

From the outputs of **model.radtn_ardl** model and its summary :

- The p-value is reported at 0.0694 which is greater than 5% level of significance.
- The adjusted R-squared value reported was 0.3689 which is descent.
- Only one of the coefficient of model is significant.
- The F-statistic test is reported at 2.32 on 11 and 14 Degrees of Freedom.

Residual Analysis of **model.radtn_ardl**

```
checkresiduals(model.radtn_ardl$model,test = FALSE)
```



From the outputs of Model **model.radtn_ardl** residuals :

- The ACF plot reveals that there is no seasonality and correlation exist in residual due to no significant lags present in the ACF.
- The time-series plot of residual seems to follow a downward trend overall.
- The histogram does seems to follow normal distribution overall.

So, we can conclude that the model **model.radtn_ardl** is a decent model with an appropriate composition of residuals.

Modelling using Dynamic Linear Models

Creating several possible combinations of Dynamic models

```
# Fitting univariate models with RBO

dynamic_model1 <- dynlm(RBO ~ L(RBO , k = 1 )+ trend(RBO))
dynamic_model2 <- dynlm(RBO~ L(RBO , k = 2 )+ trend(RBO))
dynamic_model3 <-dynlm(RBO~ L(RBO , k = 1 ))
dynamic_model4 <-dynlm(RBO~ L(RBO , k = 1 )+ L(RBO , k = 2 ))
dynamic_model5 <-dynlm(RBO~ L(RBO , k = 1 )+ L(RBO , k = 2 )+L(RBO , k = 3 )+trend(RBO))
dynamic_model6 <-dynlm(RBO~ L(RBO , k = 1 )+ L(RBO , k = 2 )+L(RBO , k = 3 ) + L(RBO)+trend(R
BO))

# Fitting models with temptr as independent variable

temptr_dynlm <-dynlm(RBO ~ temptr + L(RBO, k=1))
temptr_dynlm1 <-dynlm(RBO ~ temptr + L(RBO, k=2))
temptr_dynlm2 <-dynlm(RBO ~ temptr)
temptr_dynlm3 <-dynlm(RBO ~ temptr - 1)
temptr_dynlm4 <-dynlm(RBO ~ temptr + L(temptr, k=1))
temptr_dynlm5 <-dynlm(RBO ~ temptr + L(temptr, k=2))
temptr_dynlm6 <-dynlm(RBO ~ temptr + L(temptr, k=3))
temptr_dynlm7 <-dynlm(RBO ~ temptr + L(RBO, k=3))

# Fitting models with rain as independent variable

raindynlm <-dynlm(RBO ~ rain + L(RBO, k=1))
raindynlm1 <-dynlm(RBO ~ rain + L(RBO, k=2))
raindynlm2 <-dynlm(RBO ~ rain)
raindynlm3 <-dynlm(RBO ~ rain - 1)
raindynlm4 <-dynlm(RBO ~ rain + L(rain, k=1))
raindynlm5 <-dynlm(RBO ~ rain + L(rain, k=2))
raindynlm6 <-dynlm(RBO ~ rain + L(RBO, k=3))
raindynlm7 <-dynlm(RBO ~ rain + L(rain, k=3))

# Fitting models with humid as independent variable

humid_dynlm <- dynlm(RBO ~ humid + L(RBO, k=1))
humid_dynlm1 <- dynlm(RBO ~ humid + L(RBO, k=2))
humid_dynlm2 <- dynlm(RBO ~ humid)
humid_dynlm3 <- dynlm(RBO ~ humid - 1)
humid_dynlm4 <- dynlm(RBO ~ humid + L(humid, k=1))
humid_dynlm5 <- dynlm(RBO ~ humid + L(humid, k=2))
humid_dynlm6 <- dynlm(RBO ~ humid + L(RBO, k=3))
humid_dynlm7 <- dynlm(RBO ~ humid + L(humid, k=3))

# Fitting models with radtn as independent variable

radtn_dynlm <- dynlm(RBO ~ radtn + L(RBO, k=1))
radtn_dynlm1 <- dynlm(RBO ~ radtn + L(RBO, k=2))
radtn_dynlm2 <- dynlm(RBO ~ radtn)
radtn_dynlm3 <- dynlm(RBO ~ radtn - 1)
radtn_dynlm4 <- dynlm(RBO ~ radtn + L(radtn, k=1))
radtn_dynlm5 <- dynlm(RBO ~ radtn + L(radtn, k=2))
radtn_dynlm6 <- dynlm(RBO ~ radtn + L(radtn, k=3))
radtn_dynlm7 <- dynlm(RBO ~ radtn + L(RBO, k=3))
```

We have successfully fitted models above using dynamic linear method. We will now evaluate them based on MASE scores.

