MATH1307 Forecasting Final Project

Adity Bhargav (s3814507)

Table of Contents

Introduction

The assignment report is divided into three different independent tasks and in each task we have to analyze different time series. The first task requires to analyze five different weekly series- mortality, temperature, pollutants particle size and two chemical emissions (chem1, chem2) between 2010-2020 and give 4 weeks ahead forecasts for mortality. The second task we have to analyze FFD and give its four year ahead prediction using different climate indicators. In task 3, we have two parts (a & b) where we have to analyze RBO and give three year ahead predictions by modelling different climate predictors for it one by one (uni variate analysis). In part (b) of task 3 we will again give three year ahead prediction for RBO by incorporating the Australian drought period from 1996-2009.

Task 1 - Time series analysis & forecast of mortality series

Data Description

Between 2010 and 2020, a team of researchers looked at disease-specific mean weekly mortality in Paris, France, as well as the city's local environment (temperature degrees Fahrenheit), size of pollutants, and levels of toxic chemical emissions from automobiles and industries in the air. The 'mort' file contains weekly data on all 5 series i.e. mortality, temperature, pollutants particle size and two chemical emissions (chem1, chem2) between 2010-2020 (508 time points).

Objective & Methodology

The main objective of task 1 is to analyze the five series and give 4 week ahead forecast for mortality.

1. Firstly, necessary data prepossessing steps will be performed on all the five series. The data file 'mort.csv' will be checked for missing/unique/unknown values and will be dealt with accordingly. Each series from 'mort.csv' will also be converted to time-series to demonstrate weekly values for all the five series to carry out a time series analysis.

2. The next step will be to check and understand the nature/characteristics of all the time series by plotting respective time series plots and plotting them together. All the five series will also be checked for stationarity through ACF plots and Dicker-Fuller Unit tests.
3. The correlation between each series will be checked by plotting a correlation chart for all of them. The series will also be decomposed using STL decomposition method (as X12 is not compatible with weekly data) to understand the characteristics of the series more deeply.
4. Further, different distributed lag models will also be fitted on the series.
5. Different type of DLMs, dynlms, exponential smoothing & state-space models will also be fitted by converting them to monthly data depending upon the results generated from previous steps.
6. Finally, the best model/models based on R squared, AIC, BIC, MASE etc. will be selected for predicting the values of four weeks ahead mortality.

Data Prepossessing

Before executing the task, we need to perform some necessary data prepossessing steps to make the data set ready for analysis.

1. Importing dataset into Rstudio:

```
# Importing dataset `mort'

mort <- read_csv("/cloud/mort .csv")
```

```
## New names:
## * `` -> ...1
```

```
## Rows: 508 Columns: 6
```

```
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (6): ...1, mortality, temp, chem1, chem2, particle size
```

```
##
## 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.
```

```
# Checking it has been loaded successfully

head(mort)
```

```
## # A tibble: 6 × 6
##    ...1 mortality  temp chem1 chem2 `particle size`
##   <dbl>     <dbl> <dbl> <dbl> <dbl>           <dbl>
## 1     1     11.9   72.4  3.37  45.8            72.7
## 2     2     10.8   67.2  2.59  43.9            49.6
## 3     3      9.33  62.9  3.29  32.2            55.7
## 4     4      9.54  72.5  3.04  40.4            55.2
## 5     5      8.27  74.2  3.39  48.5            66.0
## 6     6      7.55  67.9  2.57  48.6            44.0
```

```
# Checking class

class(mort)
```

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

2. Checking for missing and special values:

Checking the dataset for missing values-

```
# Checking for missing values in `mort'

colSums(is.na(mort))
```

```
##          ...1     mortality          temp         chem1         chem2
##             0             0             0             0             0
## particle size
##             0
```

As per the above output there are no missing values in the dataset, which makes it a consistent dataset. Now checking for special values-

```
#Checking for special values

is.specialorNA <- function(x){
  if (is.numeric(x)) (is.infinite(x) | is.nan(x) | is.na(x))
}

sapply(mort, function(x) sum( is.specialorNA(x) ))
```

```
##          ...1     mortality          temp         chem1         chem2
##             0             0             0             0             0
## particle size
##             0
```

As per the above outputs, there are no missing and special values found which means the `mort` dataset is clean and ready for analysis.

3. Deleting extra column in `mort`:

```
# Deleting all the unnecessary columns from mort

mort = subset(mort, select = -c(1))
```

4. Converting each variable to time series and storing separately:

The five different series will be extracted from mort Then they will converted to time series and stored in separate datasets.The dataset mort will also be converted to time-series.

```
# Converting 'mort' to time series and storing in 'mort_ts'

mort_ts <- ts(mort, start = c(2010,1), frequency = 52)

# Converting mortality from 'mort' to time series and storing in 'mor'

mor <- ts(mort$mortality, start = c(2010,1), frequency = 52)


# Converting temp from 'mort' to time series and storing in 'temp'

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


# Converting chem1 from 'mort' to time series and storing in 'chem1'

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


# Converting chem2 from 'mort' to time series and storing in 'chem2'

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



# Converting particle size from 'mort' to time series and storing in 'part'

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



# Checking conversion has been successfully done for each series

class(mort_ts)
```

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

```
class(mor)
```

```
## [1] "ts"
```

```
class(temp)
```

```
## [1] "ts"
```

```
class(chem1)
```

```
## [1] "ts"
```

```
class(chem2)
```

```
## [1] "ts"
```

```
class(part)
```

```
## [1] "ts"
```

The conversion was successfully done.The mort_ts data was converted to time series and each of the five variables in the mort_ts were individually converted to five different series containing time-series data of weekly average values of mortality, temperature, chemical 1, chemical 2 and particle size respectively.

Understanding Nature of Series

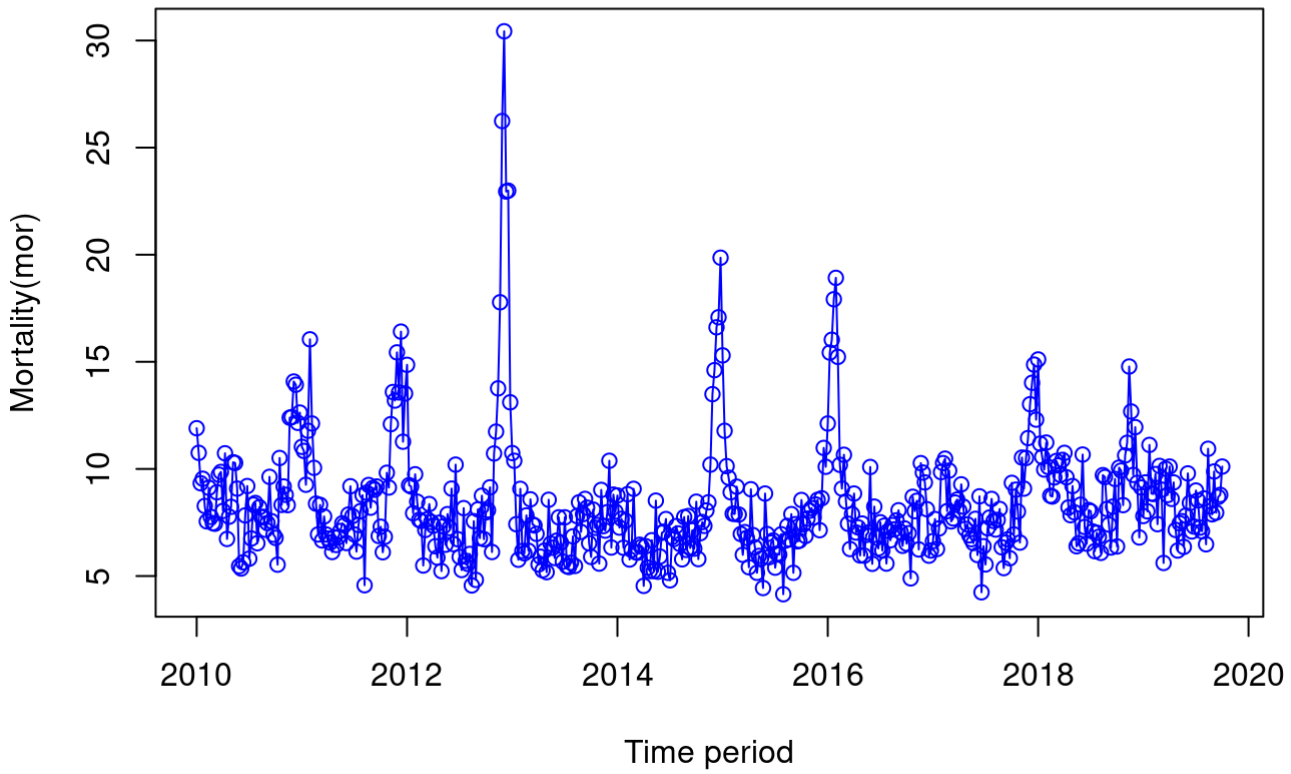
In order to understand the nature of each of the five series, we will plot a time-series plot for each of the series separately and together. We will also calculate correlation of the dataset `mort_ts` .

Time-series plot of mortality mort:

A time-series plot of the weekly averages of mortalityis plotted using plot function.

```
plot(mor, ylab='Mortality(mor)', xlab='Time period', type='o', col='blue', main = 'Figure 1:
 Time-series plot of mor')
```

Figure 1: Time-series plot of mor



Following observations were seen from the time series plot of the mor series-

1.) Trend : There seems to be no presence of a particular trend in the series.

2.) Seasonality : Seasonality can be observed with higher values observed in the later weeks of 2012 and starting of 2013.There can be higher values observed between 2014 and 2016.

3.) Changing variance : No particular change in variance can be observed in the series due to seasonality.

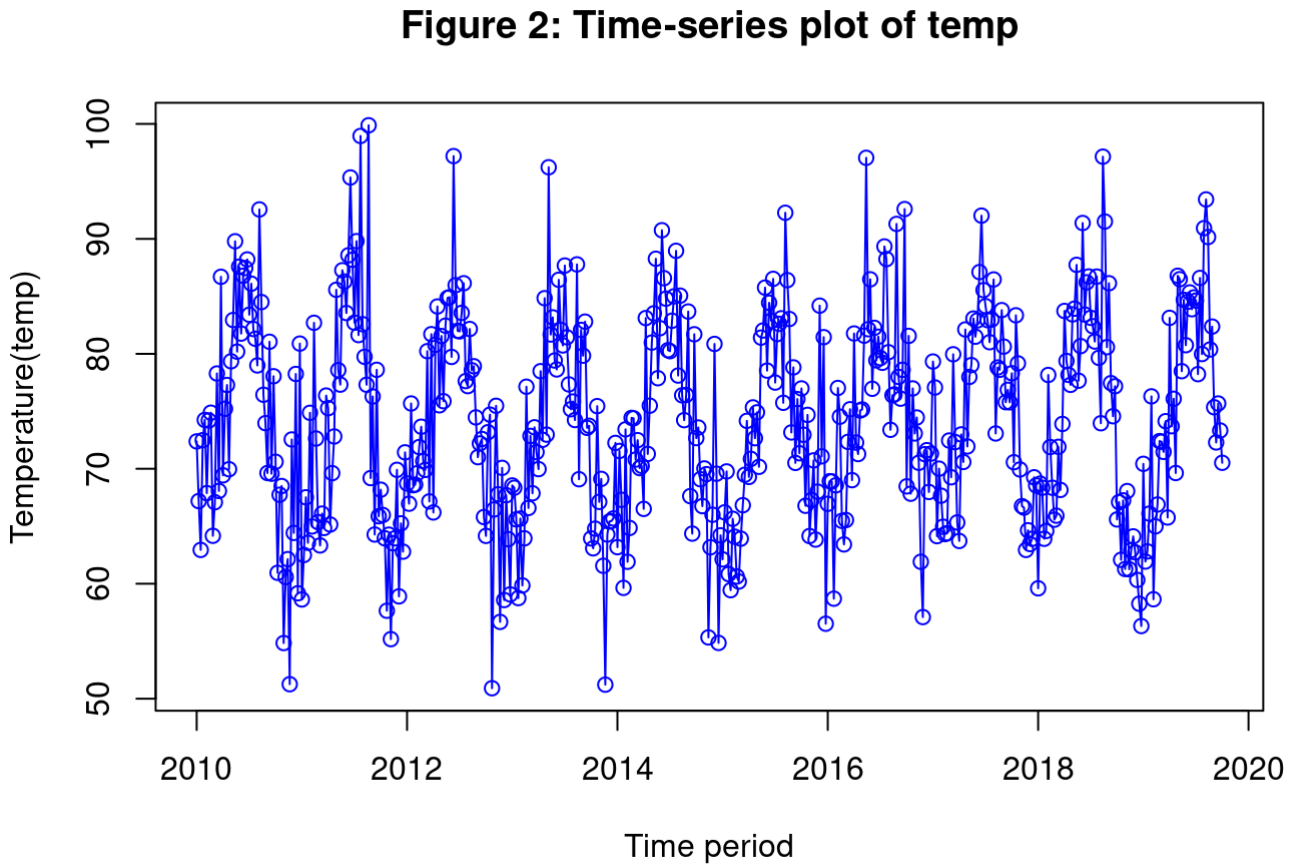
4.) Intervention : Visible intervention point around the year 2012 and 2013 can be observed in the time series plot.

5.) Behavior : No particular behavior can be observed in the series due to seasonality.

Time-series plot of temp:

A time-series plot of weekly averages of temp is plotted using plot function.

```
plot(temp, ylab='Temperature(temp)', xlab='Time period', type='o', col='blue', main = 'Figure 2: Time-series plot of temp')
```



Following observations were seen from the time series plot of the temp series-

1.) Trend : No particular trend is observed in the series.

2.) Seasonality : Seasonality can be observed in the series with changing pattern over time. In the earlier weeks of 2011, highest value can be observed while in the later weeks of 2013, lowest values of temperature can be seen.

3.) Changing variance : No particular change in variance can be observed in the series due to seasonality.

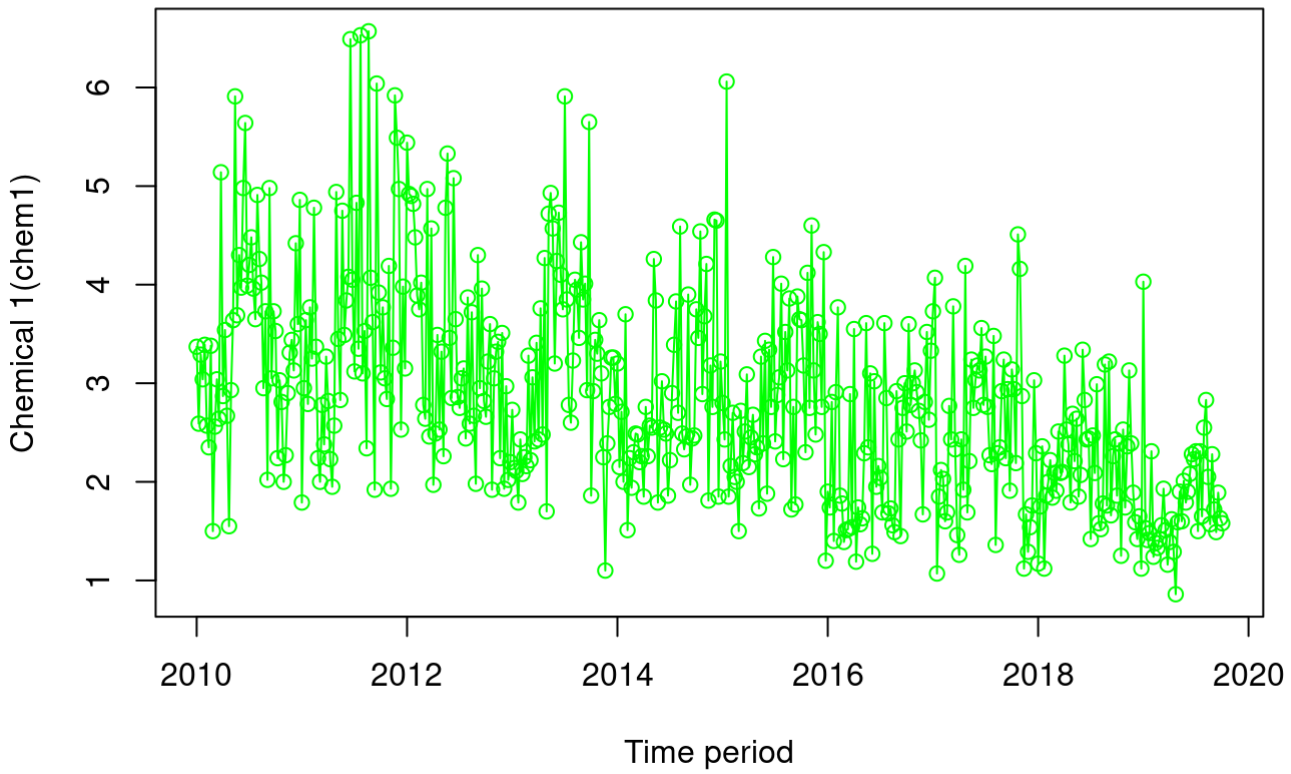
4.) Intervention : No obvious intervention point can be seen.

5.) Behavior : No particular behavior can be observed in the series due to seasonality.

Time-series plot of chem1:

A time-series plot of weekly averages of chem1 is plotted using plot function.

```
plot(chem1, ylab='Chemical 1(chem1)', xlab='Time period', type='o', col='green', main = 'Figu
re 3: Time-series plot of chem1')
```



Following observations were seen from the time series plot of the chem1 series-

1.) Trend : A gradual decreasing pattern can be observed (downward trend).

2.) Seasonality : Seasonality can be observed in the series with changing pattern over time. In the later weeks of 2011, highest value can be observed while in the later weeks of 2013, lowest value of chem1 can be seen.

3.) Changing variance : No particular change in variance can be observed in the series because of seasonality.

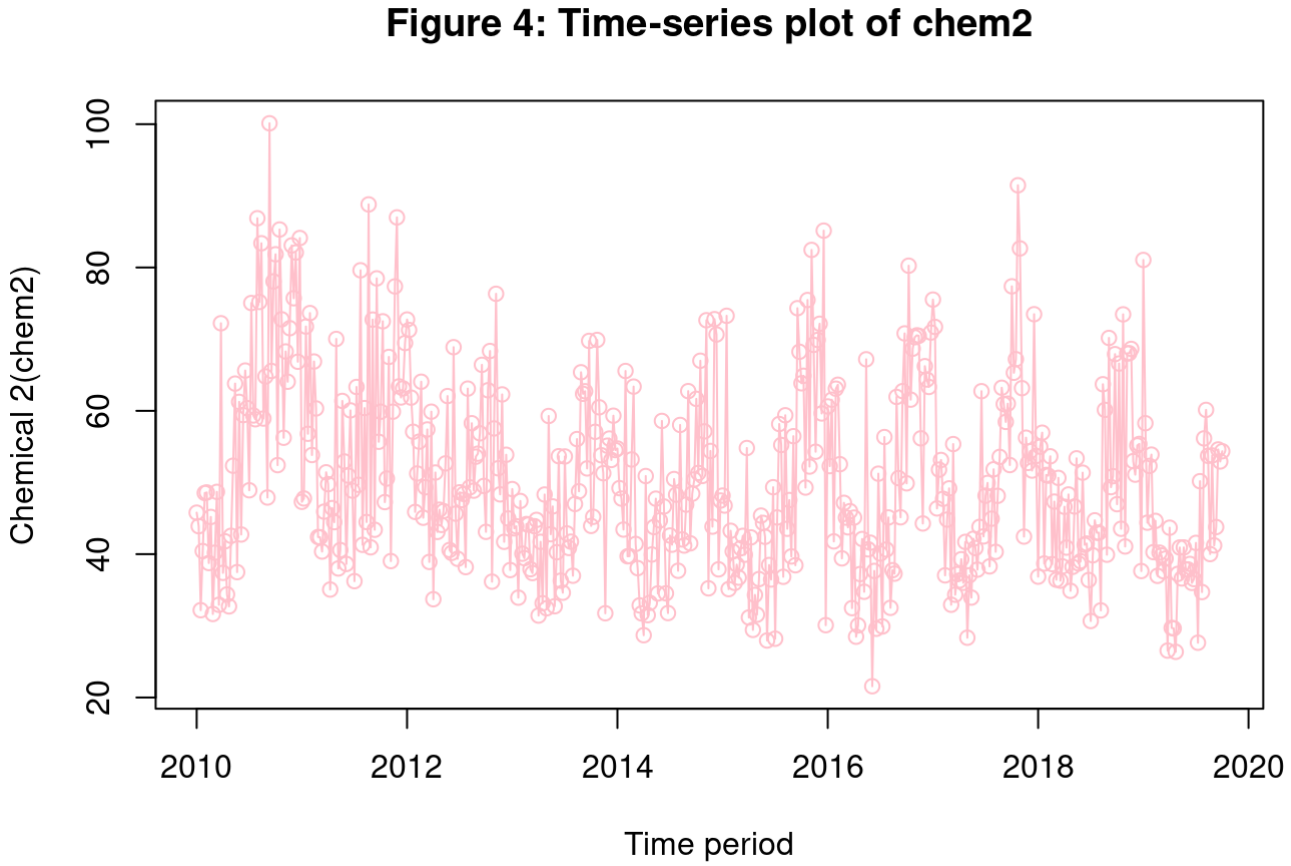
4.) Intervention : No obvious intervention point can be seen in the series.