Evaluating models based on MASE

```
dynlm_MASE3 <- MASE(lm(temptr_dynlm), lm(temptr_dynlm1), lm(temptr_dynlm2), lm(temptr_dynlm
3), lm(temptr_dynlm4),lm(temptr_dynlm5),lm(temptr_dynlm6),lm(temptr_dynlm7),lm(raindynlm),lm
(raindynlm1),lm(raindynlm2),lm(raindynlm3),
lm(raindynlm4),lm(raindynlm5),lm(raindynlm6),lm(raindynlm7),lm(humid_dynlm),lm(humid_dynlm1),
lm(humid_dynlm2),
lm(humid_dynlm3),lm(humid_dynlm4),lm(humid_dynlm5),lm(humid_dynlm6),lm(humid_dynlm7),lm(radtn
_dynlm),lm(radtn_dynlm1),lm(radtn_dynlm2),lm(radtn_dynlm3),lm(radtn_dynlm4),lm(radtn_dynlm5),
lm(radtn_dynlm6),lm(radtn_dynlm7),lm(dynamic_model1),lm(dynamic_model2),lm(dynamic_model3),lm
(dynamic_model4),lm(dynamic_model5))

arrange(dynlm_MASE3,MASE)
```

```
##                      n      MASE
## lm(dynamic_model1) 30 0.8252265
## lm(temptr_dynlm)   30 0.8260073
## lm(radtn_dynlm)    30 0.8434230
## lm(humid_dynlm)    30 0.8517631
## lm(raindynlm)      30 0.8528680
## lm(dynamic_model5) 28 0.8555020
## lm(dynamic_model3) 30 0.8559138
## lm(dynamic_model2) 29 0.8900758
## lm(dynamic_model4) 29 0.9008622
## lm(temptr_dynlm1)  29 0.9129771
## lm(raindynlm1)     29 0.9225460
## lm(raindynlm4)     30 0.9417954
## lm(radtn_dynlm1)   29 0.9641928
## lm(raindynlm6)     28 0.9688508
## lm(temptr_dynlm2)  31 0.9918512
## lm(temptr_dynlm4)  30 1.0056219
## lm(humid_dynlm1)   29 1.0058463
## lm(radtn_dynlm7)   28 1.0175256
## lm(raindynlm2)     31 1.0280987
## lm(temptr_dynlm7)  28 1.0546593
## lm(radtn_dynlm4)   30 1.0630293
## lm(humid_dynlm6)   28 1.0649315
## lm(radtn_dynlm2)   31 1.0799582
## lm(humid_dynlm4)   30 1.1010995
## lm(humid_dynlm2)   31 1.1065670
## lm(raindynlm5)     29 1.1090943
## lm(radtn_dynlm6)   28 1.1295639
## lm(temptr_dynlm5)  29 1.1311104
## lm(raindynlm7)     28 1.1481806
## lm(temptr_dynlm6)  28 1.1660166
## lm(radtn_dynlm5)   29 1.1667122
## lm(humid_dynlm3)   31 1.1808328
## lm(humid_dynlm7)   28 1.1968852
## lm(humid_dynlm5)   29 1.2370385
## lm(radtn_dynlm3)   31 1.4041012
## lm(temptr_dynlm3)  31 1.5252468
## lm(raindynlm3)     31 2.6180543
```

From the above results, we can say that model **dynamic_model1** turns out to be the best with respect to the MASE scores.

Analyzing dynamic_model1 using summary & residual analysis

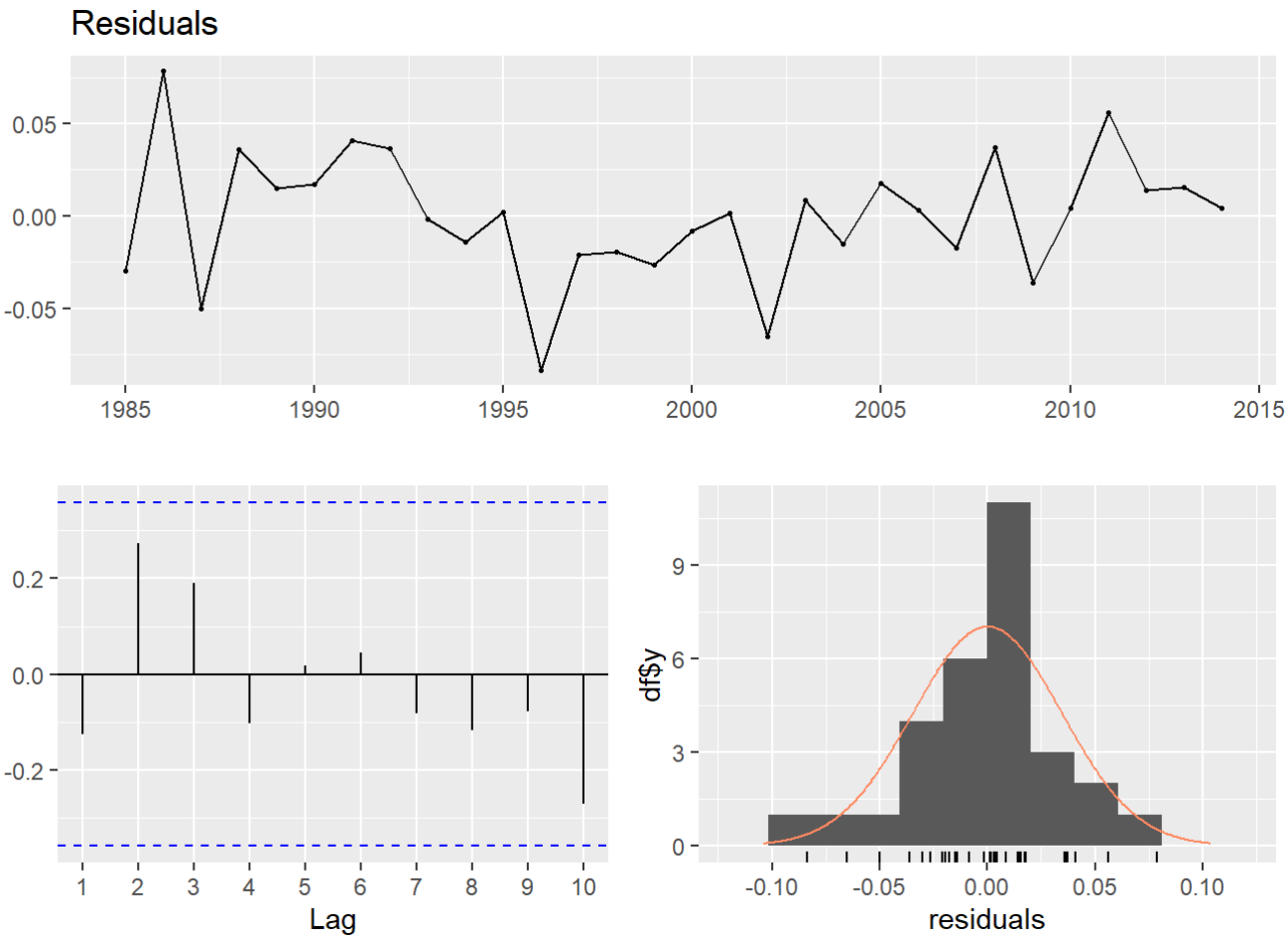
```
summary(dynamic_model1)
```

```
##
## Time series regression with "ts" data:
## Start = 1985, End = 2014
##
## Call:
## dynlm(formula = RBO ~ L(RBO, k = 1) + trend(RBO))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.083854 -0.018945  0.002608  0.016801  0.078745
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.5010057  0.1406937   3.561   0.0014 **
## L(RBO, k = 1)  0.3622445  0.1781223   2.034   0.0519 .
## trend(RBO)    -0.0018993  0.0009113  -2.084   0.0467 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03587 on 27 degrees of freedom
## Multiple R-squared:  0.415,  Adjusted R-squared:  0.3717
## F-statistic: 9.579 on 2 and 27 DF,  p-value: 0.0007181
```