5.) Behavior : No obvious behavior can be observed in the series due to seasonality.

Time-series plot of chem2:

A time-series plot of weekly averages of chem2 is plotted using plot function.

```
plot(chem2, ylab='Chemical 2(chem2)', xlab='Time period', type='o', col='pink', main = 'Figur
e 4: Time-series plot of chem2')
```



Following observations were seen from the time series plot of the chem2 series-

1.) Trend : No obvious trend observed in the series.

2.) Seasonality : Seasonality can be observed in the series with changing pattern over time. In the later weeks of 2010,highest value can be observed while in the middle weeks of 2016, lowest value of chem2 can be seen.

3.) Changing variance : No particular change in variance can be observed in the series because of seasonality.

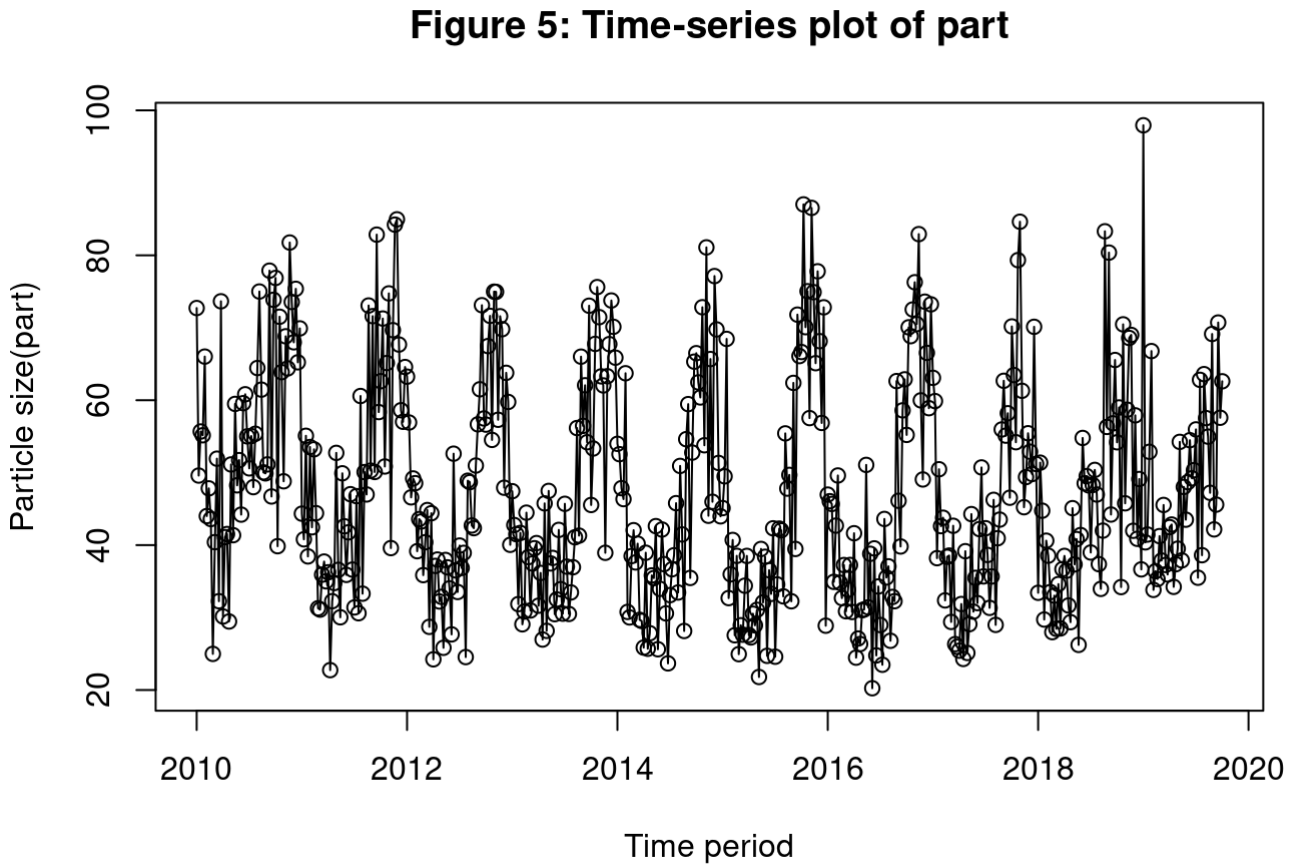
4.) Intervention : No obvious intervention point can be seen in the series.

5.) Behavior : No obvious behavior can be observed in the series due to seasonality.

Time-series plot of part:

A time-series plot of weekly averages of particle size is plotted using plot function.

```
plot(part, ylab="Particle size(part)", xlab="Time period", type="o", col="black", main = "Fig
ure 5: Time-series plot of part")
```



Following observations were seen from the time series plot of the particle size series-

1.) Trend : No obvious trend observed in the series.

2.) Seasonality : Seasonality can be observed in the series with changing pattern over time. In the earlier weeks of 2018, highest value can be observed while in the middle weeks of 2016, lowest value of particle size(part) can be seen.

3.) Changing variance : No particular change in variance can be observed in the series because of seasonality.

4.) Intervention : No obvious intervention point can be seen in the series.

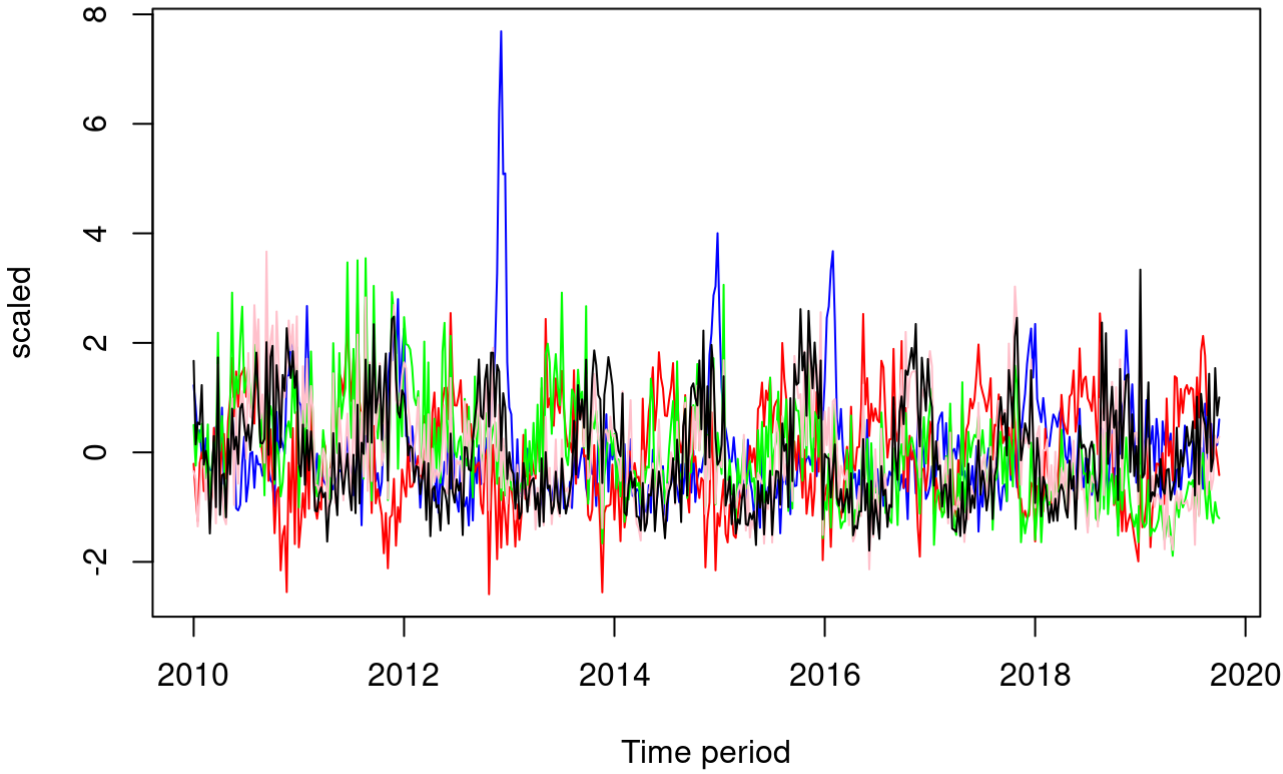
5.) Behavior : No obvious behavior can be observed in the series due to seasonality.

Time-series plot of data set mort_ts:

After analyzing each of the series independently, we will plot them together to have a visual comparison in the nature of the each of the series. In order to plot them together in the same plot, scaling will be done for the `mort_ts` .

```
# Scaling data mort_ts

scaled  = scale(mort_ts)

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

plot(scaled, plot.type = "s", col = c("blue", "red","green","pink","black"), main = "Figure
 6: Time Series plot of  scaled mort_ts",xlab="Time period")
```

Figure 6: Time Series plot of scaled mort_ts



All the five series are likely to be highly correlated and are seasonal in nature. In the next step, correlation between the five series will be calculated to mathematically verify the above visual results- `

Calculating correlation for `mort_ts`

The correlation chart consisting of bivariate scatterplots and correlation value between each of the series was constructed.

```
# Finding Correlation for mort_ts

library("PerformanceAnalytics")
```

```
##
## Attaching package: 'PerformanceAnalytics'
```

```
## The following objects are masked from 'package:timeDate':
##
##     kurtosis, skewness
```

```
## The following objects are masked from 'package:TSA':
##
##     kurtosis, skewness
```

```
## The following object is masked from 'package:graphics':
##
##     legend
```

```
# Constructing scatterplot and calculating correlation using chart.correlation

chart.Correlation(mort_ts, histogram=FALSE, pch=19)

title(main = "Figure 7: Correlation Chart", cex.main= 0.85)
```

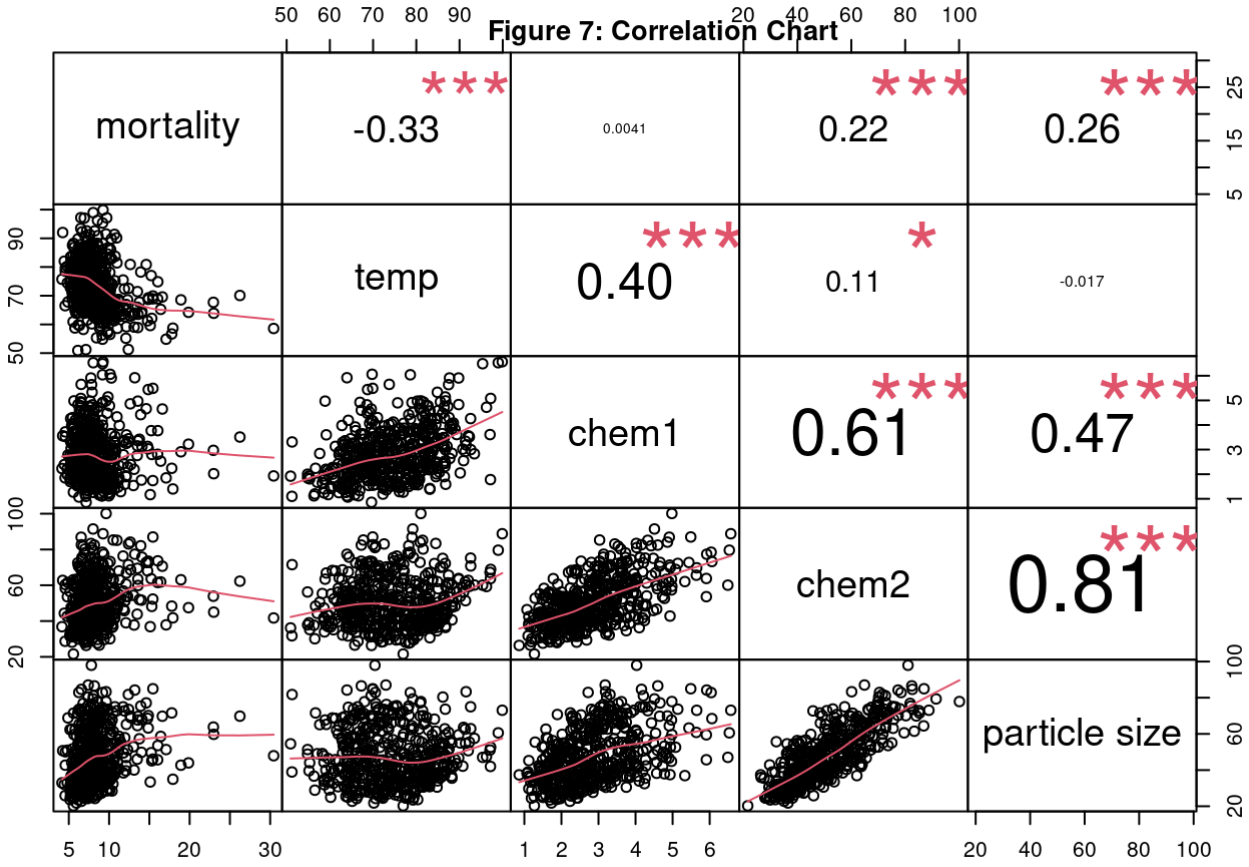


Figure 7: Correlation Chart

- In the above plot the distribution of each variable is shown on the diagonal.
- On the bottom of the diagonal, the bivariate scatter plots with a fitted line are displayed.
- On the top of the diagonal, the value of the correlation plus the significance level as stars are shown.
- Each significance level is related to a star sign : p-values(0, 0.001, 0.01, 0.05, 0.1, 1) <=> symbols("***", "**", "*", ".", " ").

Interpretation:

The scatter plots and the correlation calculated show that there is a very strong positive correlation observed between the chem2 series and particle size series (0.81). Other than that, there is a moderate positive correlation observed between chem1 & chem2 (0.61). Also, there is a minute correlation observed between chem1 & particle size (0.47). Thus, all the five series have some level of correlation between each other as seen from the correlation chart and time-series plot of mort_ts.

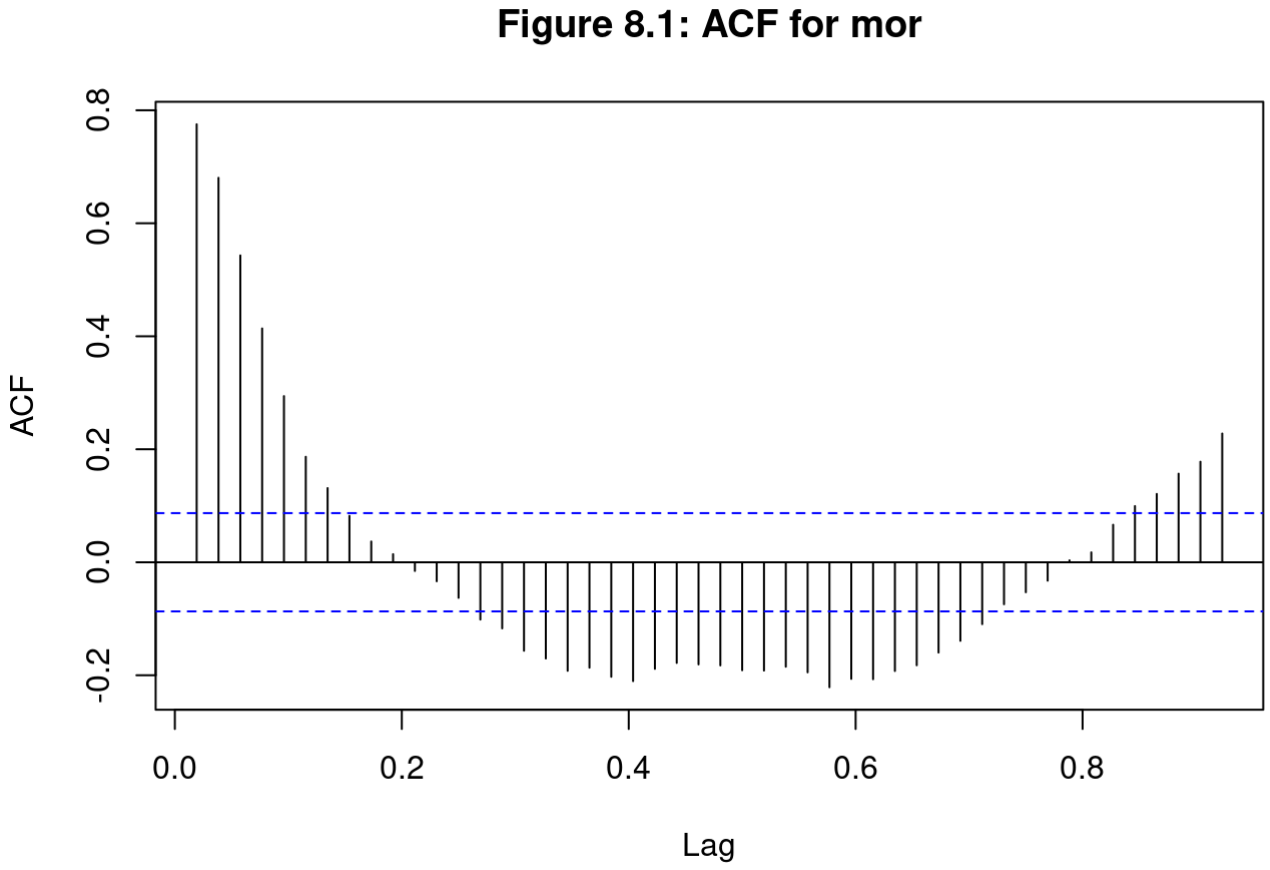
Checking non-stationarity in dataset

Let's now check whether there is existence of non-stationarity in the five series.The first check can be done through visualization by plotting an ACF for each of the series.This can be further explored and verified by performing ADF tests for each series.

Checking stationarity for mor series:

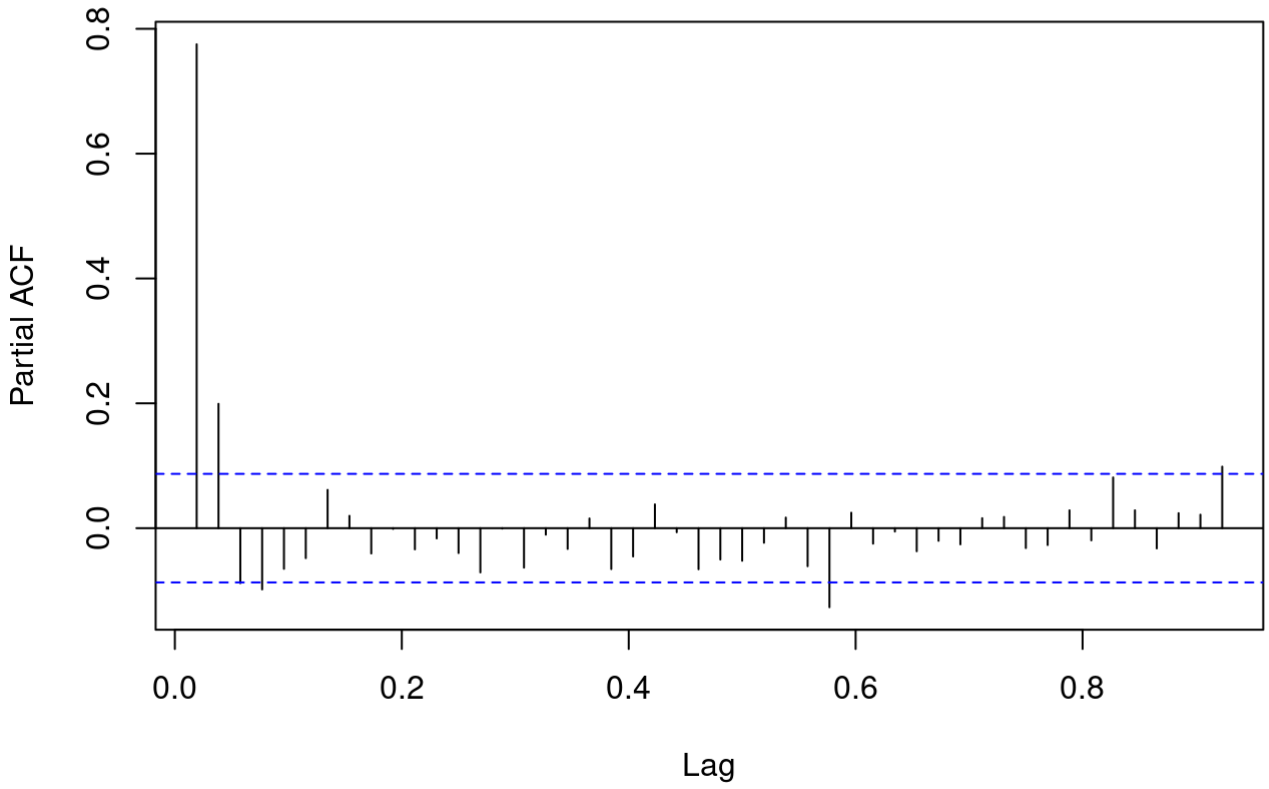
Plotting ACF for mor series:

```
acf(mor,lag.max=48, main = "Figure 8.1: ACF for mor")
```




```
pacf(mor,lag.max=48, main = "Figure 8.2: ACF for mor")
```

Figure 8.2: ACF for mor



It can be seen from the ACF plot above that the series have a trend pattern. The PACF plot reveals that the series has some autocorrelation as first lag is significant.