From the outputs of **dynamic_model1** model and its summary :

- The p-value is reported at 0.000718 which is greater than 5% level of significance.
- The adjusted R-squared value reported was 0.3717 which is descent.
- All of the coefficients of this model are significant.
- The F-statistic test is reported at 9.579 on 2 and 27 Degrees of Freedom.

```
checkresiduals(dynamic_model1)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 6
##
## data:  Residuals
## LM test = 5.9191, df = 6, p-value = 0.4323
```

From the outputs of Model **dynamic_model1** residuals :

- The p-values is reported at 0.432 which is greater than 5% level of significance.
- The ACF plot reveals that there is no seasonality and correlation exist in residual due to no significant lags present in the ACF.
- The time-series plot of residual seems to follow a upward trend overall.
- The histogram does seems to follow normal distribution overall.

So, we can conclude that the model **model.radtn_ardl** is a decent model with an appropriate composition of residuals.

Overview of Task 3

```
# Summarising table

overview2 <- data.frame(Model = c("model.rain","model.nointercepthum","model.rain_poly","mode
l.humid_koyck","model.temptr_ardl","model.radtn_ardl","dynamic_model1"), MASE= c(0.9417954,0.
9918515,0.9993747,0.8400135,0.7335979,0.7145698,0.8252265))


# Evaluating MASE score for all

arrange(overview2,MASE)
```

```
##                  Model      MASE
## 1     model.radtn_ardl 0.7145698
## 2    model.temptr_ardl 0.7335979
## 3       dynamic_model1 0.8252265
## 4    model.humid_koyck 0.8400135
## 5           model.rain 0.9417954
## 6 model.nointercepthum 0.9918515
## 7      model.rain_poly 0.9993747
```

According to the MASE metric, the top-performing models from all the DLM and dynamic linear techniques are ranked based on their MASE scores. Notably, the models from the ARDL method (namely **model.radtn_ardl** & **model.temptr_ardl**) yield the best MASE scores. We'll utilize these two models to forecast RBO for the upcoming **three** years.

Forecasting RBO series

Three years ahead values for both radiation and temperature

```
# Employing Radiation
task2_covariate[,4]
```

```
## # A tibble: 4 × 1
##   Radiation
##       <dbl>
## 1      14.6
## 2      14.6
## 3      14.8
## 4      14.8
```

```
# Employing Temperature
task2_covariate[,2]
```

```
## # A tibble: 4 × 1
##   Temperature
##         <dbl>
## 1        20.7
## 2        20.5
## 3        20.5
## 4        20.6
```

Calculating point predictions and confidence intervals for the **model.radtn_ardl**.

```
# # Calculating the three-year ahead RBO using the initial three radiation values from task3_
covariate.

# Creating point forecasts

forecast_RBO_point <- dLagM::forecast(model.radtn,x = c(14.60,14.56,14.79) ,h = 3)

forecast_RBO_point <- round(forecast_RBO_point$forecasts,2)
```

Calculating point predictions and confidence intervals for the **model.temptr_ardl**

```
# # Calculating the three-year ahead RBO using the initial three temperature values from task
3_covariate.

# Creating point forecasts

forecast_RBO_point2 <- dLagM::forecast(model.temptr,x = c(20.74,20.49,20.52) ,h = 3)

forecast_RBO_point2 <- round(forecast_RBO_point2$forecasts,2)
```

Plotting the Forecasted Model

```
plot(ts(c(as.vector(RBO),forecast_RBO_point),start=1984),type="o",col="darkmagenta", ylab="RB
O_forecasted",
main="Figure 64: RBO three year ahead predicted values from 2015-2017")
lines(ts(c(as.vector(RBO),forecast_RBO_point2),start=1984),type="o",col="darkorange2")
lines(ts(as.vector(RBO),start=1984),col="black",type="o")
legend("topright", lty = 1,pch=1,text.width=11, col = c("darkmagenta","darkorange2","black"),
c("model.radtn_ardl","model.temptr_ardl","Data (RBO)"))
```

Figure 64: RBO three year ahead predicted values from 2015-2017

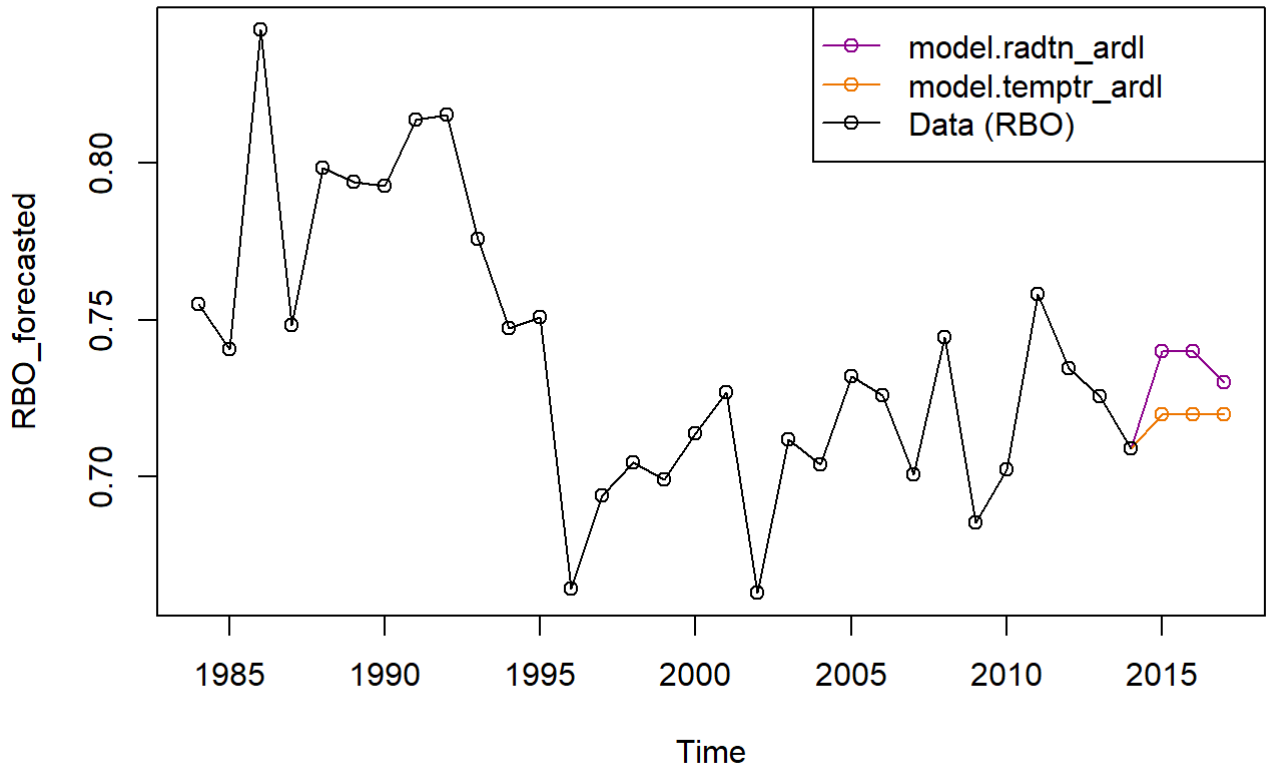


Figure 64

The forecasts provided by the two models show slight variances. According to the model.radtn_ardl, the RBO value is predicted to dip in 2015, only to rise in the subsequent two years. Conversely, the model.temptr_ardl suggests a consistent upward trajectory for the RBO values over the next three years.

Both models effectively captured the historical RBO data. Furthermore, the projections they offer can be considered trustworthy given the narrowness of their 95% prediction intervals. Summarizing, both ARDL models suggest that from 2015 to 2017, there will be a marked increase in RBO values, indicating a greater alignment in the timing of the first flowering events.