Dicker-Fuller Unit-Root test (ADF) for mor series:

The ADF has two hypothesis-

H0 indicates series is non-stationary while Ha (alternate hypothesis) say series is stationary.

```
#Performing ADF test on mor series

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  mor
## Dickey-Fuller = -6.9431, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

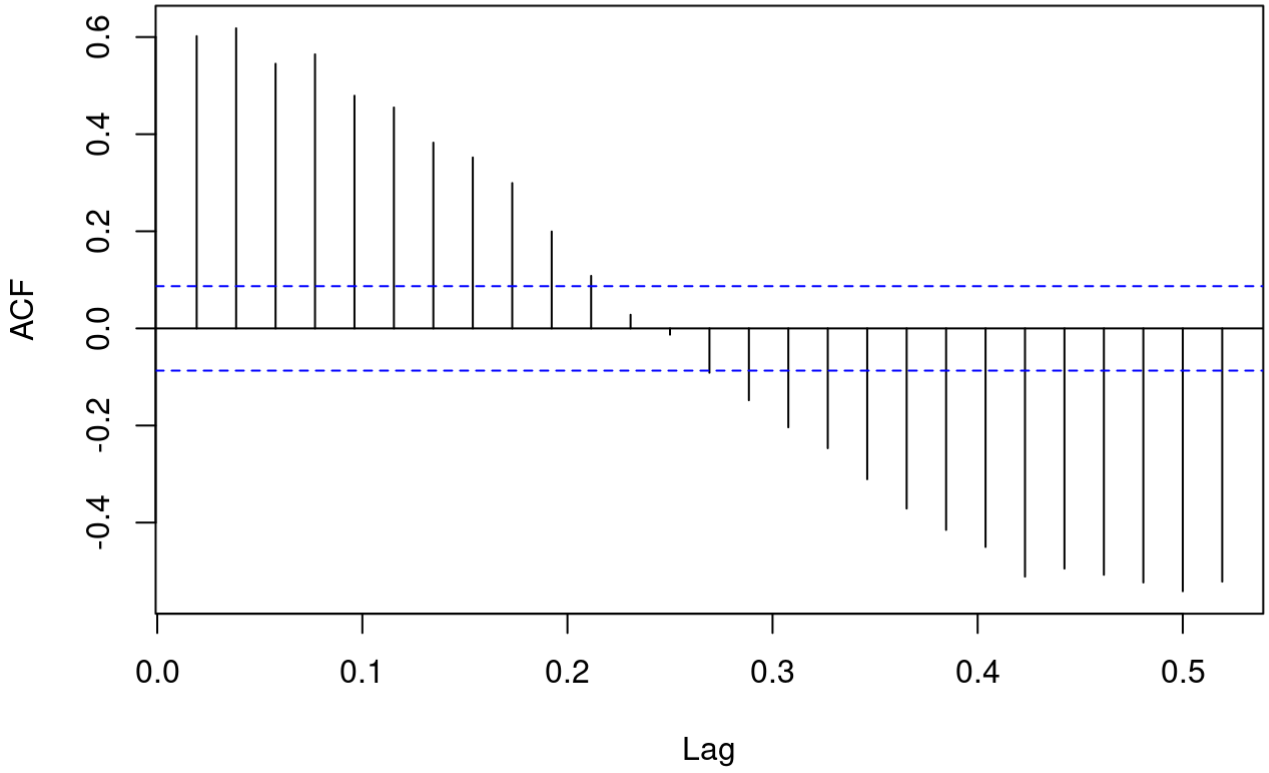
As per the results obtained from the ADF test of 5 lag order, the p-value is less than 5% level of significance which indicates the mortality (mor) series is stationary.

Checking stationarity for temp series:

Plotting ACF for temp series:

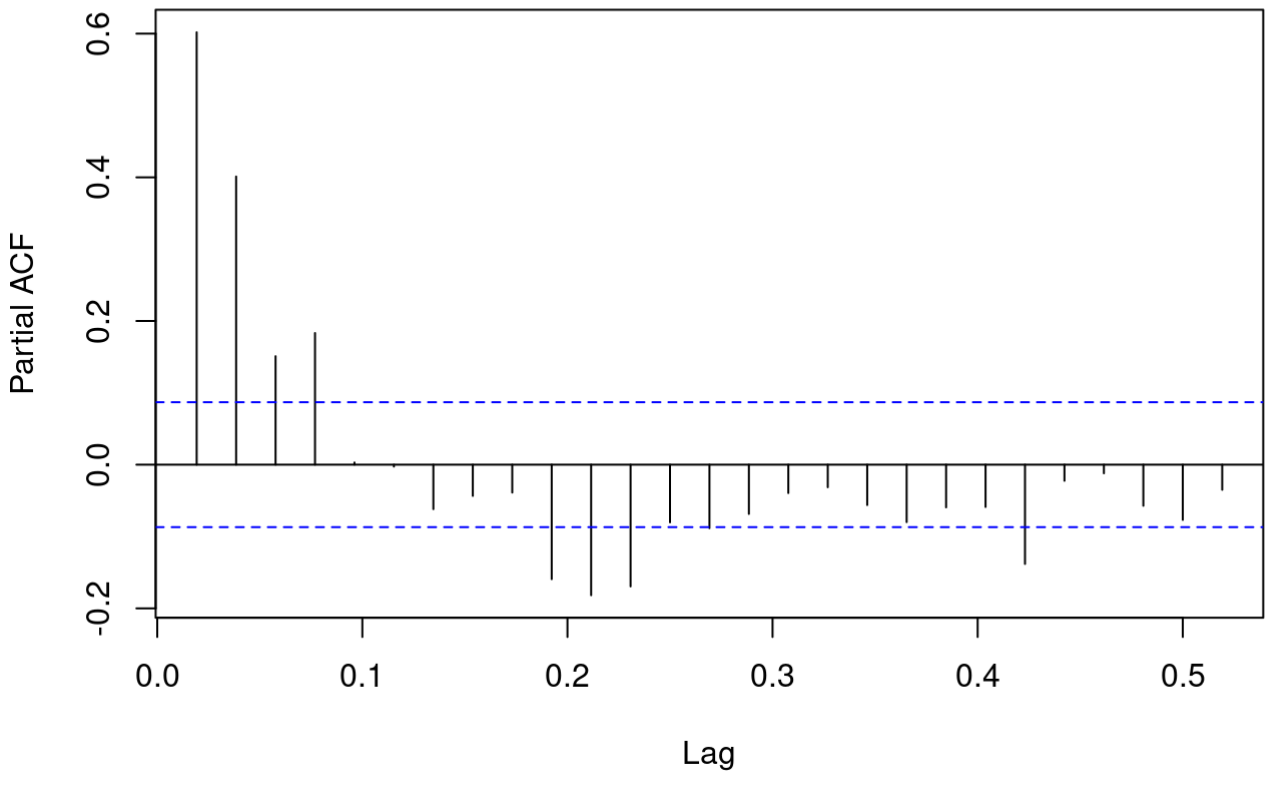
```
acf(temp, main = "Figure 9.1: ACF for temp")
```

Figure 9.1: ACF for temp




```
pacf(temp, main = "Figure 9.2: ACF for temp")
```

Figure 9.2: ACF for temp



It can be seen from the ACF plot above that the series has slowly decaying pattern while the PACF shows some autocorrelation is present in the series as first two lags are significant.

Dicker-Fuller Unit-Root test (ADF) for temp series:

```
#Performing ADF test on temp series

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  temp
## Dickey-Fuller = -8.2554, Lag order = 22, p-value = 0.01
## alternative hypothesis: stationary
```

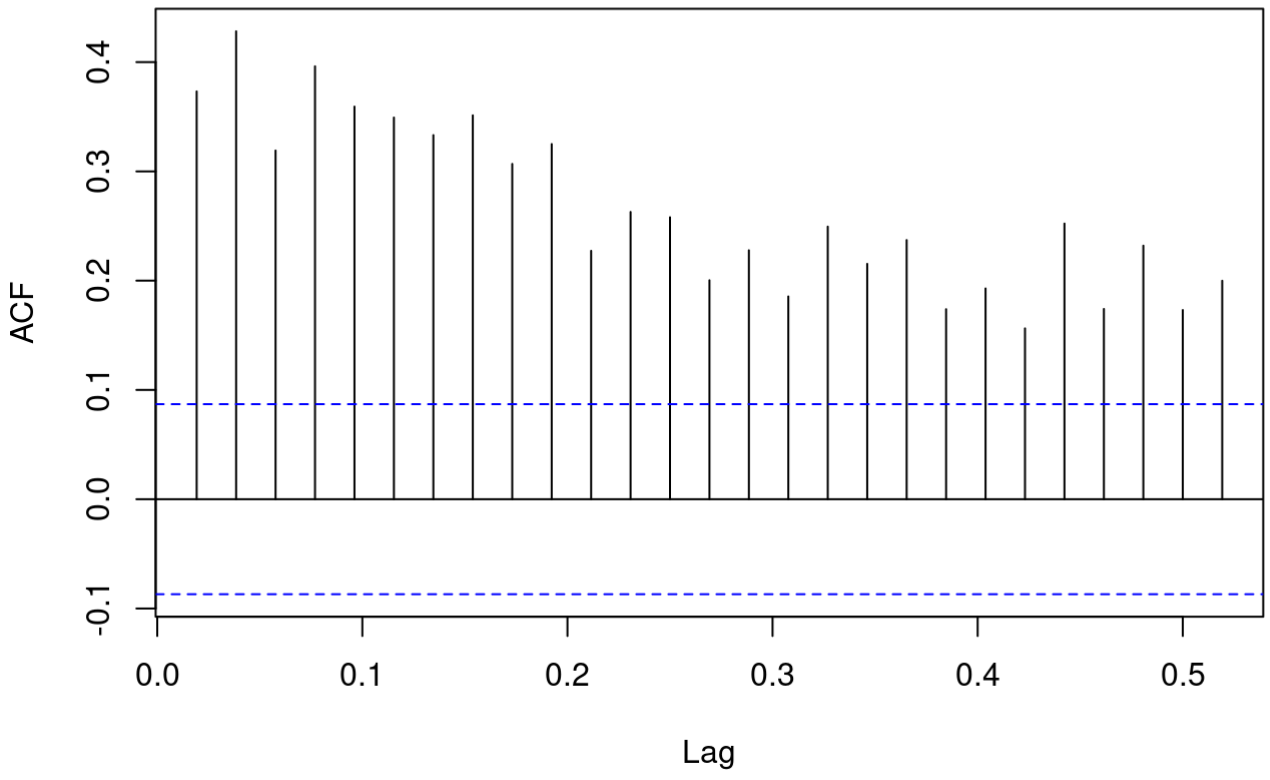
As per the results obtained from the ADF test of 22 lag order, the p-value is less than 5% level of significance which indicates the temperature (temp) series is stationary.

Checking stationarity for chem1 series:

Plotting ACF for chem1 series:

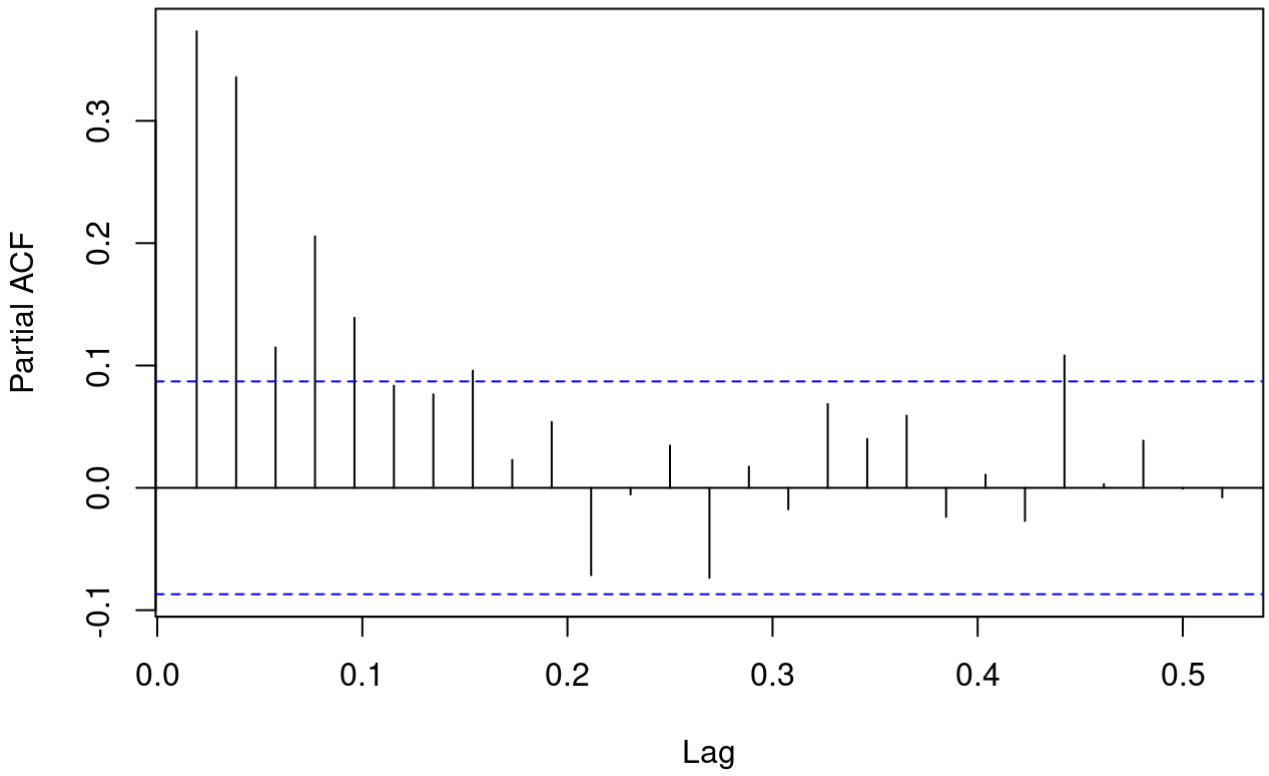
```
acf(chem1, main = "Figure 10.1: ACF for chem1")
```

Figure 10.1: ACF for chem1




```
pacf(chem1, main = "Figure 10.2: ACF for chem1")
```

Figure 10.2: ACF for chem1



It can be seen from the ACF plot above that the series has some kind of decreasing trend. The PACF plot reveals that first two lags are significant which suggests presence of seasonality in the series.

Dicker-Fuller Unit-Root test (ADF) for chem1 series:

```
#Performing ADF test on chem1 series

adf.test(chem1, k=ar(chem1)$order)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  chem1
## Dickey-Fuller = -5.3362, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

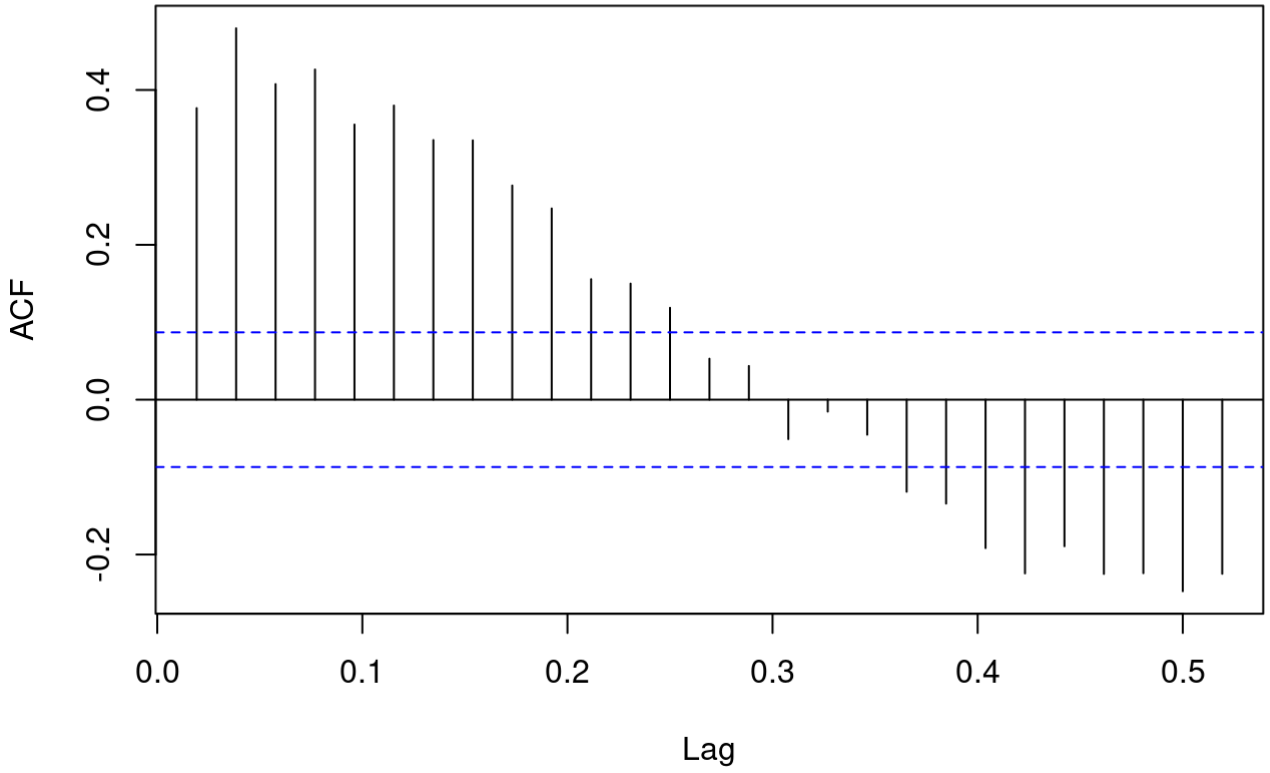
As per the results obtained from the ADF test of 8 lag order, the p-value is less than 5% significance level which indicates the chemical1 (chem1) series is stationary.