Task 3 Part(b)- Model Fitting for Drought affected RBO

Modelling the drough effect

```
# Incorporating the impact of the drought that occurred in 1996 as a pulse effect.

Y.t = RBO
T = 13 # 13 years effect on the series
S.t = 1*(seq(Y.t)>=T)
S.t.1 =Lag(S.t,+1)


# Fitting Dynamic linear models

model1 =dynlm(Y.t~ L(Y.t , k = 1 )+S.t+ trend(Y.t))
model2 =dynlm(Y.t~ L(Y.t , k = 2 )+S.t+ trend(Y.t))
model3 =dynlm(Y.t~ L(Y.t , k = 1 )+S.t)
model4 =dynlm(Y.t~ L(Y.t , k = 1 )+S.t+trend(Y.t))
model5 =dynlm(Y.t~ L(Y.t , k = 1 )+ L(Y.t , k = 2 )+S.t+S.t.1)
model6 =dynlm(Y.t~ L(Y.t , k = 1 )+ L(Y.t , k = 2 )+S.t.1)
model7 = dynlm(Y.t~ L(Y.t , k = 1 )+ L(Y.t , k = 2 )+ L(Y.t , k = 3 )+S.t.1)
model8 = dynlm(Y.t~ L(Y.t , k = 1 )+ L(Y.t , k = 2 )+L(Y.t , k = 3 )+S.t+S.t.1+ trend(Y.t))
model9 = dynlm(Y.t~rain + L(Y.t , k = 1 ) + L(Y.t , k = 2 )+ S.t+ + S.t.1 + trend(Y.t))
model10 = dynlm(Y.t~temptr + L(Y.t , k = 1 ) + L(Y.t , k = 2 )+ S.t+ + S.t.1 + trend(Y.t))
model11 = dynlm(Y.t~radtn + L(Y.t , k = 1 ) + L(Y.t , k = 2 )+ S.t+ + S.t.1 + trend(Y.t))
model12 = dynlm(Y.t~humid + L(Y.t , k = 1 ) + L(Y.t , k = 2 )+ S.t+ + S.t.1 + trend(Y.t))
```

Evaluating models based on MASE

```
dynlm_MASE5 <- MASE(lm(model1),lm(model2),lm(model3),lm(model4),lm(model5),lm(model6),lm(mode
l7),lm(model8),lm(model9),lm(model10),lm(model11),lm(model12))

head(arrange(dynlm_MASE5,MASE))
```

```
##              n      MASE
## lm(model9)  29 0.6560217
## lm(model11) 29 0.6734863
## lm(model8)  28 0.6748310
## lm(model10) 29 0.6762744
## lm(model1)  30 0.6768533
## lm(model4)  30 0.6768533
```

Based on the results presented, we'll examine model9 (which uses rain as a predictor) since it boasts the lowest MASE value among all the evaluated models.

Analyzing model9 using summary & residual analysis

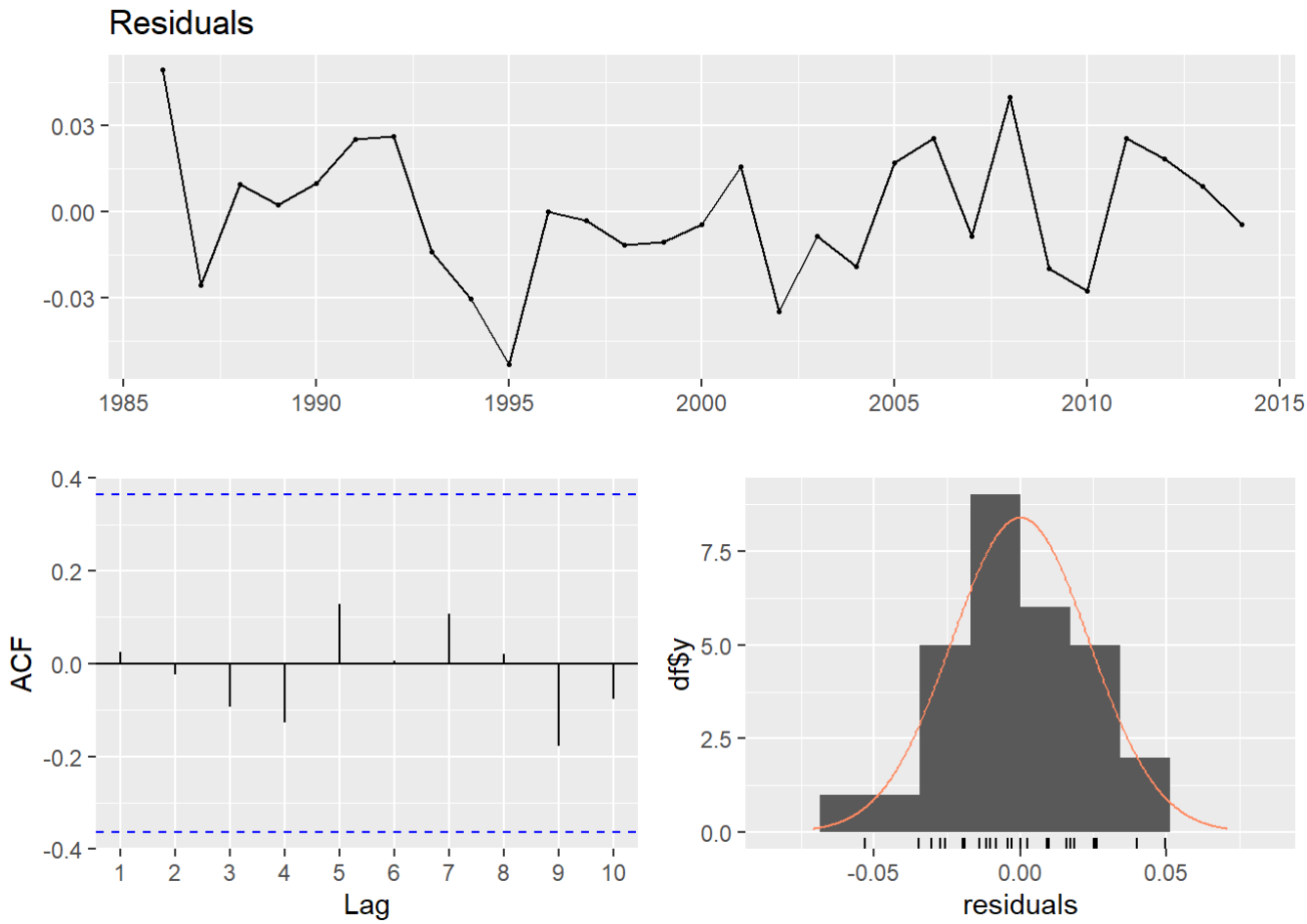
```
summary(model9)
```

```
##
## Time series regression with "ts" data:
## Start = 1986, End = 2014
##
## Call:
## dynlm(formula = Y.t ~ rain + L(Y.t, k = 1) + L(Y.t, k = 2) +
##     S.t + +S.t.1 + trend(Y.t))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.053222 -0.014019 -0.002957  0.017038  0.049497
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.9100154  0.2987065   3.047 0.005919 **
## rain           0.0216702  0.0175875   1.232 0.230904
## L(Y.t, k = 1) -0.1762220  0.2140198  -0.823 0.419122
## L(Y.t, k = 2) -0.0536160  0.2226122  -0.241 0.811902
## S.t           -0.1403487  0.0335485  -4.183 0.000385 ***
## S.t.1          0.0477295  0.0420454   1.135 0.268513
## trend(Y.t)     0.0004731  0.0015555   0.304 0.763847
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02663 on 22 degrees of freedom
## Multiple R-squared:  0.7372, Adjusted R-squared:  0.6655
## F-statistic: 10.28 on 6 and 22 DF,  p-value: 1.861e-05
```

From the summary of model9, we can deduce the following points:

- Most variables in the model are not significant at the 5% significance level, with the exception of the intercept and S.t.
- The adjusted R2 value is quite good, standing at 66.55%. This suggests that the dynamic model can account for approximately 66.55% of the variability in the dependent variable, which is RBO.
- The residuals range from a minimum of -0.053222 to a maximum of 0.049497, and the residual standard error (RSE) is 0.02663.
- Based on the F-test for overall model significance, model9 is deemed highly significant at the 5% significance level.

```
checkresiduals(model9)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 4.6585, df = 10, p-value = 0.9128
```

From the Residual analysis we can say that, Based on the BG test, there is no evidence of serial correlation in the residuals. The distribution of the residuals appears to be random. The ACF plot does not display any significant lags, indicating a lack of clear autocorrelation in the residuals. The histogram suggests that the residuals are not normally distributed.

Forecasting for RBO

Plot of the original series alongside the model's predicted values.

```
par(mfrow=c(1,1))
plot(Y.t,ylab='RBO',xlab='Year',main = "Figure 65: Time series plot of the  yearly RBO with f
itted model9")
lines(model9$fitted.values,col="blue")
```

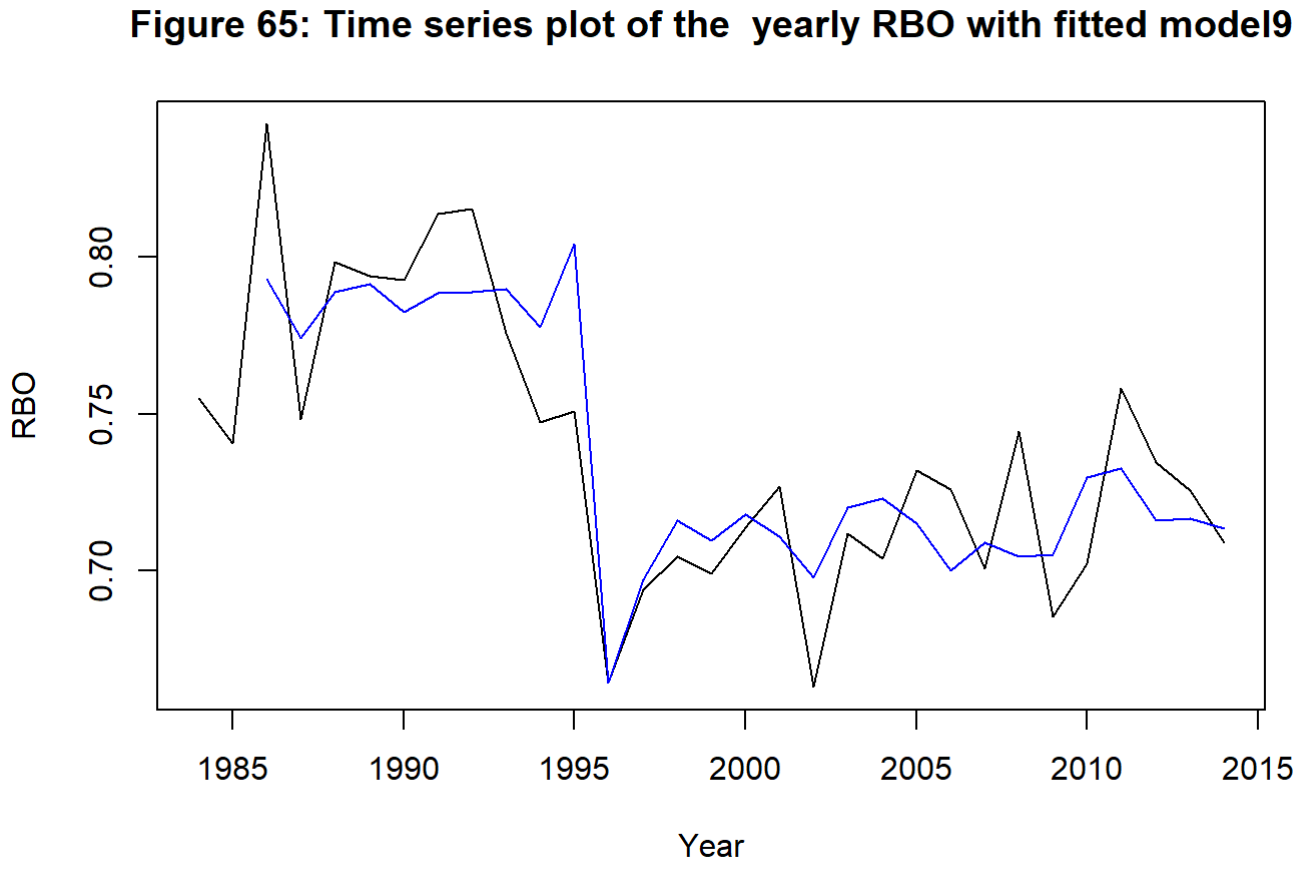


Figure 65

From the above observations, it's evident that the model doesn't perfectly match all the prior RBO values. Yet, when evaluated on metrics like MASE, R2, and the F-test, it emerges as the best. Let's proceed to predict the upcoming three years using the dynamic model.

Plotting with 3 year ahead forecast

```
q = 3
n = nrow(model9$model)
rbo.frc = array(NA , (n + q))
rbo.frc[1:n] = Y.t[3:length(Y.t)]
trend = array(NA,q)
trend.start = model9$model[n,"trend(Y.t)"]
trend = seq(trend.start , trend.start + q/12, 1/12)

for(i in 1:q){
data.new =c(1,1,rbo.frc[n-2+i],rbo.frc[n-2+i],1,1,trend[i])
rbo.frc[n+i] = as.vector(model9$coefficients) %*% data.new
}

plot(RBO,xlim=c(1984,2018),
ylab='RBO',xlab='Year',
main = "Figure 66: Time series plot of RBO with 3 years ahead forecast")
lines(ts(rbo.frc[(n+1):(n+q)],start=c(2015)),col="blue")
```