Checking stationarity for chem2 series:

Plotting ACF for chem2 series:

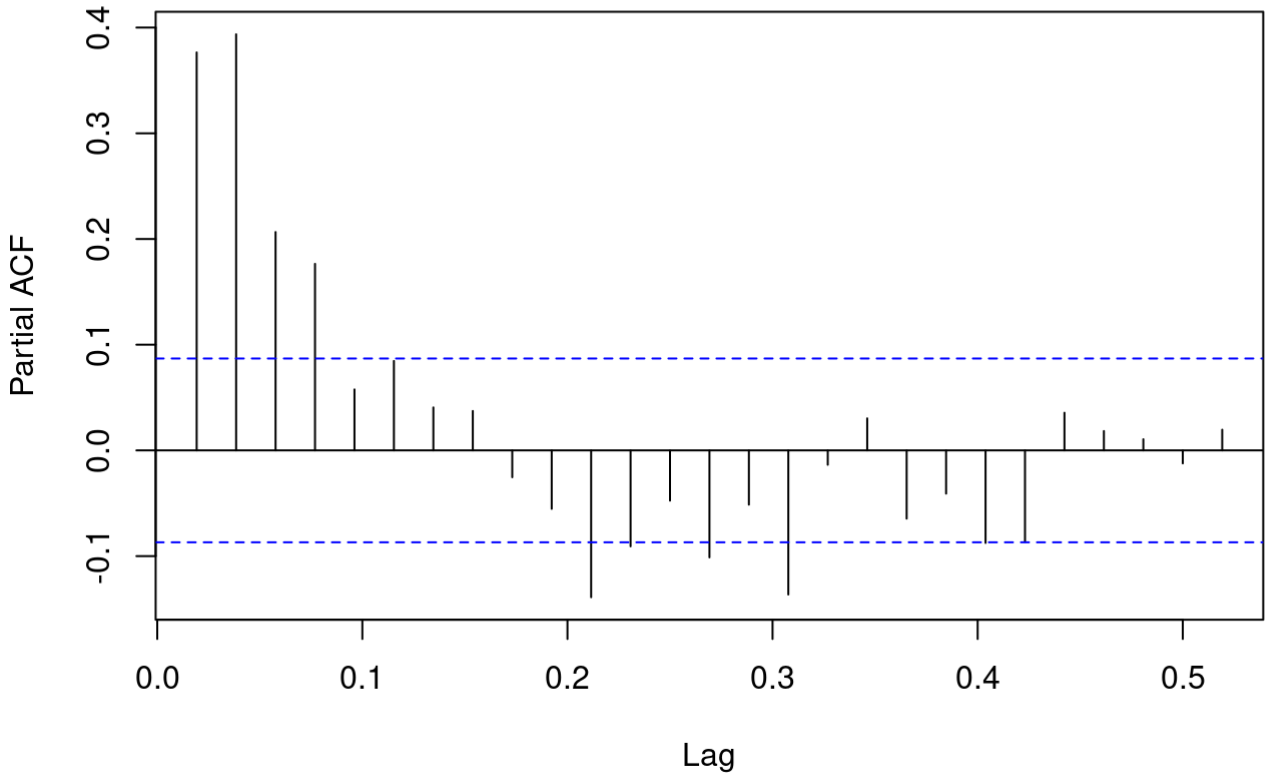
```
acf(chem2, main = "Figure 11.1: ACF for chem2")
```

Figure 11.1: ACF for chem2




```
pacf(chem2, main = "Figure 11.2: ACF for chem2")
```

Figure 11.2: ACF for chem2



It can be seen from the ACF plot above that the series has decreasing trend. The PACF on the other hand, reveals that the series has serial correlation as many lags are significant.

Dicker-Fuller Unit-Root test (ADF) for chem2 series:

```
#Performing ADF test on chem2 series

adf.test(chem2, k=ar(chem2)$order)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  chem2
## Dickey-Fuller = -6.5194, Lag order = 16, p-value = 0.01
## alternative hypothesis: stationary
```

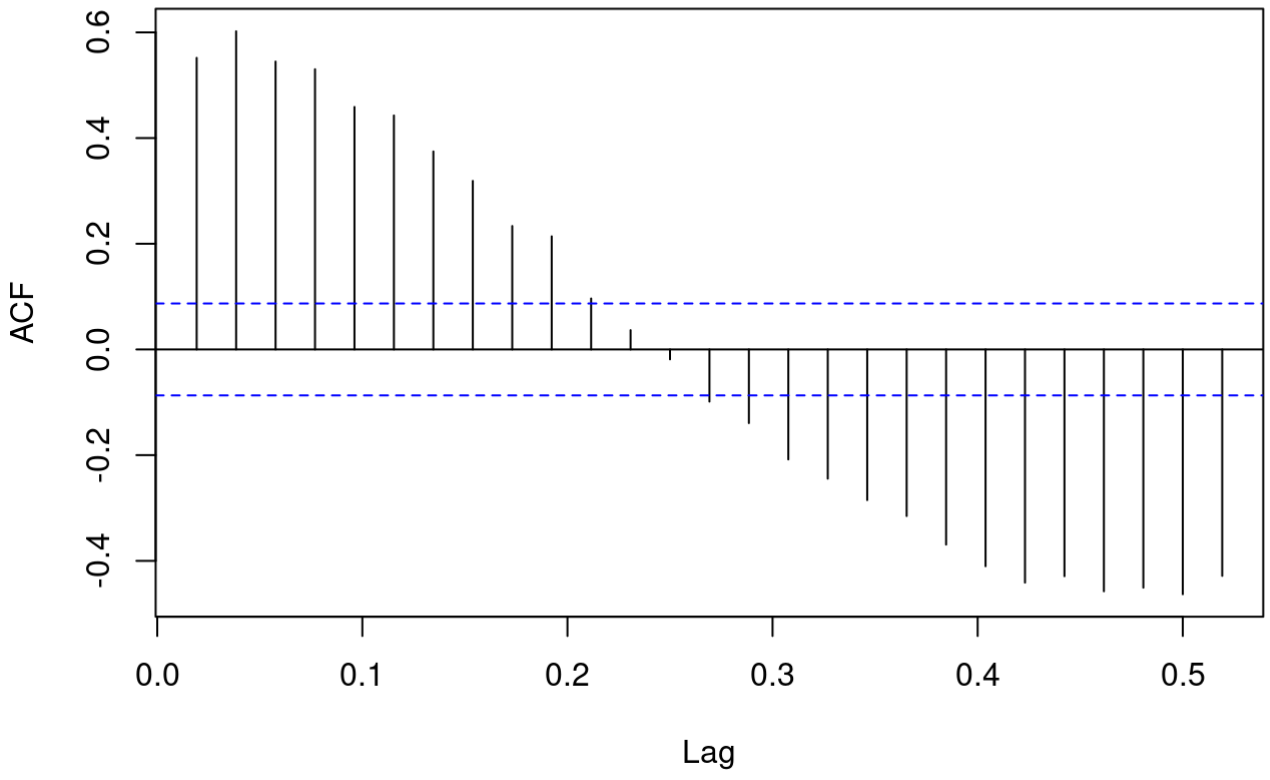
As per the results obtained from the ADF test of 16 lag order, the p-value is less than 5% significance level which indicates the chemical2 (chem2) series is stationary.

Checking stationarity for particle size series:

Plotting ACF for part series:

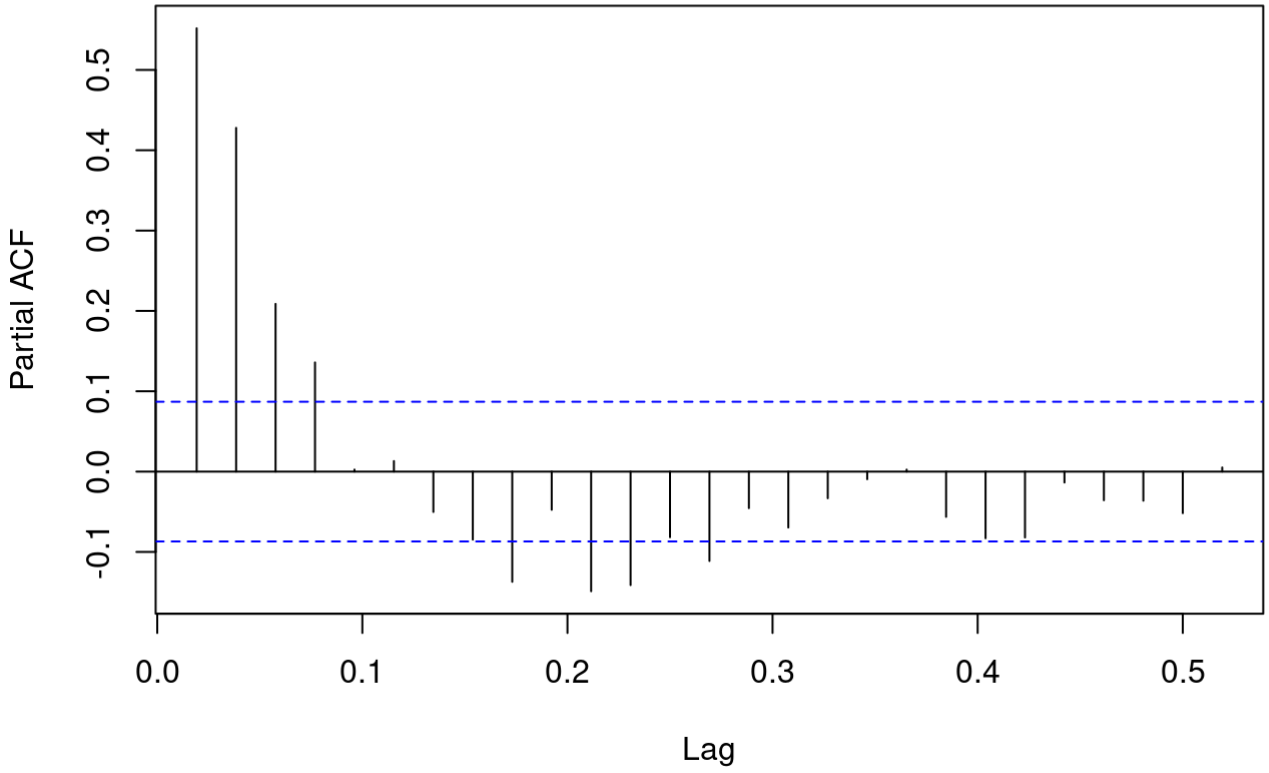
```
acf(part, main = "Figure 12.1: ACF for part")
```

Figure 12.1: ACF for part




```
pacf(part, main = "Figure 12.2: ACF for part")
```

Figure 12.2: ACF for part



It can be seen from the ACF plot above that the series has some decaying pattern. The PACF reveals that the series has serial correlation as many lags are significant.

Dicker-Fuller Unit-Root test (ADF) for part series:

```
#Performing ADF test on part series

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  part
## Dickey-Fuller = -7.2956, Lag order = 14, p-value = 0.01
## alternative hypothesis: stationary
```

As per the results obtained from the ADF test of 14 lag order, the p-value is less than 0.05 significance level which indicates the particle size (part) series is stationary.

Intepretation on Stationarity:

As observed from the ACF plots and ADF tests, all the five series are stationary. Thus, let's move to next steps of performing decomposition analysis on each of the five series.

Decomposition Analysis

Decomposition is the process of decomposing the series into different components and understanding the individual effect of that component on the series. Since all the four series are stationary, decomposition can be performed next on each series. There are 3 components of any time series-

1. Trend
2. Seasonality
3. Remainder

In order to understand the impact of each of these on the series, the four series will be decomposed one by one using STL decomposition only as **X12 decomposition cannot process weekly average series**.

Loess STL Decomposition:

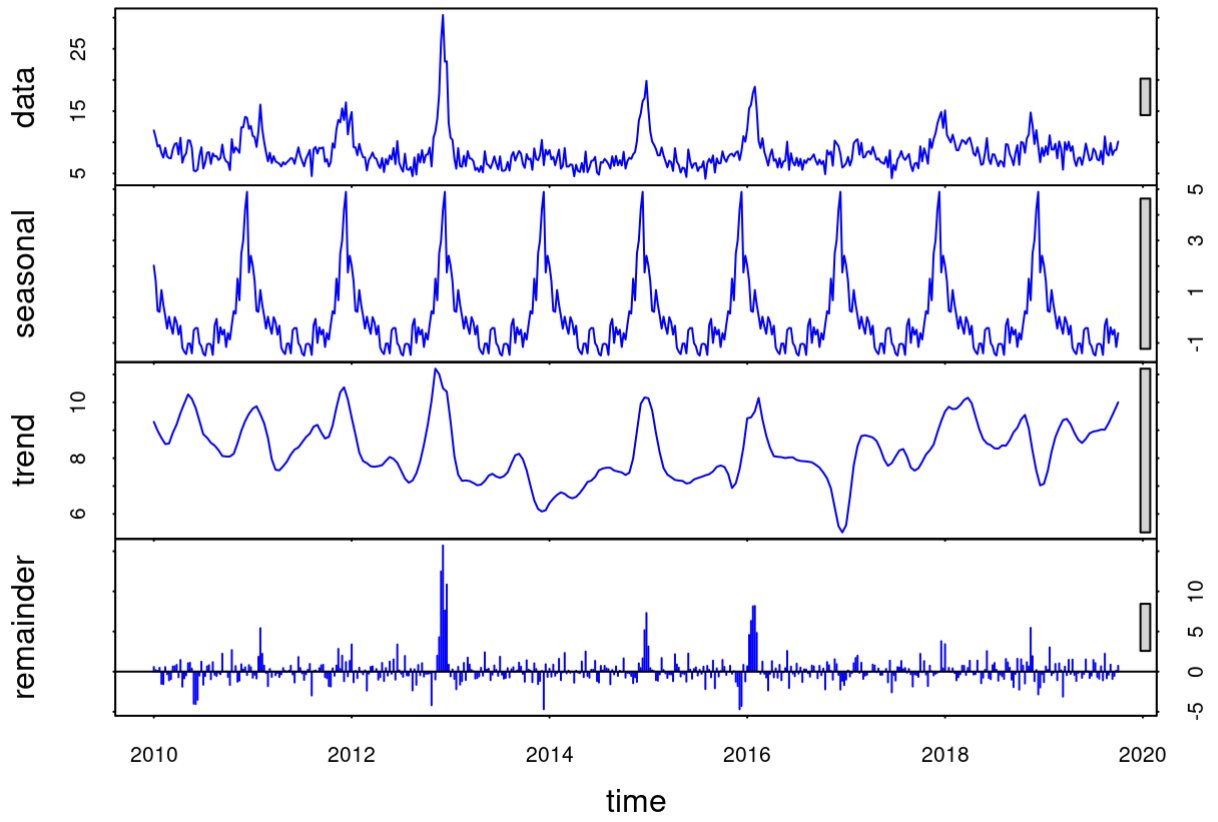
The STL decomposition is superior than traditional decomposition since it can handle any sort of seasonality and is resistant to outliers.

STL Decomposition of mor series:

Performing STL decomposition on mor series-

```
morstl = stl(mor, t.window = 15, s.window = "periodic", robust = TRUE)
plot(morstl, main = "Figure 13: STL decomposition of mor series", col="blue")
```

Figure 13: STL decomposition of mor series



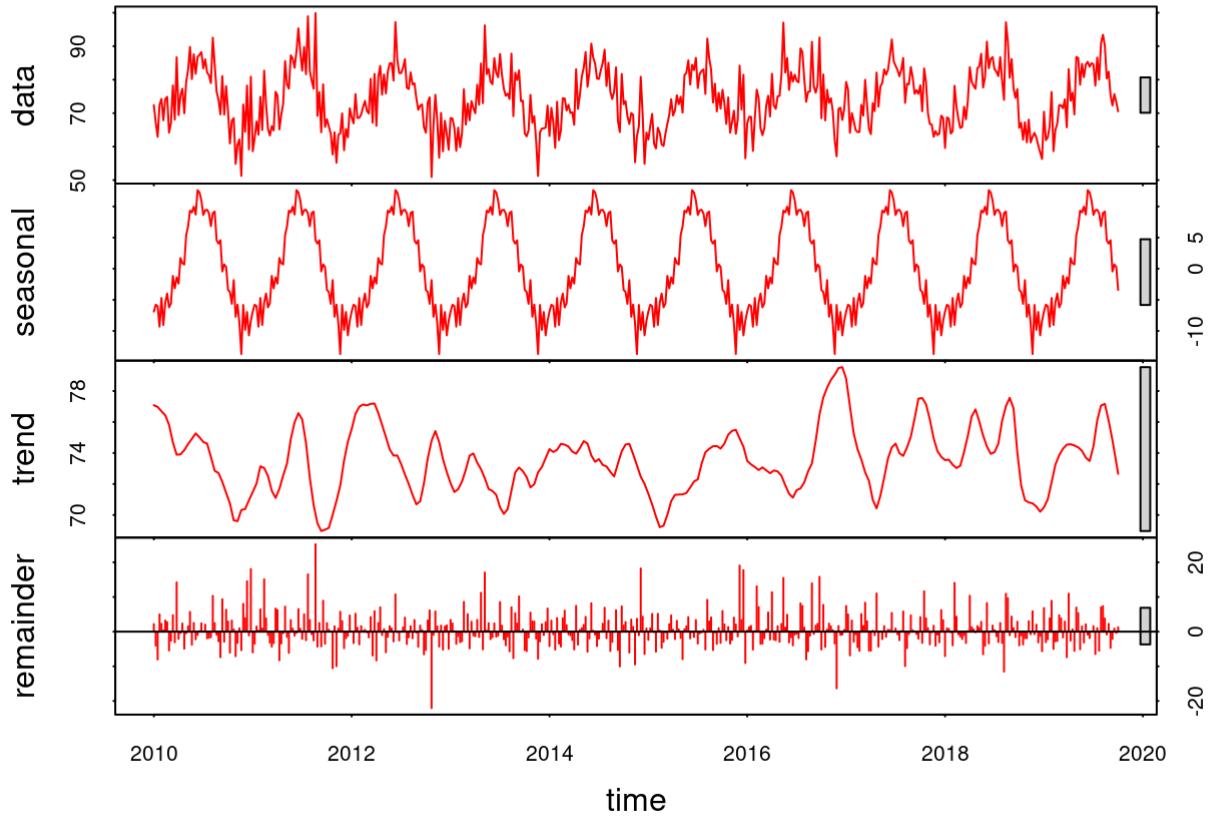
From the STL decomposition obtained for the mor series, the trend pattern is similar to the original series and does not exist. The ACF and plot for the series showed signs of seasonality, thus, the seasonal pattern observed above can't be ignored. Some moderate interventions can be observed in the remainder part of the decomposition between 2012 & 2014 and in the weeks of 2016.

STL Decomposition of temp series:

Performing STL decomposition on temp series-

```
tempstl = stl(temp, t.window = 15, s.window = "periodic", robust = TRUE)
plot(tempstl, main = "Figure 14: STL decomposition of temp series",col="red")
```

Figure 14: STL decomposition of temp series



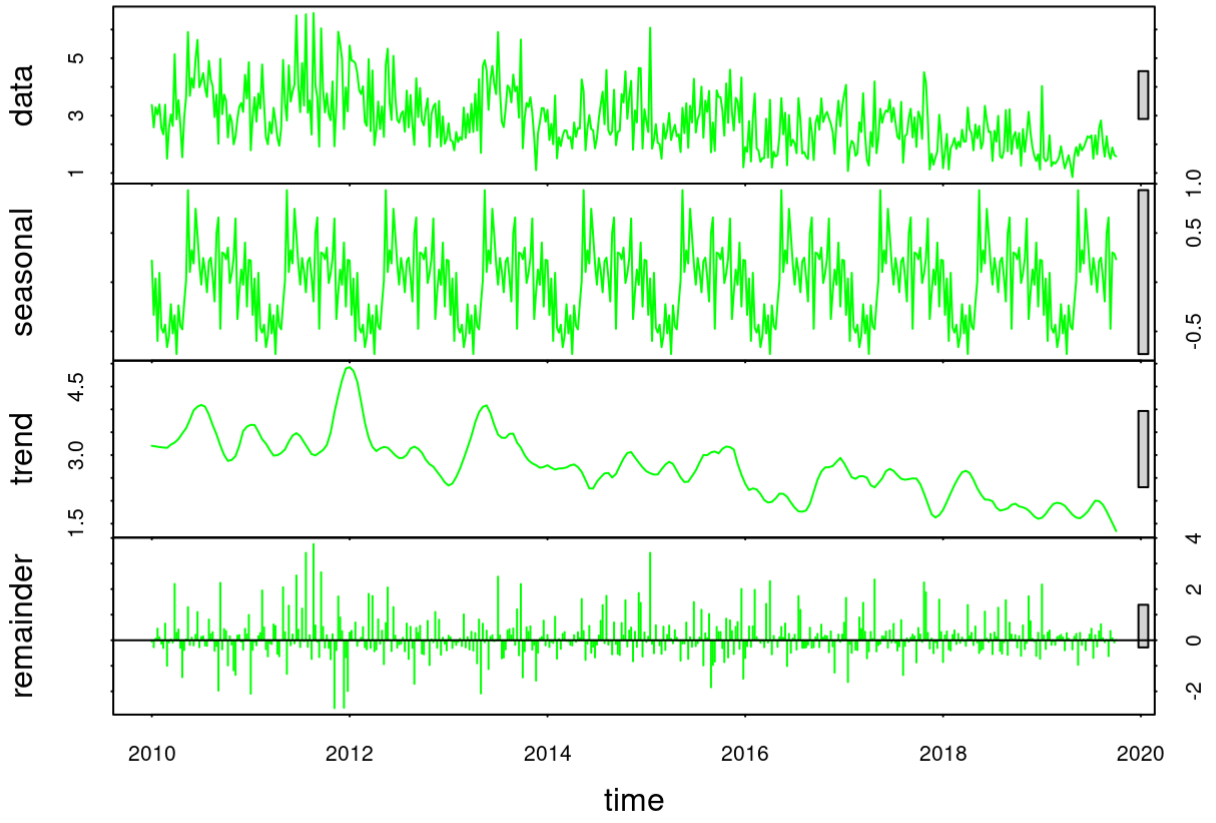
From the STL decomposition obtained for the temp series, the trend pattern is similar to the original series and does not seem to exist. The ACF plot for the series showed obvious seasonality, thus, the seasonal pattern observed above can't be ignored.Some moderate interventions can be observed in the remainder part of the decomposition between 2012 & 2013 and between 2016 & 2017.

STL Decomposition of chem1 series:

Performing STL decomposition on chem1 series-

```
chem1stl = stl(chem1, t.window = 15, s.window = "periodic", robust = TRUE)
plot(chem1stl, main = "Figure 15: STL decomposition of chem1 series",col="green")
```

Figure 15: STL decomposition of chem1 series



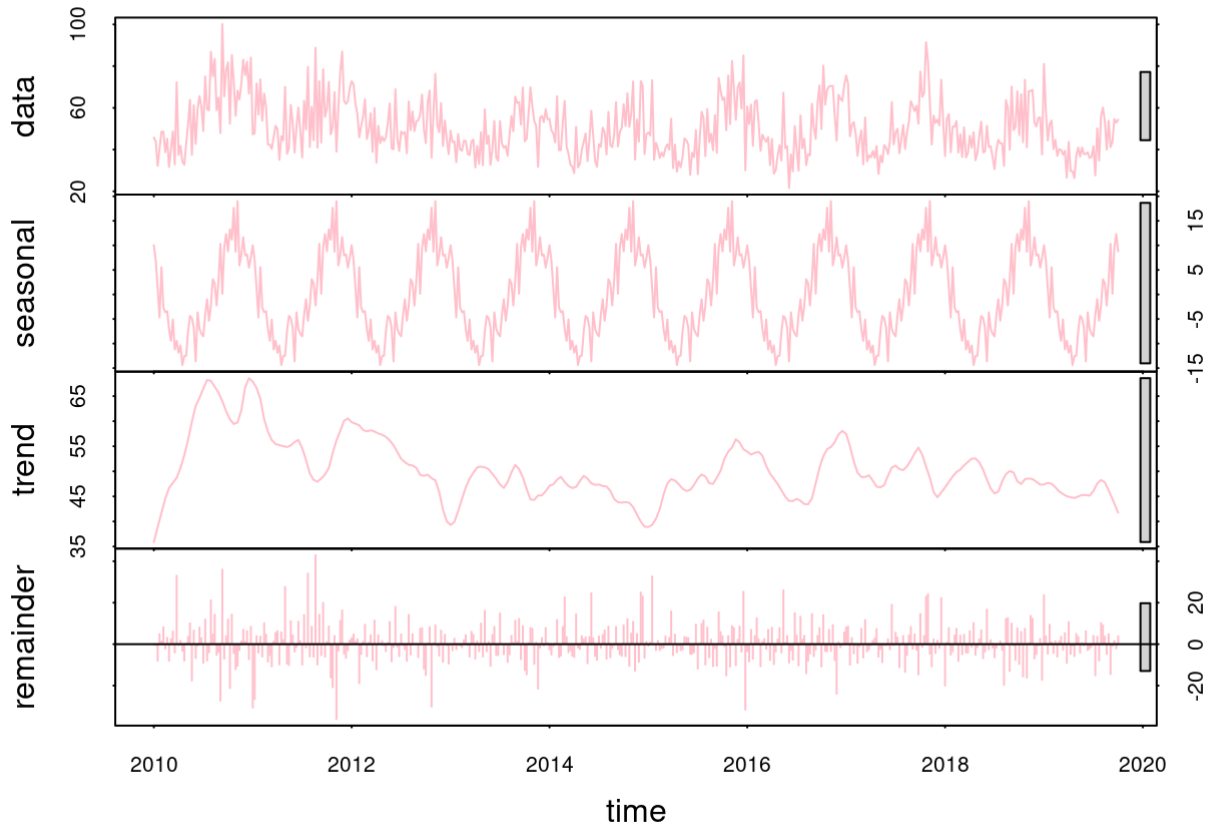
From the STL decomposition obtained for the chem1 series, the trend pattern is similar to the original series and a decreasing pattern is seen. The ACF plot for the series showed obvious seasonality, thus, the seasonal pattern is also observed in the decomposition. Several large interventions can be observed in the remainder part of the decomposition.

STL Decomposition of chem2 series:

Performing STL decomposition on chem2 series-

```
chem2stl = stl(chem2, t.window = 15, s.window = "periodic", robust = TRUE)
plot(chem2stl, main = "Figure 16: STL decomposition of chem2 series",col="pink")
```

Figure 16: STL decomposition of chem2 series



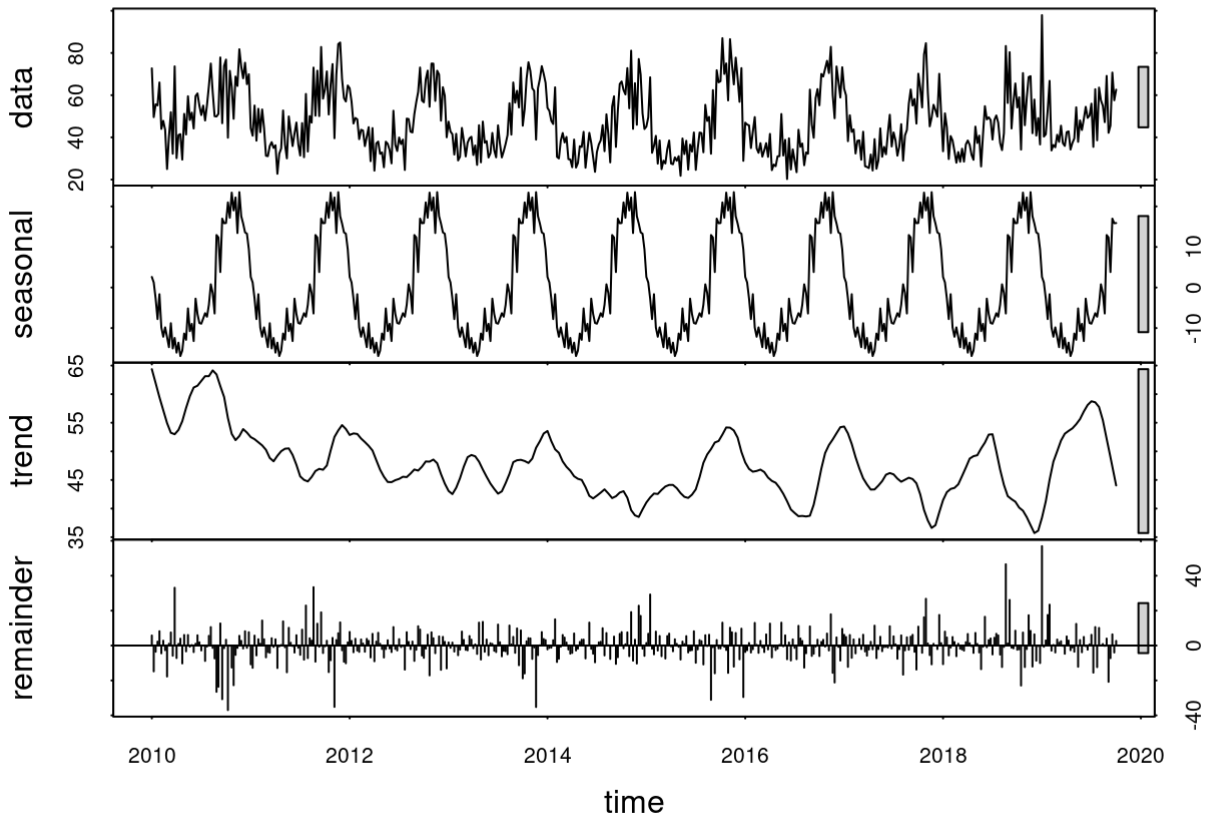
From the STL decomposition obtained for the chem2 series, the trend pattern is similar to the original series and a gradual decreasing pattern is observed. The ACF plot for the series showed obvious seasonality, thus, the seasonal pattern is also observed in the decomposition. Several large interventions can be observed in the remainder part of the decomposition.

STL Decomposition of part series:

Performing STL decomposition on part series-

```
partstl = stl(part, t.window = 15, s.window = "periodic", robust = TRUE)
plot(partstl, main = "Figure 17: STL decomposition of part series",col="black")
```

Figure 17: STL decomposition of part series



From the STL decomposition obtained for the part series, the trend pattern is similar to the original series and no trend is observed. The ACF plot for the series showed obvious seasonality, thus, the seasonal pattern is also observed in the decomposition. Several large interventions can be observed in the remainder part of the decomposition after 2018.

Modelling using Distributed Lag Models

Distributed lag model (DLM) is a type of regression model in which each lag of the predictor variable tries to explain overall variation and correlation structure in the dependent variable. There are 4 different types of DLMs that will be used to fit all the five series-

1. Finite DLM
2. Polynomial DLM
3. Koyck transformation DLM
4. Autoregressive DLM

Finite DLM:

For finding the best model using finite DLM, we will consider all the four series (except mortality) as predictor one by one and fit finite DLM based on appropriate lag length.

Temperature vs Mortality:

Computing lag length for the model with temp as predictor and mor as dependent variable by using finiteDLMauto function that will perform a comparison based on AIC, BIC and MASE values for lags ranging from 1 to 10 and choosing model having lowest of AIC,BIC and MASE values-

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

finiteDLMauto( x=as.vector(temp+chem1+chem2+part), y= as.vector(mor), 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 1.15477 2383.868 2438.606 1.11891 -1.04522  0.18059         0
## 9      9 1.15985 2398.491 2449.042 1.03841 -0.25224  0.16109         0
## 8      8 1.17944 2413.980 2460.340 1.08726 -4.15778  0.13969         0
## 7      7 1.19884 2428.873 2471.039 1.11368  0.58010  0.11879         0
## 6      6 1.22335 2448.068 2486.036 1.15381 -2.10328  0.08973         0
## 5      5 1.23543 2460.486 2494.251 1.16747  0.49698  0.07244         0
## 4      4 1.25719 2475.906 2505.464 1.17151  0.79142  0.04908         0
## 3      3 1.27285 2487.168 2512.515 1.21066  2.00062  0.03353         0
## 2      2 1.28116 2494.829 2515.962 1.23695  1.74173  0.02464         0
## 1      1 1.29035 2501.931 2518.845 1.25764  0.38566  0.01790         0
```

```
finiteDLMauto( x=as.vector(temp+chem1+chem2+part), y= as.vector(mor), 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 1.15477 2383.868 2438.606 1.11891 -1.04522  0.18059         0
## 9      9 1.15985 2398.491 2449.042 1.03841 -0.25224  0.16109         0
## 8      8 1.17944 2413.980 2460.340 1.08726 -4.15778  0.13969         0
## 7      7 1.19884 2428.873 2471.039 1.11368  0.58010  0.11879         0
## 6      6 1.22335 2448.068 2486.036 1.15381 -2.10328  0.08973         0
## 5      5 1.23543 2460.486 2494.251 1.16747  0.49698  0.07244         0
## 4      4 1.25719 2475.906 2505.464 1.17151  0.79142  0.04908         0
## 3      3 1.27285 2487.168 2512.515 1.21066  2.00062  0.03353         0
## 2      2 1.28116 2494.829 2515.962 1.23695  1.74173  0.02464         0
## 1      1 1.29035 2501.931 2518.845 1.25764  0.38566  0.01790         0
```

```
finiteDLMauto( x=as.vector(temp+chem1+chem2+part), y= as.vector(mor), 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 1.15477 2383.868 2438.606 1.11891 -1.04522  0.18059         0
## 9      9 1.15985 2398.491 2449.042 1.03841 -0.25224  0.16109         0
## 8      8 1.17944 2413.980 2460.340 1.08726 -4.15778  0.13969         0
## 7      7 1.19884 2428.873 2471.039 1.11368  0.58010  0.11879         0
## 6      6 1.22335 2448.068 2486.036 1.15381 -2.10328  0.08973         0
## 5      5 1.23543 2460.486 2494.251 1.16747  0.49698  0.07244         0
## 4      4 1.25719 2475.906 2505.464 1.17151  0.79142  0.04908         0
## 3      3 1.27285 2487.168 2512.515 1.21066  2.00062  0.03353         0
## 2      2 1.28116 2494.829 2515.962 1.23695  1.74173  0.02464         0
## 1      1 1.29035 2501.931 2518.845 1.25764  0.38566  0.01790         0
```

As per the above output, lag-length should be 10 based on the MASE error type. Thus, for now fitting finite DLM model with temp as predictor-

```
model.temp = dlm(x=as.vector(temp), y=as.vector(mor), q=10)
```

Chem1 vs Mor:

Fitting finite DLM model with chemical1 as predictor-

```
model.chem1 = dlm(x=as.vector(chem1), y=as.vector(mor), q=10)
```

Chem2 vs Mor:

Fitting finite DLM model with chemical2 as predictor-

```
model.chem2 = dlm(x=as.vector(chem2), y=as.vector(mor), q=10)
```

Part vs Mor:

Fitting finite DLM model with particle size as predictor-

```
model.part = dlm(x=as.vector(part), y=as.vector(mor), q=10)
```

Temp and chem 1 vs Mor:

Fitting finite DLM model with temp and chem1 as predictor-

```
model.tempchem1 = dlm(x=as.vector(temp+chem1), y=as.vector(mor), q=10)
```

Temp and chem 2 vs Mor:

Fitting finite DLM model with temp and chem2 as predictor-

```
model.tempchem2 = dlm(x=as.vector(temp+chem2), y=as.vector(mor), q=10)
```

Chem1 and chem 2 vs Mor:

Fitting finite DLM model with chem1 and chem2 as predictor-

```
model.chem1chem2 = dlm(x=as.vector(chem1+chem1), y=as.vector(mor), q=10)
```

Temp and part vs Mor:

Fitting finite DLM model with temp and part as predictor-

```
model.temppart = dlm(x=as.vector(temp+part), y=as.vector(mor), q=10)
```

Chem1 and part vs Mor:

Fitting finite DLM model with chem1 and part as predictor-

```
model.chem1part = dlm(x=as.vector(chem1+part), y=as.vector(mor), q=10)
```

Chem2 and part vs Mor:

Fitting finite DLM model with chem2 and part as predictor-

```
model.chem2part = dlm(x=as.vector(chem2+part), y=as.vector(mor), q=10)
```

Temp, chem1, chem2 vs Mor:

Fitting finite DLM model with temp, chem1 and chem2 as predictor-

```
model.tempchem1chem2 = dlm(x=as.vector(temp+chem1+chem2), y=as.vector(mor), q=10)
```

Temp, chem1, part vs Mor:

Fitting finite DLM model with temp, chem1 and part as predictor-

```
model.tempchem1part = dlm(x=as.vector(temp+chem1+part), y=as.vector(mor), q=10)
```

Temp, chem2, part vs Mor:

Fitting finite DLM model with temp, chem2 and part as predictor-

```
model.tempchem2part = dlm(x=as.vector(temp+chem2+part), y=as.vector(mor), q=10)
```

Chem1, chem2, part vs Mor:

Fitting finite DLM model with chem1, chem2 and part as predictor-

```
model.chem1chem2part = dlm(x=as.vector(chem1+chem2+part), y=as.vector(mor), q=10)
```

Temp, Chem1, chem2, part vs Mor:

Fitting finite DLM model with temp, chem1, chem2 and part as predictor-

```
model.tempchem1chem2part = dlm(x=as.vector(temp+chem1+chem2+part), y=as.vector(mor), q=10)
```

Comparing all the finite DLMs fitted based on AIC,BIC and MASE values:

We will use the user-defined function sort.score() to compare all the fitted finite DLMS using AIC, BIC and MASE values:

```
# Defining function 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")')
  }
}
```

Comparison based on AIC:

```
# Sorting based on AIC

sort.score(AIC(model.temp$model,model.chem1$model,model.chem2$model,model.part$model,model.te
mpchem1$model,model.tempchem2$model,model.temppart$model,model.chem1chem2$model,model.chem1pa
rt$model,model.chem2part$model,model.tempchem1chem2$model,model.tempchem1part$model,model.tem
pchem2part$model,model.chem1chem2part$model,model.tempchem1chem2part$model), score = "aic")
```

```
##                                df      AIC
## model.part$model               13 2318.164
## model.chem1part$model          13 2323.363
## model.chem2part$model          13 2341.546
## model.chem1chem2part$model     13 2344.918
## model.temp$model               13 2380.059
## model.tempchem2part$model      13 2380.687
## model.tempchem1chem2part$model 13 2383.868
## model.tempchem1$model          13 2384.980
## model.chem2$model              13 2393.253
## model.temppart$model           13 2397.244
## model.tempchem1part$model      13 2401.098
## model.tempchem2$model          13 2455.059
## model.tempchem1chem2$model     13 2456.223
## model.chem1$model              13 2483.558
## model.chem1chem2$model         13 2483.558
```

From the above results we can clearly say that model.part is the best model based on AIC while model with all the predictors is the second best model.

Comparison based on BIC:

```
# Sorting based on BIC

sort.score(BIC(model.temp$model,model.chem1$model,model.chem2$model,model.part$model,model.te
mpchem1$model,model.tempchem2$model,model.temppart$model,model.chem1chem2$model,model.chem1pa
rt$model,model.chem2part$model,model.tempchem1chem2$model,model.tempchem1part$model,model.tem
pchem2part$model,model.chem1chem2part$model,model.tempchem1chem2part$model), score = "bic")
```

```
##                                df      BIC
## model.part$model               13 2372.902
## model.chem1part$model          13 2378.101
## model.chem2part$model          13 2396.284
## model.chem1chem2part$model     13 2399.656
## model.temp$model               13 2434.796
## model.tempchem2part$model      13 2435.425
## model.tempchem1chem2part$model 13 2438.606
## model.tempchem1$model          13 2439.717
## model.chem2$model              13 2447.991
## model.temppart$model           13 2451.982
## model.tempchem1part$model      13 2455.836
## model.tempchem2$model          13 2509.797
## model.tempchem1chem2$model     13 2510.961
## model.chem1$model              13 2538.296
## model.chem1chem2$model         13 2538.296
```

From the above results we can clearly say that model.part is the best model based on BIC while model with only particle size as predictor is the second best model.

Comparison based on MASE:

```
# Sorting based on MASE

sort.score(MASE(model.temp$model,model.chem1$model,model.chem2$model,model.part$model,model.t
empchem1$model,model.tempchem2$model,model.temppart$model,model.chem1chem2$model,model.chem1p
art$model,model.chem2part$model,model.tempchem1chem2$model,model.tempchem1part$model,model.te
mpchem2part$model,model.chem1chem2part$model,model.tempchem1chem2part$model), score = "mase")
```

```
##                                   n     MASE
## model.part$model                498 1.087711
## model.chem1part$model           498 1.097456
## model.chem2part$model           498 1.101983
## model.chem1chem2part$model      498 1.107963
## model.tempchem2part$model       498 1.148070
## model.chem2$model               498 1.150336
## model.tempchem1chem2part$model 498 1.154767
## model.temp$model                498 1.169444
## model.tempchem1$model           498 1.170722
## model.temppart$model            498 1.173587
## model.tempchem1part$model       498 1.180025
## model.tempchem2$model           498 1.235505
## model.tempchem1chem2$model      498 1.238762
## model.chem1$model               498 1.294917
## model.chem1chem2$model          498 1.294917
```

From the above results we can clearly say that model.part is the best model based on MASE while model with all the variables as predictor is the second best model.

Interpretation:

Thus, it is clearly seen that model.part with particle size as the only predictor is the best model based on AIC, BIC and MASE values. Hence, this model will be further analysed using diagnostic checking.

Analysing Part vs Mortality finite DLM:

Using summary and checkresiduals function, analysing the model.part finite DLM-

```
model.finite = dlm(x=as.vector(part), y=as.vector(mor), q=10)
summary(model.finite)
```