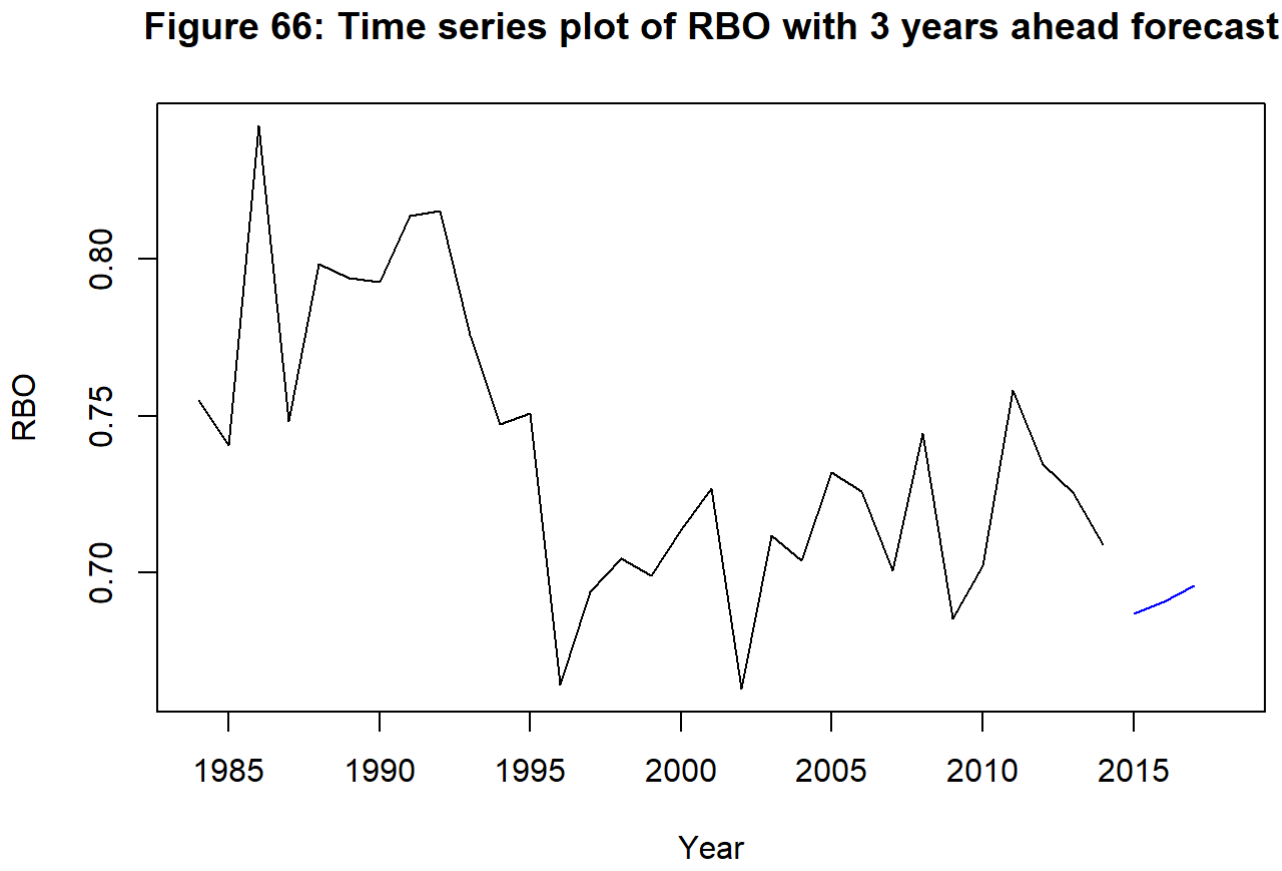


Figure 66

The displayed graph illustrates a four-year forecast based on model9. According to this prediction, the RBO values in the upcoming three years are expected to be below the past average RBO figures, attributed to the impact of drought as indicated by the dynamic linear model.

Point Forecast for next three years

```
round(rbo.frc[(n+1):(n+q)],2)
```

```
## [1] 0.69 0.69 0.70
```

The displayed results indicate forecasts for the upcoming three years, suggesting that the predicted RBO values will range between 0.69 and 0.70. These projected values are below the average RBO values observed in past years. This decrease might be attributed to the prolonged drought that spanned from 1996 to 2009. However, given that the model didn't aptly fit the RBO series, the reliability of these forecasts is questionable.

Conclusion

The analysis for Task 3 was effectively carried out on the RBO.csv dataset. In section (a), we undertook a univariate study on all five series (RBO, temperature, rainfall, radiation, relhumidity). Since their frequencies were under 2, these series couldn't be broken down. A variety of univariate models were employed to forecast the RBO values for the upcoming three years (2015-2017) using various modeling techniques (**DLM, koyck, polyDLM, ARDL, and dynlm**). The optimal models, as determined by metrics like R2, F-test, AIC, BIC, and MASE, were two ARDL versions with temperature and radiation as their respective predictors. Both these models were used to predict the RBO for the subsequent three years and aligned well with historical RBO data.

However, while the temperature-based model anticipated a consistent RBO rise, the radiation-based model forecasted an initial dip in RBO in the first year followed by growth in 2016-2017. The collective inference from these models is that there's likely to be a steady RBO increase for the years 2015-2017.

For section (b), we applied univariate **dynlm** models after factoring in the drought that spanned 13 years (from 1996-2009). The most promising model, in terms of MASE, R2, and F-test, turned out to be model9, which used rainfall as a predictor. When this model was applied to the main series, it significantly deviated from the actual values, indicating its forecasting limitations. Despite being the top-performing model, its predictions were below the RBO average, potentially due to the prolonged drought from 1996-2009. Given the model's inability to closely match the historical RBO values, the predictions it provided can't be deemed entirely trustworthy.

References

- RMIT University, School of Science, Mathematical Sciences, MATH1307 Forecasting, Module 3: Time Series Regression Models I - Distributed lag models - By using dLagM R-package {https://rmit.instructure.com/courses/112639/modules/items/5127056 (https://rmit.instructure.com/courses/112639/modules/items/5127056)}
- RMIT University, School of Science, Mathematical Sciences, MATH1307 Forecasting,Module 4 - Time Series Regression Models II - Dynamic Models {https://rmit.instructure.com/courses/112639/files/30915164?module_item_id=5127064&fd_cookie_set=1 (https://rmit.instructure.com/courses/112639/files/30915164?module_item_id=5127064&fd_cookie_set=1)}
- RMIT University, School of Science, Mathematical Sciences, MATH1307 Forecasting, Module 5 - Exponential Smoothing Methods {https://rmit.instructure.com/courses/112639/pages/module-5-introduction?module_item_id=5127067 (https://rmit.instructure.com/courses/112639/pages/module-5-introduction?module_item_id=5127067)}
- RMIT University, School of Science, Mathematical Sciences, MATH1307 Forecasting, Module 6 - Linear Innovations State Space Models {https://rmit.instructure.com/courses/112639/pages/module-6-introduction?module_item_id=5127072 (https://rmit.instructure.com/courses/112639/pages/module-6-introduction?module_item_id=5127072)}
- Jonathan D. Cryer , Kung-Sik Chan, Time Series Analysis With Applications in R, Textbook 2008 {https://link-springer-com.ezproxy.lib.rmit.edu.au/book/10.1007/978-0-387-75959-3 (https://link-springer-com.ezproxy.lib.rmit.edu.au/book/10.1007/978-0-387-75959-3)}