```
##
## Call:
## lm(formula = model.formula, data = design)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7002 -1.3059 -0.1555  0.9361 20.0489
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.0116427  0.4927144   4.083  5.2e-05 ***
## x.t          0.0105446  0.0101416   1.040  0.29898
## x.1          0.0002472  0.0102817   0.024  0.98083
## x.2          0.0012723  0.0106010   0.120  0.90452
## x.3         -0.0015232  0.0107943  -0.141  0.88784
## x.4          0.0032978  0.0109400   0.301  0.76320
## x.5          0.0135526  0.0109482   1.238  0.21636
## x.6          0.0075328  0.0109549   0.688  0.49202
## x.7          0.0211822  0.0108377   1.955  0.05121 .
## x.8          0.0155020  0.0106744   1.452  0.14707
## x.9          0.0294183  0.0102919   2.858  0.00444 **
## x.10         0.0337457  0.0101393   3.328  0.00094 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.446 on 486 degrees of freedom
## Multiple R-squared:  0.2978, Adjusted R-squared:  0.2819
## F-statistic: 18.73 on 11 and 486 DF,  p-value: < 2.2e-16
##
## AIC and BIC values for the model:
##        AIC      BIC
## 1 2318.164 2372.902
```

From the above summary, we can generate following insights about the model.temp-

1. The model model.part is a poor fitted model. All the lags are insignificant except x.9 and x.10.
2. The value of adjusted R2 is 28.2%. This means that model.finite is able to explain only 28.2% variation in the dependent variable (mortality).
3. The residuals have maximum value of 20.0489 and minimum value of -5.7002 with residual standard error (RSE) as 2.446.
4. The AIC & BIC values of the model are 2318.164 & 2372.902 respectively.
5. As per the F-test of the overall significance of model, the model.part is a significant model at 5% level of significance despite having many insignificant number of lags.

Calculating VIF for model.finite finite DLM:

Calculating VIF for the model.finite DLM to check for presence of multicollinearity in the model-

```
vif(model.finite$model)
```

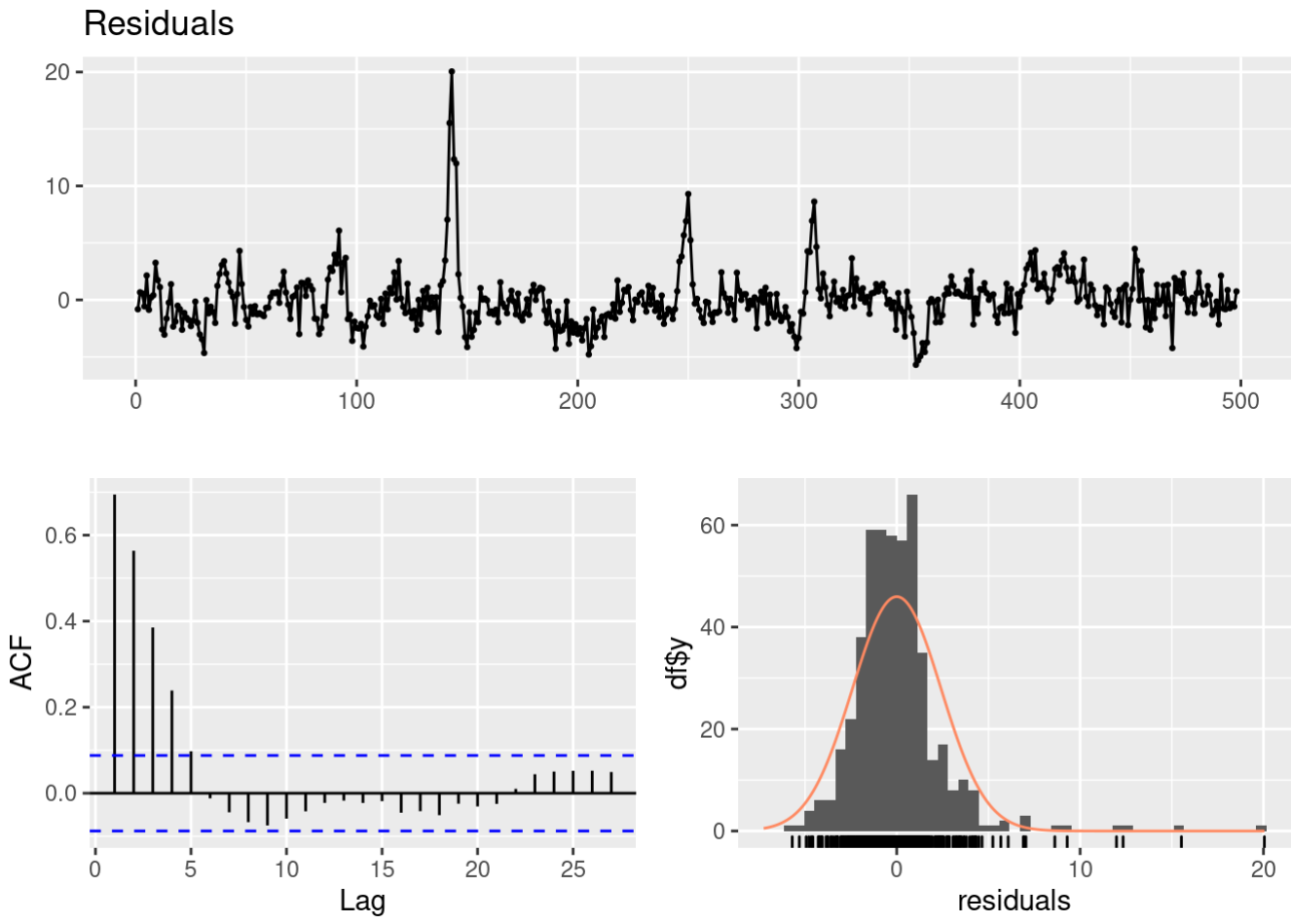
```
##      x.t      x.1      x.2      x.3      x.4      x.5      x.6      x.7
## 1.967498 2.018952 2.153637 2.222411 2.282775 2.285863 2.286108 2.238692
##      x.8      x.9     x.10
## 2.171987 2.017317 1.964519
```

As seen above, the vif <10 for all the lags in the model. Hence, there is no issue of multicollinearity in the model.finite model.

Residual Analysis for model.finite DLM:

The residual analysis will be performed on the model.finite using checkresiduals function. In addition to generating residual plots, the Breusch-Godfrey test will be performed. In this test, H0 indicates there is no serial correlation in the residuals while Ha indicates there is serial correlation in the residuals.

```
#Residual analysis of model.finite model


checkresiduals(model.finite$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 15
##
## data:  Residuals
## LM test = 255.73, df = 15, p-value < 2.2e-16
```

From the Breusch-Godfrey test conducted, the p-value is less than 5% level of significance which indicates that serial correlation exists among the residuals. The results obtained from the Breusch-Godfrey test are supported by the residuals plots generated as it is seen that residuals seem to follow a trend and are not

distributed randomly. From the ACF plot, it is clearly seen that serial correlation & seasonality exists in the residuals. The histogram shows that residuals are not following normal distribution.

Overall, we can conclude that the finite DLM model.finite is a poor fit significant finite DLM as it explains only 28.2% of the variation in mortality and have insignificant components.

Polynomial DLM:

As the finite DLM model was a poor fit , we will try to fit a polynomial model.Before, we fit the polynomial model, using finiteDLMauto, we will find the best lag length based on fitted models MASE values-

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

finiteDLMauto(x = as.vector(temp+chem1+chem2+part), y = as.vector(mor), q.min = 1, q.max = 10
, 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 1.16004 2370.609 2391.662 1.09458 0.59736  0.18941         0
## 9   9 - 2 1.16575 2386.791 2407.854 1.03468 1.05329  0.16913         0
## 8   8 - 2 1.18770 2404.256 2425.329 1.11569 0.41622  0.14622         0
## 7   7 - 2 1.20381 2420.283 2441.366 1.11020 1.63230  0.12520         0
## 6   6 - 2 1.22530 2441.679 2462.772 1.13018 1.40049  0.09414         0
## 5   5 - 2 1.23389 2455.414 2476.517 1.15497 0.78596  0.07631         0
## 4   4 - 2 1.25679 2472.301 2493.414 1.16849 0.51433  0.05214         0
## 3   3 - 2 1.27490 2485.654 2506.777 1.21643 1.29587  0.03453         0
## 2   2 - 2 1.28116 2494.829 2515.962 1.23695 1.74173  0.02464         0
## 1   1 - 2 1.29035 2501.931 2518.845 1.25764 0.38566  0.01790         0
```

```
finiteDLMauto(x = as.vector(temp+chem1+chem2+part), y = as.vector(mor), q.min = 1, q.max = 10
, 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 1.16004 2370.609 2391.662 1.09458 0.59736  0.18941         0
## 9   9 - 2 1.16575 2386.791 2407.854 1.03468 1.05329  0.16913         0
## 8   8 - 2 1.18770 2404.256 2425.329 1.11569 0.41622  0.14622         0
## 7   7 - 2 1.20381 2420.283 2441.366 1.11020 1.63230  0.12520         0
## 6   6 - 2 1.22530 2441.679 2462.772 1.13018 1.40049  0.09414         0
## 5   5 - 2 1.23389 2455.414 2476.517 1.15497 0.78596  0.07631         0
## 4   4 - 2 1.25679 2472.301 2493.414 1.16849 0.51433  0.05214         0
## 3   3 - 2 1.27490 2485.654 2506.777 1.21643 1.29587  0.03453         0
## 2   2 - 2 1.28116 2494.829 2515.962 1.23695 1.74173  0.02464         0
## 1   1 - 2 1.29035 2501.931 2518.845 1.25764 0.38566  0.01790         0
```

```
finiteDLMauto(x = as.vector(temp+chem1+chem2+part), y = as.vector(mor), q.min = 1, q.max = 10
, 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 1.16004 2370.609 2391.662 1.09458 0.59736  0.18941         0
## 9   9 - 2 1.16575 2386.791 2407.854 1.03468 1.05329  0.16913         0
## 8   8 - 2 1.18770 2404.256 2425.329 1.11569 0.41622  0.14622         0
## 7   7 - 2 1.20381 2420.283 2441.366 1.11020 1.63230  0.12520         0
## 6   6 - 2 1.22530 2441.679 2462.772 1.13018 1.40049  0.09414         0
## 5   5 - 2 1.23389 2455.414 2476.517 1.15497 0.78596  0.07631         0
## 4   4 - 2 1.25679 2472.301 2493.414 1.16849 0.51433  0.05214         0
## 3   3 - 2 1.27490 2485.654 2506.777 1.21643 1.29587  0.03453         0
## 2   2 - 2 1.28116 2494.829 2515.962 1.23695 1.74173  0.02464         0
## 1   1 - 2 1.29035 2501.931 2518.845 1.25764 0.38566  0.01790         0
```

Thus, based on above results, lag-length 10 will be used and the order of the polynomial is adjusted to 2.

Fitting Polynomial DLM:

As all the other three series are to be considered predictors, let's fit the possible combination of 15 possible models when the other three series are taken as predictors together and alone.

```
# Fitting all combination of polynomial models

model.poly = polyDlm(x = as.vector(temp+chem1), y = as.vector(mor), q = 10, k = 2)
model.poly2 = polyDlm(x = as.vector(temp+chem2), y = as.vector(mor), q = 10, k = 2)
model.poly3 = polyDlm(x = as.vector(temp+part), y = as.vector(mor), q = 10, k = 2)
model.poly4 = polyDlm(x = as.vector(chem1+part), y = as.vector(mor), q = 10, k = 2)
model.poly5 = polyDlm(x = as.vector(chem2+part), y = as.vector(mor), q = 10, k = 2)
model.poly6 = polyDlm(x = as.vector(temp+chem1+chem2), y = as.vector(mor), q = 10, k = 2)
model.poly7 = polyDlm(x = as.vector(temp+chem1+part), y = as.vector(mor), q = 10, k = 2)
model.poly8 = polyDlm(x = as.vector(temp+chem2+part), y = as.vector(mor), q = 10, k = 2)
model.poly9 = polyDlm(x = as.vector(chem1+chem2+part), y = as.vector(mor), q = 10, k = 2)
model.poly10 = polyDlm(x = as.vector(temp), y = as.vector(mor), q = 10, k = 2)
model.poly11 = polyDlm(x = as.vector(chem1), y = as.vector(mor), q = 10, k = 2)
model.poly12 = polyDlm(x = as.vector(chem2), y = as.vector(mor), q = 10, k = 2)
model.poly13 = polyDlm(x = as.vector(part), y = as.vector(mor), q = 10, k = 2)
model.poly14 = polyDlm(x = as.vector(temp+chem1+chem2+part), y = as.vector(mor), q = 10, k =
2)
model.poly15 = polyDlm(x = as.vector(chem1+chem2), y = as.vector(mor), q = 10, k = 2)
```

Comparison based on AIC,BIC and MASE:

```
# Sorting based on AIC, BIC and MASE

sort.score(AIC(model.poly$model,model.poly2$model,model.poly3$model,model.poly4$model,model.p
oly5$model,model.poly6$model,model.poly7$model,model.poly8$model,model.poly9$model,model.poly
10$model,model.poly11$model,model.poly12$model,model.poly13$model,model.poly14$model,model.po
ly15$model), score = "aic")
```

```
##                     df      AIC
## model.poly13$model  5 2303.729
## model.poly4$model   5 2308.926
## model.poly5$model   5 2327.700
## model.poly9$model   5 2331.071
## model.poly8$model   5 2367.457
## model.poly10$model  5 2369.024
## model.poly14$model  5 2370.609
## model.poly$model    5 2373.583
## model.poly12$model  5 2379.735
## model.poly3$model   5 2383.690
## model.poly15$model  5 2385.176
## model.poly7$model   5 2387.510
## model.poly2$model   5 2442.407
## model.poly6$model   5 2443.494
## model.poly11$model  5 2469.016
```

```
sort.score(BIC(model.poly$model,model.poly2$model,model.poly3$model,model.poly4$model,model.p
oly5$model,model.poly6$model,model.poly7$model,model.poly8$model,model.poly9$model,model.poly
10$model,model.poly11$model,model.poly12$model,model.poly13$model,model.poly14$model,model.po
ly15$model), score = "bic")
```

```
##                     df      BIC
## model.poly13$model  5 2324.782
## model.poly4$model   5 2329.979
## model.poly5$model   5 2348.753
## model.poly9$model   5 2352.124
## model.poly8$model   5 2388.510
## model.poly10$model  5 2390.077
## model.poly14$model  5 2391.662
## model.poly$model    5 2394.636
## model.poly12$model  5 2400.788
## model.poly3$model   5 2404.743
## model.poly15$model  5 2406.229
## model.poly7$model   5 2408.563
## model.poly2$model   5 2463.460
## model.poly6$model   5 2464.547
## model.poly11$model  5 2490.069
```

```
sort.score(MASE(model.poly$model,model.poly2$model,model.poly3$model,model.poly4$model,model.
poly5$model,model.poly6$model,model.poly7$model,model.poly8$model,model.poly9$model,model.pol
y10$model,model.poly11$model,model.poly12$model,model.poly13$model,model.poly14$model,model.p
oly15$model), score = "mase")
```

```
##                       n     MASE
## model.poly13$model 498 1.091676
## model.poly4$model  498 1.100952
## model.poly5$model  498 1.105860
## model.poly9$model  498 1.112082
## model.poly8$model  498 1.153637
## model.poly12$model 498 1.157062
## model.poly14$model 498 1.160041
## model.poly15$model 498 1.170545
## model.poly3$model  498 1.180714
## model.poly10$model 498 1.186847
## model.poly$model   498 1.187319
## model.poly7$model  498 1.187372
## model.poly2$model  498 1.246725
## model.poly6$model  498 1.249934
## model.poly11$model 498 1.300309
```

Based on AIC and BIC and MASE values, model.poly13 is the best model with only part as predictor. Let's analyze this model.

Analyzing model.poly13:

```
summary(model.poly13$model)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.5312 -1.3248 -0.1206  0.9283 20.0827
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.0126002  0.4890325   4.115 4.53e-05 ***
## z.t0         0.0056541  0.0069709   0.811    0.418
## z.t1        -0.0025051  0.0040304  -0.622    0.535
## z.t2         0.0005464  0.0003980   1.373    0.170
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.43 on 494 degrees of freedom
## Multiple R-squared:  0.2956, Adjusted R-squared:  0.2913
## F-statistic: 69.09 on 3 and 494 DF,  p-value: < 2.2e-16
```

From the above summary, we can generate following insights about the model.poly13-

1. The model model.poly13 is a poorly fitted model and almost all lags are insignificant.
2. The value of adjusted R2 is 0.2956. This means that model.poly13 is able to explain only (~30%) variation in the dependent variable (mortality).
3. The residuals have maximum value of 20.0827 and minimum value of -5.5312 with residual standard error (RSE) as 2.43.
4. As per the F-test of the overall significance of model, the model.poly13 is a significant model at 5% level of significance despite having many insignificant number of lags and poor R2 value.

Calculating VIF for model.poly13 DLM:

Calculating VIF for the model.poly13 DLM to check for presence of multicollinearity in the model-

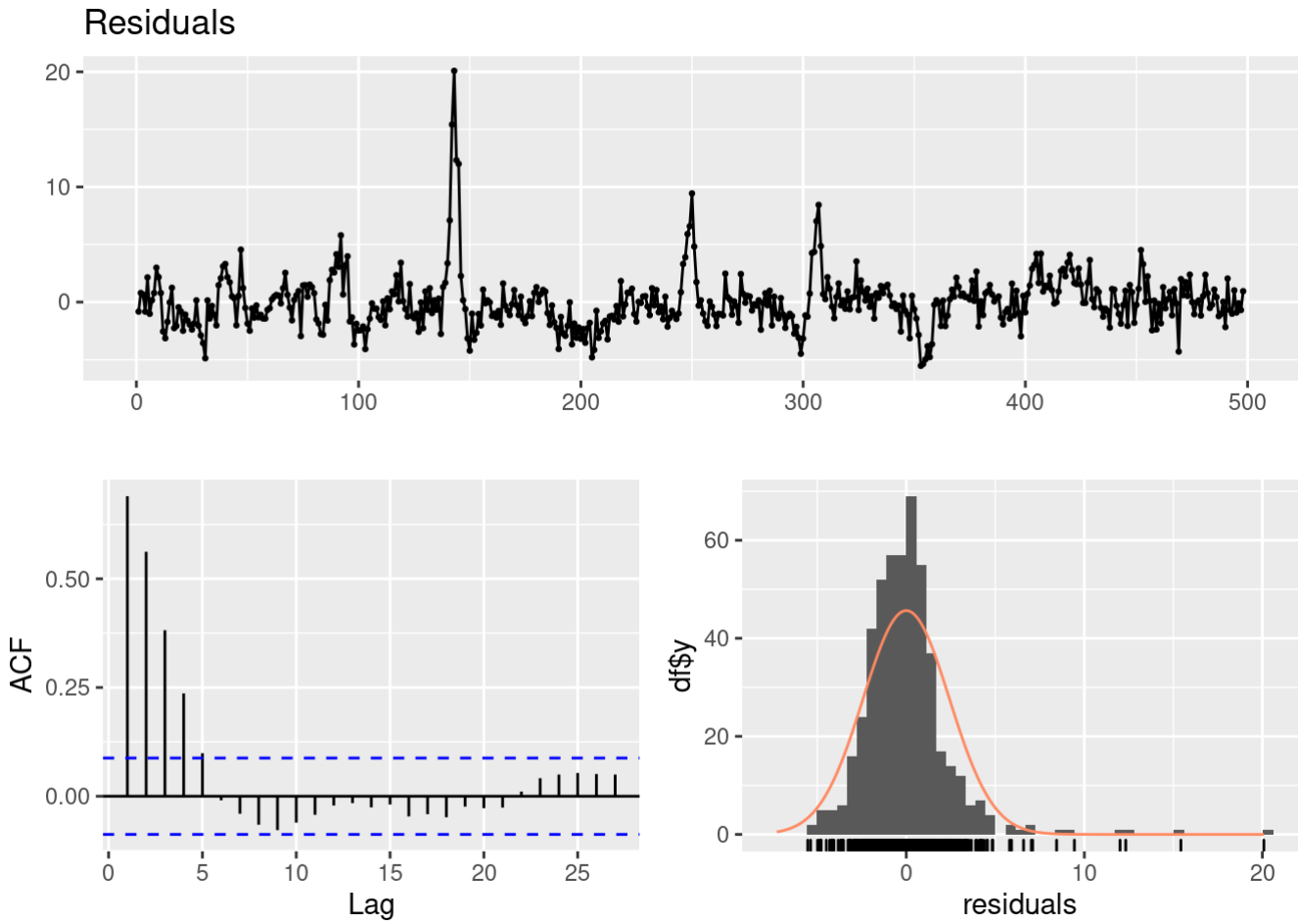
```
vif(model.poly13$model)
```

```
##      z.t0      z.t1      z.t2
##  61.86387 561.60794 283.59598
```

As seen above, the vif > 10 for all the lags in the model. Hence, there is serious issue of multicollinearity in the model.poly13 model.

Residual Analysis for model.poly DLM:

```
#Residual analysis of model.poly13 model


checkresiduals(model.poly13$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 252.95, df = 10, p-value < 2.2e-16
```

As per the results of the Breusch-Godfrey test, the p-value is less than 5% level of significance which rejects the null hypothesis. Thus, there is presence of serial correlation in the residuals. The residual analysis generated after fitting polynomial DLM are similar to the ones generated for finite DLM. The residuals seem to

follow a trend and not distributed randomly. There is a decaying pattern seen in the ACF which means there is serial correlation present in the residuals (supporting the Breusch-Godfrey test) . Other than that, there is weavy pattern observed at seasonal delays which show seasonlity & autocorrelation still exist in residuals. The histogram of residuals also does not seem to appear normal.

Overall, the polynomial DLM fitted is a poor model and is worse than finite DLM due to presence of multicollinearity.

Koyck Transformation DLM:

The Koyck model is similar to a regression model. The Koyck DLM will be fitted with multiple variables as predictor and mor as dependent variable.

Fittng all possible Koyck models:

```
model.koyck = koyckDlm(x = as.vector(temp+chem1), y = as.vector(mor))
model.koyck2 = koyckDlm(x = as.vector(temp+chem2), y = as.vector(mor))
model.koyck3 = koyckDlm(x = as.vector(temp+part), y = as.vector(mor))
model.koyck4 = koyckDlm(x = as.vector(chem1+part), y = as.vector(mor))
model.koyck5 = koyckDlm(x = as.vector(chem2+part), y = as.vector(mor))
model.koyck6 = koyckDlm(x = as.vector(temp+chem1+chem2), y = as.vector(mor))
model.koyck7 = koyckDlm(x = as.vector(temp+chem1+part), y = as.vector(mor))
model.koyck8 = koyckDlm(x = as.vector(temp+chem2+part), y = as.vector(mor))
model.koyck9 = koyckDlm(x = as.vector(chem1+chem2+part), y = as.vector(mor))
model.koyck10 = koyckDlm(x = as.vector(temp), y = as.vector(mor))
model.koyck11 = koyckDlm(x = as.vector(chem1), y = as.vector(mor))
model.koyck12 = koyckDlm(x = as.vector(chem2), y = as.vector(mor))
model.koyck13 = koyckDlm(x = as.vector(part), y = as.vector(mor))
model.koyck14 = koyckDlm(x = as.vector(temp+chem1+chem2+part), y = as.vector(mor))
model.koyck15 = koyckDlm(x = as.vector(chem1+chem2), y = as.vector(mor))
```

Comparison based on MASE:

```
# Sorting based on  MASE


Mase <- MASE(model.koyck,model.koyck2,model.koyck3,model.koyck4,model.koyck5,model.koyck6,mod
el.koyck7,model.koyck8,model.koyck9,model.koyck10,model.koyck11,model.koyck12,model.koyck13,m
odel.koyck14,model.koyck15)


# Sorting in ascending order

arrange(Mase,MASE)
```

```
##                 n      MASE
## model.koyck5  507 0.9194400
## model.koyck9  507 0.9201017
## model.koyck13 507 0.9221834
## model.koyck4  507 0.9228863
## model.koyck12 507 0.9255483
## model.koyck15 507 0.9281007
## model.koyck8  507 0.9392184
## model.koyck14 507 0.9411883
## model.koyck3  507 0.9549093
## model.koyck7  507 0.9581315
## model.koyck11 507 0.9718873
## model.koyck10 507 1.0024611
## model.koyck   507 1.0132028
## model.koyck6  507 1.1075274
## model.koyck2  507 1.1093516
```

From the above results, model.koyck5 (with chem1 and part as predictor) is the best model based on MASE value. Let's analyze this model further.

Analyzing model.koyck5:

```
summary(model.koyck5,diagnostics = TRUE)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.1067 -1.1350 -0.1509  1.0293 10.3496
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.908934   0.601936   1.510   0.1317
## Y.1         0.757612   0.029361  25.803   <2e-16 ***
## X.t         0.011440   0.006511   1.757   0.0795 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.773 on 504 degrees of freedom
## Multiple R-Squared: 0.6185,  Adjusted R-squared: 0.617
## Wald test:   400 on 2 and 504 DF,  p-value: < 2.2e-16
##
## Diagnostic tests:
##                  df1 df2   statistic      p-value
## Weak instruments   1 504 128.1108568 1.277602e-26
## Wu-Hausman         1 503   0.1044454 7.466931e-01
##
##                             alpha       beta       phi
## Geometric coefficients:  3.749909 0.01144024 0.7576119
```

From the above summary, we can generate following insights about the model.koyck5-

1. The model model.koyck is a decently fitted model and most of the lags are insignificant.

2. The value of adjusted R2 is 0.617. This means that model.koyck5 is able to explain 61.7% variation in the dependent variable (mortality).
3. The residuals have maximum value of 10.3496 and minimum value of -6.1067 with residual standard error (RSE) as 1.773.
4. As per the F-test of the overall significance of model, the model.koyck5 is a pretty significant model at 5% level of significance despite having a few insignificant number of lags and decent R2 value.
5. The model in the first stage of least-squares estimation is significant at the 5% level, according to the Weak instruments test.
6. We may infer that there is a significant connection between the explanatory variable and the error term at the 5% level based on the Wu-Hausman test.
7. The geometric coefficients alpha, beta and phi are 3.749909, 0.01144024 and 0.7576119 respectively.
8. This model is a better fit in comparison to other previous models in terms of R2 and F-test p-value.

Calculating VIF for model.koyck5 DLM:

Calculating VIF for the model.koyck5 DLM to check for presence of multicollinearity in the model-

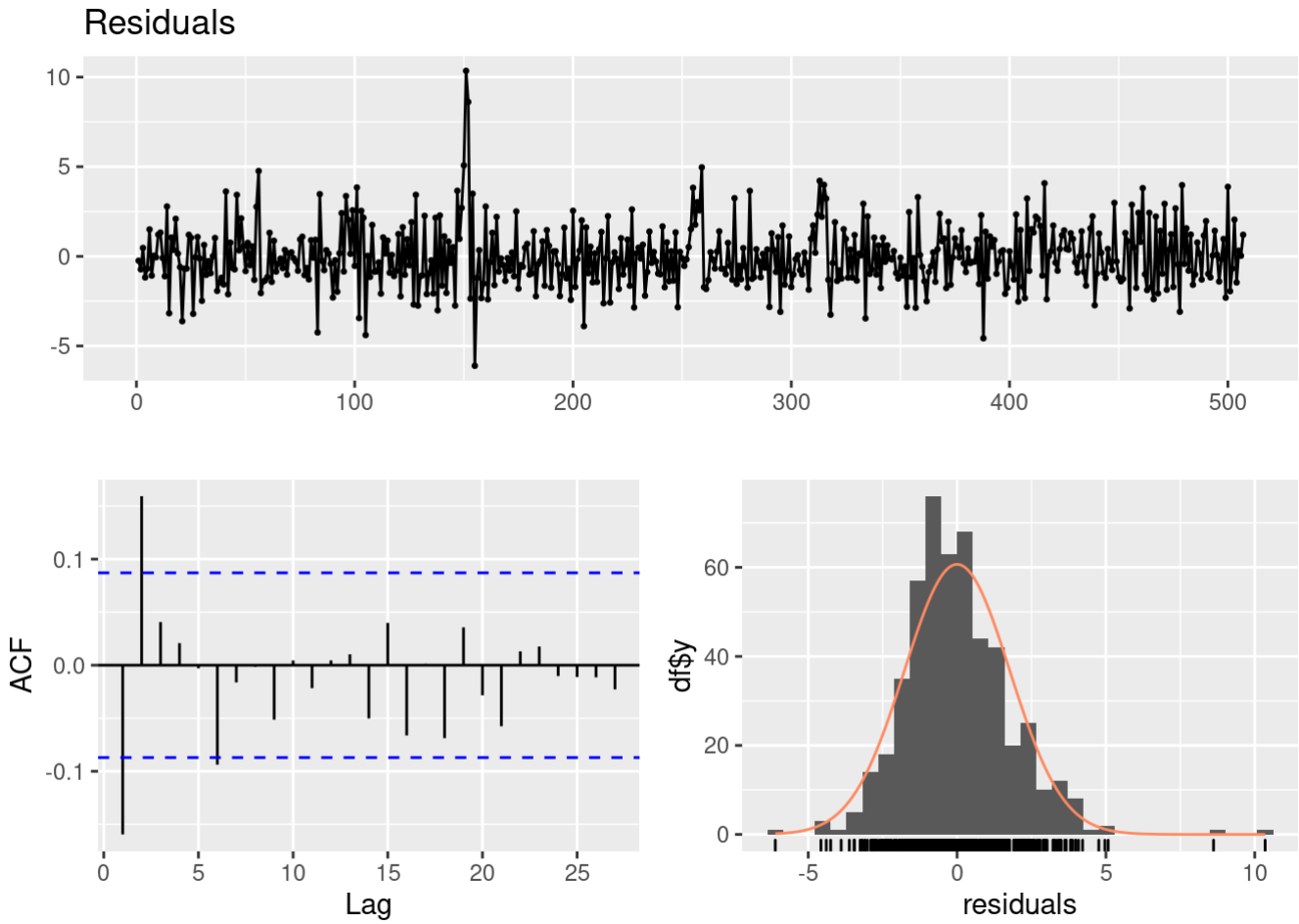
```
vif(model.koyck5$model)
```

```
##      Y.1      X.t
## 1.141484 1.141484
```

As seen above, the vif < 10 for all the lags in the model. Hence, there is no issue of multicollinearity in the model.koyck5 model.

Residual Analysis for model.koyck5 DLM:

```
#Residual analysis of model.koyck5 model

checkresiduals(model.koyck5$model)
```



From the above plot, we can say that, the residuals are not randomly distributed. There seems to be presence of serial correlation & seasonality as per the ACF as two lags are significant. The histogram shows that the residuals are not normally distributed.

As per the diagnostic checking (residual analysis and other tests), Koyck DLM seems to be largely better in comparison to the previous two models as the model is significant and does not face any issue of multicollinearity.

Autoregressive DLM:

Finding autoregressive DLMs for a range of lag lengths and AR process orders using the loop, and fitting the model with the lowest information criterion-

```
for (i in 1:5){
  for(j in 1:5){
    model.autoreg = ardlDlm(x= as.vector(temp+chem1+chem2+part),y=as.vector(mor), 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 = 2032.008 BIC = 2053.15 MASE = 0.9261488
## p = 1 q = 2 AIC = 2008.699 BIC = 2034.059 MASE = 0.8958695
## p = 1 q = 3 AIC = 2004.177 BIC = 2033.749 MASE = 0.8969723
## p = 1 q = 4 AIC = 1999.668 BIC = 2033.449 MASE = 0.8994944
## p = 1 q = 5 AIC = 1996.897 BIC = 2034.882 MASE = 0.8947375
## p = 2 q = 1 AIC = 2029.096 BIC = 2054.455 MASE = 0.9245304
## p = 2 q = 2 AIC = 2010.09 BIC = 2039.676 MASE = 0.8953451
## p = 2 q = 3 AIC = 2005.693 BIC = 2039.49 MASE = 0.8966598
## p = 2 q = 4 AIC = 2001.272 BIC = 2039.276 MASE = 0.8980602
## p = 2 q = 5 AIC = 1998.563 BIC = 2040.769 MASE = 0.8942801
## p = 3 q = 1 AIC = 2026.831 BIC = 2056.403 MASE = 0.9251114
## p = 3 q = 2 AIC = 2007.744 BIC = 2041.54 MASE = 0.896793
## p = 3 q = 3 AIC = 2006.111 BIC = 2044.132 MASE = 0.9002296
## p = 3 q = 4 AIC = 2001.897 BIC = 2044.123 MASE = 0.9023498
## p = 3 q = 5 AIC = 1999.29 BIC = 2045.716 MASE = 0.8983536
## p = 4 q = 1 AIC = 2023.084 BIC = 2056.864 MASE = 0.9187279
## p = 4 q = 2 AIC = 2004.522 BIC = 2042.526 MASE = 0.8929983
## p = 4 q = 3 AIC = 2002.804 BIC = 2045.03 MASE = 0.8962018
## p = 4 q = 4 AIC = 2000.511 BIC = 2046.959 MASE = 0.8955198
## p = 4 q = 5 AIC = 1997.976 BIC = 2048.623 MASE = 0.8911169
## p = 5 q = 1 AIC = 2017.776 BIC = 2055.762 MASE = 0.9130554
## p = 5 q = 2 AIC = 1999.399 BIC = 2041.604 MASE = 0.8874112
## p = 5 q = 3 AIC = 1997.547 BIC = 2043.973 MASE = 0.8911133
## p = 5 q = 4 AIC = 1995.124 BIC = 2045.771 MASE = 0.8906554
## p = 5 q = 5 AIC = 1994.843 BIC = 2049.711 MASE = 0.8863831
```

From the above output, the model with lowest MASE is ARDL(5,5). Hence, this model will be explored further using summary and diagnostic checking.

Fitting ARDL(5,5) for all combination of predictors:

```
model.autoreg = ardlDlm(x=as.vector(temp), y=as.vector(mor), p = 5, q = 5)
model.autoreg2 = ardlDlm(x=as.vector(chem1), y=as.vector(mor), p = 5, q = 5)
model.autoreg3 = ardlDlm(x=as.vector(chem2), y=as.vector(mor), p = 5, q = 5)
model.autoreg4 = ardlDlm(x=as.vector(temp+chem1), y=as.vector(mor), p = 5, q = 5)
model.autoreg5 = ardlDlm(x=as.vector(temp+chem2), y=as.vector(mor), p = 5, q = 5)
model.autoreg6 = ardlDlm(x=as.vector(temp+part), y=as.vector(mor), p = 5, q = 5)
model.autoreg7 = ardlDlm(x=as.vector(chem1+chem2), y=as.vector(mor), p = 5, q = 5)
model.autoreg8 = ardlDlm(x=as.vector(chem1+part), y=as.vector(mor), p = 5, q = 5)
model.autoreg9 = ardlDlm(x=as.vector(chem2+part), y=as.vector(mor), p = 5, q = 5)
model.autoreg10 = ardlDlm(x=as.vector(temp+chem1+chem2), y=as.vector(mor), p = 5, q = 5)
model.autoreg11 = ardlDlm(x=as.vector(temp+chem1+part), y=as.vector(mor), p = 5, q = 5)
model.autoreg12 = ardlDlm(x=as.vector(temp+chem2+part), y=as.vector(mor), p = 5, q = 5)
model.autoreg13 = ardlDlm(x=as.vector(chem1+chem2+part), y=as.vector(mor), p = 5, q = 5)
model.autoreg14 = ardlDlm(x=as.vector(part), y=as.vector(mor), p = 5, q = 5)
model.autoreg15 = ardlDlm(x=as.vector(temp+chem1+chem2+part), y=as.vector(mor), p = 5, q = 5)
```

Comparison based on AIC,BIC and MASE:

```
# Sorting based on AIC, BIC and MASE

sort.score(AIC(model.autoreg$model,model.autoreg2$model,model.autoreg3$model,model.autoreg4$m
odel,model.autoreg5$model,model.autoreg6$model,model.autoreg7$model,model.autoreg8$model,mode
l.autoreg9$model,model.autoreg10$model,model.autoreg11$model,model.autoreg12$model,model.auto
reg13$model,model.autoreg14$model,model.autoreg15$model), score = "aic")
```

```
##                        df      AIC
## model.autoreg$model   13 1984.211
## model.autoreg4$model  13 1985.261
## model.autoreg14$model 13 1987.645
## model.autoreg9$model  13 1987.803
## model.autoreg13$model 13 1988.432
## model.autoreg8$model  13 1988.437
## model.autoreg3$model  13 1993.320
## model.autoreg12$model 13 1994.392
## model.autoreg7$model  13 1994.394
## model.autoreg15$model 13 1994.843
## model.autoreg6$model  13 1998.809
## model.autoreg11$model 13 1999.157
## model.autoreg5$model  13 2002.008
## model.autoreg10$model 13 2002.184
## model.autoreg2$model  13 2008.913
```

```
sort.score(BIC(model.autoreg$model,model.autoreg2$model,model.autoreg3$model,model.autoreg4$m
odel,model.autoreg5$model,model.autoreg6$model,model.autoreg7$model,model.autoreg8$model,mode
l.autoreg9$model,model.autoreg10$model,model.autoreg11$model,model.autoreg12$model,model.auto
reg13$model,model.autoreg14$model,model.autoreg15$model), score = "bic")
```

```
##                        df      BIC
## model.autoreg$model   13 2039.078
## model.autoreg4$model  13 2040.129
## model.autoreg14$model 13 2042.512
## model.autoreg9$model  13 2042.671
## model.autoreg13$model 13 2043.299
## model.autoreg8$model  13 2043.304
## model.autoreg3$model  13 2048.188
## model.autoreg12$model 13 2049.259
## model.autoreg7$model  13 2049.261
## model.autoreg15$model 13 2049.711
## model.autoreg6$model  13 2053.677
## model.autoreg11$model 13 2054.025
## model.autoreg5$model  13 2056.876
## model.autoreg10$model 13 2057.052
## model.autoreg2$model  13 2063.780
```

```
Mase2 <- MASE(model.autoreg,model.autoreg2,model.autoreg3,model.autoreg4,model.autoreg5,mode
l.autoreg6,model.autoreg7,model.autoreg8,model.autoreg9,model.autoreg10,model.autoreg11,mode
l.autoreg12,model.autoreg13,model.autoreg14,model.autoreg15)


arrange(Mase2,MASE)
```

```
##                   n      MASE
## model.autoreg9  503 0.8823001
## model.autoreg13 503 0.8828656
## model.autoreg   503 0.8829339
## model.autoreg14 503 0.8833224
## model.autoreg4  503 0.8837403
## model.autoreg8  503 0.8837951
## model.autoreg12 503 0.8859138
## model.autoreg15 503 0.8863831
## model.autoreg3  503 0.8870239
## model.autoreg6  503 0.8878067
## model.autoreg11 503 0.8881237
## model.autoreg7  503 0.8881683
## model.autoreg5  503 0.8915162
## model.autoreg10 503 0.8917270
## model.autoreg2  503 0.8991869
```

From the AIC and BIC values, model with only temp as predictor is the best model. While, based on MASE, the model with chem2 & part as predictors is the best model. Let's analyze the latter model as it is based on MASE.

Fitting ARDL(5,5) with chem2 & part as predictors:

```
model.reg = ardlDlm(x=as.vector(chem2+part), y=as.vector(mor), p = 5, q = 5)
summary(model.reg)
```

```
##
## Time series regression with "ts" data:
## Start = 6, End = 508
##
## Call:
## dynlm(formula = as.formula(model.text), data = data, start = 1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.0560 -1.0654 -0.1063  1.0234 10.5455
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.5928999  0.4178914   1.419   0.1566
## X.t          0.0087438  0.0036974   2.365   0.0184 *
## X.1         -0.0058362  0.0037323  -1.564   0.1185
## X.2         -0.0003246  0.0038359  -0.085   0.9326
## X.3          0.0006187  0.0038457   0.161   0.8723
## X.4          0.0054989  0.0037749   1.457   0.1458
## X.5          0.0093816  0.0037840   2.479   0.0135 *
## Y.1          0.5909291  0.0448077  13.188  < 2e-16 ***
## Y.2          0.2657059  0.0520893   5.101 4.84e-07 ***
## Y.3         -0.0103107  0.0534259  -0.193   0.8470
## Y.4         -0.0542166  0.0521089  -1.040   0.2986
## Y.5         -0.0737814  0.0445576  -1.656   0.0984 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.722 on 491 degrees of freedom
## Multiple R-squared:  0.6489, Adjusted R-squared:  0.6411
## F-statistic: 82.51 on 11 and 491 DF,  p-value: < 2.2e-16
```

From the above summary of model.reg model, we can generate following insights-

1. There are only 3 significant terms in the model.
2. The value of adjusted R2 is moderate (64%). This means that model.reg is able to explain almost 64% variation in the dependent variable (mortality).
3. The residuals have maximum value of 10.5455 and minimum value of -6.0560 with residual standard error (RSE) as 1.722 which is better in comparison to previous three models.
4. As per the F-test of the overall significance of model, the model.reg is a significant model at 5% level of significance.
5. In comparison to the previous three models, the model.reg generated good summary statistics but not better than the finite DLM model.

VIF for ARDL(5,5):

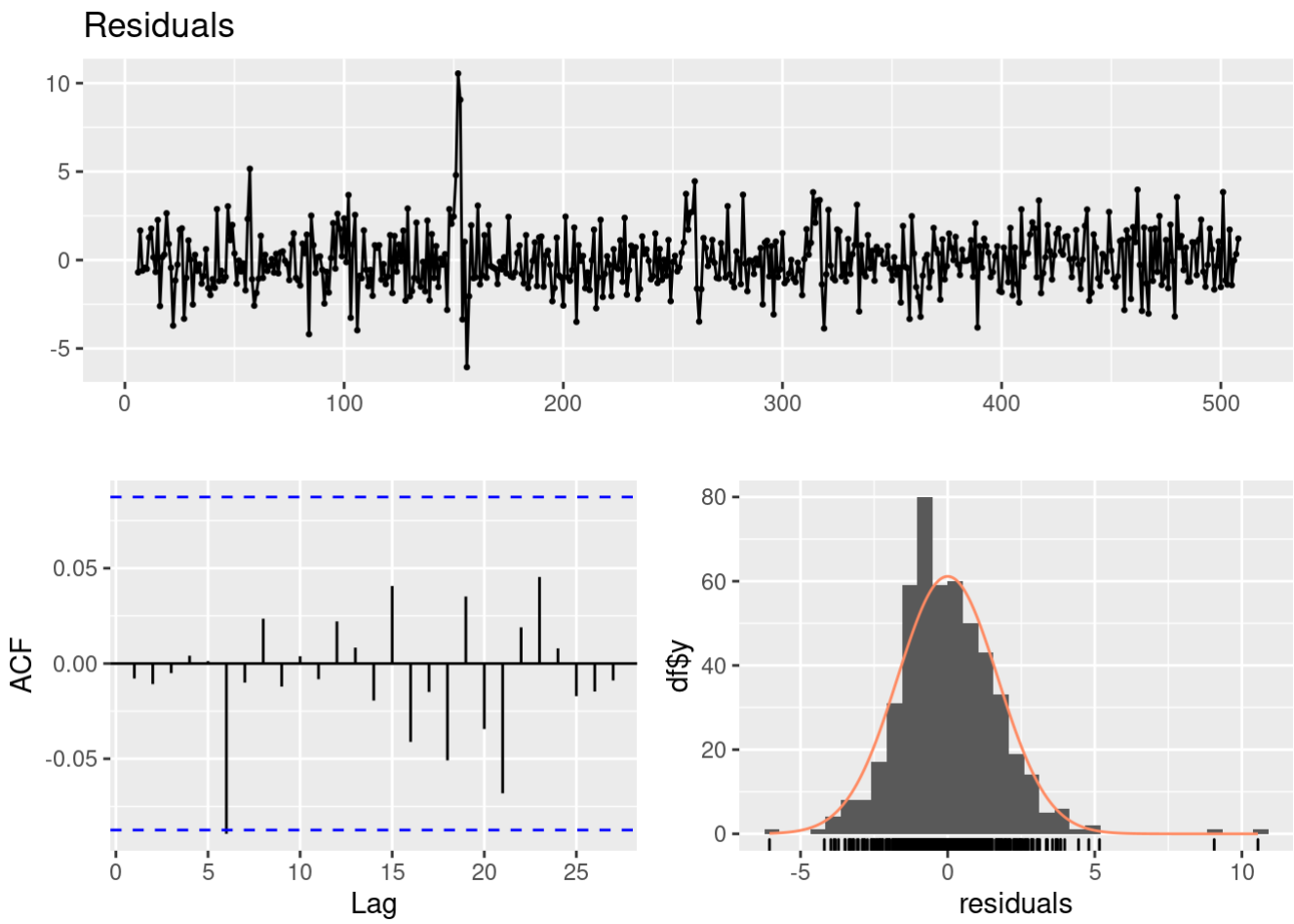
```
vif(model.reg$model)
```

```
##       X.t L(X.t, 1) L(X.t, 2) L(X.t, 3) L(X.t, 4) L(X.t, 5) L(y.t, 1) L(y.t, 2)
##  1.734211  1.766734  1.865391  1.871623  1.803057  1.812842  2.806088  3.793299
## L(y.t, 3) L(y.t, 4) L(y.t, 5)
##  3.991264  3.801933  2.786670
```

As VIF is less than 10 for all elements in the model, there is no presence of multicollinearity in the model.

Residual Analysis (Diagnostic checking) of ARDL(5,5):

```
checkresiduals(model.reg$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 15
##
## data:  Residuals
## LM test = 12.02, df = 15, p-value = 0.6775
```

The Breusch-Godfrey test indicate that there is no serial correlation among the residuals as the test with p-value being insignificant. The residuals plot indicate that residuals are not randomly distributed and seasonal pattern can be observed. In the ACF plot, there are no significant lags. As a result, autocorrelation is not found in the residuals. The histogram shows residuals do not seem to follow normal distribution.

Checking MASE for all the four models:

Now, computing MASE for all four models-

```
cal_mase <- MASE(model.finite,model.poly13,model.koyck5,model.reg)
print(cal_mase)
```

```
##                n      MASE
## model.finite 498 1.0877112
## model.poly13 498 1.0916760
## model.koyck5 507 0.9194400
## model.reg    503 0.8823001
```

Out of the 4 best models computed using finite, polynomial, koyck and autoregressive methods, the four best models were found using AIC, BIC and MASE criteria. Out of these four models, model.finite has the lowest MASE value, moderate R2 value and is free from multicollinearity. Hence, model.finite with part as predictor is the best model.

Modelling using Dynamic Linear Models

The Dynamic Linear models are often considered suitable for forecasting. Let's fit some dynamic linear models in mor series-

```
# Computing parameters

Y.t <- mor
T <-  508
S.t <-  1*(seq(Y.t) == T)
S.t.1 <-  Lag(S.t,+1)

# Fitting 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 ~ part + L(Y.t , k = 2 ) + S.t + trend(Y.t) + season(Y.t))
```

Computing MASE:

Now, let's calculate and compare the MASE value for each dynamic model fitted above-

```
#Calculating MASE

cal_mase2 <- MASE(lm(dynlm1), lm(dynlm2), lm(dynlm3), lm(dynlm4),lm(dynlm5),lm(dynlm6),lm(dyn
lm7),lm(dynlm8))
arrange(cal_mase2,MASE)
```

```
##              n      MASE
## lm(dynlm1) 507 0.8558330
## lm(dynlm3) 507 0.8558596
## lm(dynlm7) 506 0.8678083
## lm(dynlm6) 506 0.8710927
## lm(dynlm8) 506 0.8720844
## lm(dynlm5) 506 0.8724418
## lm(dynlm2) 506 0.8796086
## lm(dynlm4) 507 0.9396163
```

The best dynamic model with the lowest MASE value was found to be dynlm1 with trend & seasonality. Let's generate a summary and residual analysis of the model.

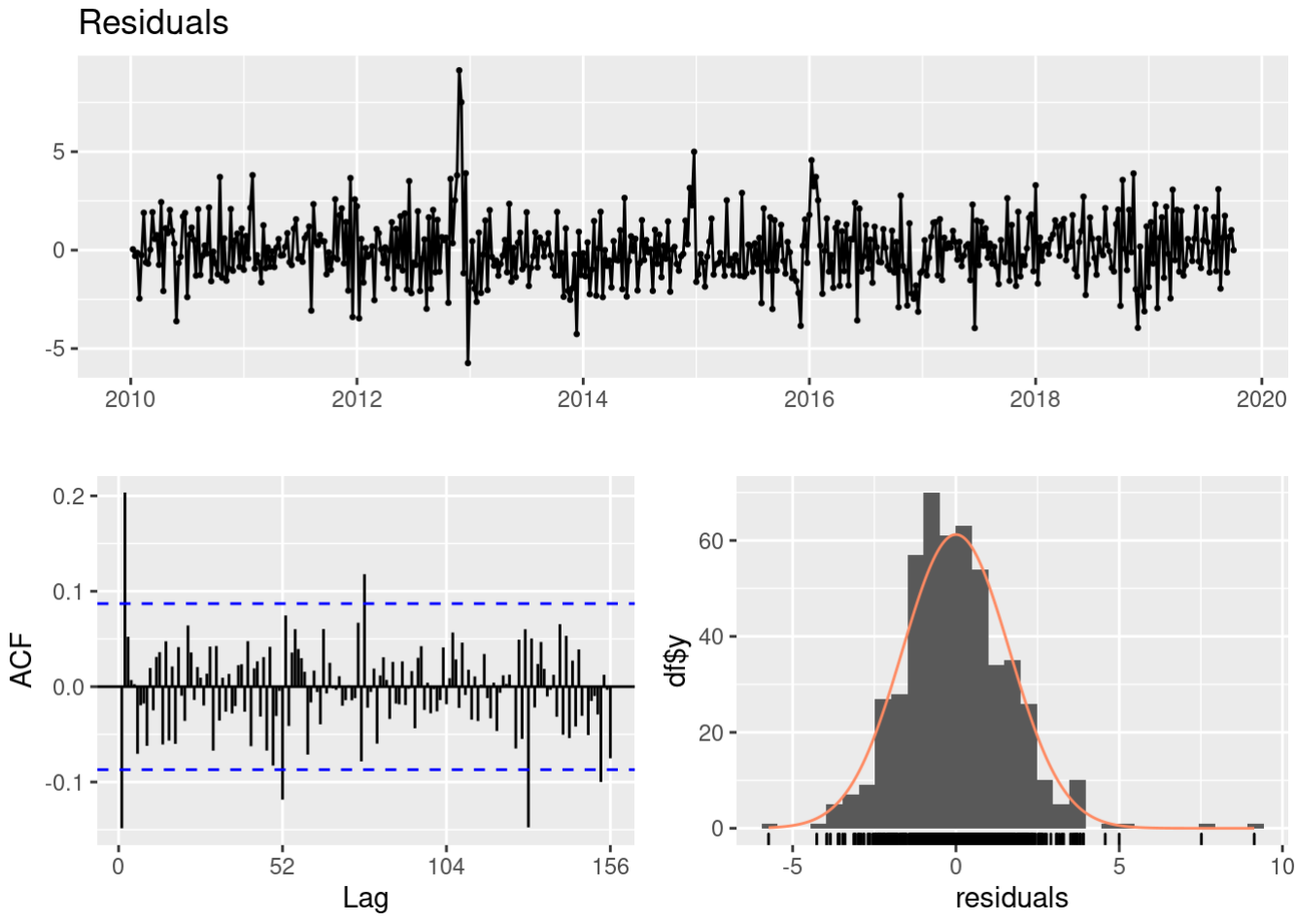
Residual Analysis for dynlm1 model:

```
# Generating summary of dynlm1 model

summary(dynlm1)
```

```
##
## Time series regression with "ts" data:
## Start = 2010(2), End = 2019(40)
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, k = 1) + S.t + trend(Y.t) + season(Y.t))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7383 -1.0525 -0.1103  0.9773  9.1357
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.5081403  0.7193276   4.877 1.49e-06 ***
## L(Y.t, k = 1)  0.6747982  0.0347499  19.419  < 2e-16 ***
## S.t            1.3997817  1.8327323   0.764   0.4454
## trend(Y.t)     0.0005345  0.0274832   0.019   0.9845
## season(Y.t)2  -0.8311356  0.7962653  -1.044   0.2971
## season(Y.t)3  -1.1350550  0.7974279  -1.423   0.1553
## season(Y.t)4  -0.1292427  0.8000964  -0.162   0.8717
## season(Y.t)5   0.7817384  0.7990436   0.978   0.3284
## season(Y.t)6  -1.2848182  0.7966844  -1.613   0.1075
## season(Y.t)7  -1.3822412  0.7995444  -1.729   0.0845 .
## season(Y.t)8  -1.2766485  0.8033524  -1.589   0.1127
## season(Y.t)9  -0.5864954  0.8061326  -0.728   0.4673
## season(Y.t)10 -1.1354167  0.8044732  -1.411   0.1588
## season(Y.t)11 -1.5473696  0.8062176  -1.919   0.0556 .
## season(Y.t)12 -0.3618586  0.8099983  -0.447   0.6553
## season(Y.t)13 -0.9773415  0.8055757  -1.213   0.2257
## season(Y.t)14 -1.3768232  0.8061404  -1.708   0.0883 .
## season(Y.t)15 -0.6440035  0.8088661  -0.796   0.4263
## season(Y.t)16 -1.9420550  0.8064385  -2.408   0.0164 *
## season(Y.t)17 -1.3931208  0.8128207  -1.714   0.0872 .
## season(Y.t)18 -1.3820089  0.8137968  -1.698   0.0902 .
## season(Y.t)19 -0.7976857  0.8143882  -0.979   0.3279
## season(Y.t)20 -1.1726642  0.8107130  -1.446   0.1487
## season(Y.t)21 -1.7025281  0.8108283  -2.100   0.0363 *
## season(Y.t)22 -0.5637978  0.8146062  -0.692   0.4892
## season(Y.t)23 -1.1824271  0.8093398  -1.461   0.1447
## season(Y.t)24 -1.1429819  0.8099513  -1.411   0.1589
## season(Y.t)25 -1.1984475  0.8101164  -1.479   0.1397
## season(Y.t)26 -1.4841975  0.8105884  -1.831   0.0678 .
## season(Y.t)27 -1.5221504  0.8128577  -1.873   0.0618 .
## season(Y.t)28 -1.4550139  0.8147588  -1.786   0.0748 .
## season(Y.t)29 -0.8791216  0.8155875  -1.078   0.2817
## season(Y.t)30 -1.2362058  0.8119839  -1.522   0.1286
## season(Y.t)31 -1.3583949  0.8121144  -1.673   0.0951 .
## season(Y.t)32 -1.7843074  0.8130545  -2.195   0.0287 *
## season(Y.t)33 -0.0404470  0.8168925  -0.050   0.9605
## season(Y.t)34 -0.6938008  0.8074156  -0.859   0.3906
## season(Y.t)35 -1.8241529  0.8058101  -2.264   0.0241 *
## season(Y.t)36 -0.6965521  0.8114758  -0.858   0.3911
## season(Y.t)37 -1.0892623  0.8083191  -1.348   0.1785
## season(Y.t)38 -0.8691343  0.8086730  -1.075   0.2831
## season(Y.t)39 -1.6042830  0.8075968  -1.986   0.0476 *
## season(Y.t)40 -0.7346202  0.8329989  -0.882   0.3783
## season(Y.t)41 -1.3258598  0.8296669  -1.598   0.1107
```

```
## season(Y.t)42 -0.4472247  0.8311860  -0.538   0.5908
## season(Y.t)43 -0.8826741  0.8269456  -1.067   0.2864
## season(Y.t)44 -0.5408417  0.8266902  -0.654   0.5133
## season(Y.t)45  0.6416678  0.8247868   0.778   0.4370
## season(Y.t)46 -0.2007474  0.8193560  -0.245   0.8066
## season(Y.t)47  1.1834065  0.8190211   1.445   0.1492
## season(Y.t)48  1.5966371  0.8168538   1.955   0.0512 .
## season(Y.t)49  1.6966881  0.8177618   2.075   0.0386 *
## season(Y.t)50  0.0785281  0.8203323   0.096   0.9238
## season(Y.t)51  0.0866673  0.8180042   0.106   0.9157
## season(Y.t)52 -0.1750765  0.8171309  -0.214   0.8304
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.733 on 452 degrees of freedom
## Multiple R-squared:  0.6733, Adjusted R-squared:  0.6342
## F-statistic: 17.25 on 54 and 452 DF,  p-value: < 2.2e-16
```

```
checkresiduals(dynlm1)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 101
##
## data:  Residuals
## LM test = 117.17, df = 101, p-value = 0.1296
```

From the above summary of dynlm1 model, we can generate following insights-

1. Most of the terms in the model are insignificant at 5% level of significance.

2. The value of adjusted R2 is moderate (63.42%). This means that dynamic model fitted is able to explain almost 63.42% variation in the dependent variable (mor).
3. The residuals have maximum value of 9.1357 and minimum value of -5.7383 with residual standard error (RSE) as 1.733 which is better in comparison to previous models.
4. As per the F-test of the overall significance of model, the dynml1 is an insignificant model at 5% level of significance.
5. In comparison to other DLM models, the dynamic model generated poor statistics.

Residual Analysis: The Breusch-Godfrey test indicate that there is no serial correlation among the residuals as the test with p-value being insignificant. The residuals plot indicate that residuals are not randomly distributed and changing variance can be observed. In the ACF plot, there are a few extremely significant seasonal lags. As a result, autocorrelation and seasonality can still be found in the residuals which defy the Breusch-Godfrey test. The histogram shows residuals do not seem to follow normal distribution.

Checking for multicollinearity:

```
# Checking multicollinearity

vif(dynlm1)
```

```
##                   GVIF Df GVIF^(1/(2*Df))
## L(Y.t, k = 1) 1.674316  1        1.293954
## S.t           1.116603  1        1.056694
## trend(Y.t)    1.010479  1        1.005226
## season(Y.t)   1.864293 51        1.006125
```

The VIF is less than 10, hence there is no issue of multicollinearity present in the model. Hence, the dynamic linear model fitted is not a better model than other DLM models fitted before as the model is not significant even though its MASE value is better than the autoreg model generated from auto regressive method.

Modelling using Exponential Smoothing

We'll also test exponential smoothing as a forecasting approach. We will only examine models that incorporate either additive or multiplicative seasonality since we have discovered a substantial seasonal component in the mortality series for which we wish to make predictions. We fit them and record their accuracy measurements since there are 6 potential combinations of exponential smoothing models with seasonality that may be implemented in R.

Converting the mortality series to monthly:

Since exponential smoothing methods do not support weekly data, converting the series to monthly using zoo and aggregate functions.

```
# Converting mortality data to monthly and storing in mortalitymonthly

mortalitymonthly = aggregate(zoo(mor),as.yearmon,sum)

# Removing NA values

mortalitymonthly[mortalitymonthly==10.11] <- NA

mortalitymonthly = na.omit(mortalitymonthly)

# Transforming to proper form

mortalitymonthly <- ts(as.vector(t(as.matrix(mortalitymonthly[,2:13]))),start=c(2010,1),frequ
ency = 12)

# View trnasformed data

mortalitymonthly
```

```
##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
## 2010 49.79 31.90 35.94 41.50 35.13 28.02 35.83 30.88 31.59 40.32 41.91 52.79
## 2011 58.96 37.49 29.70 32.69 28.74 31.40 37.26 30.90 34.19 39.19 54.31 54.77
## 2012 50.96 29.15 30.68 32.32 28.91 32.49 30.62 22.97 30.35 42.02 69.52 89.48
## 2013 43.37 26.24 30.32 28.08 27.23 26.62 31.09 28.77 31.21 34.20 31.45 33.52
## 2014 37.59 29.88 25.03 28.22 24.86 26.90 32.67 26.24 27.52 36.12 40.19 68.16
## 2015 55.71 32.85 26.78 32.86 25.04 26.06 28.53 28.59 25.84 38.56 32.80 36.85
## 2016 80.42 45.16 30.07 36.56 29.99 26.89 33.74 28.83 28.76 35.13 34.85 29.63
## 2017 37.62 38.70 32.24 40.18 27.83 25.33 36.67 29.26 24.47 39.89 41.56 54.21
## 2018 58.05 37.15 40.67 45.60 29.22 32.55 35.11 33.54 31.50 45.30 48.40 37.23
## 2019 45.16 36.15 33.43 41.42 29.20 32.59 40.49 33.55 35.32
```

Computing the possible Holt-Winters' multiplicative seasonality models:

Let's compute the possible models with seasonality as addictive & multiplicative and check their accuracy scores-

```
# Evaluating HW models and comparing them based on 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))

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

               hw_MASE[i]= accuracy(hw(mortalitymonthly,
                            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")

# Checking accuracy scores

arrange(accuracy_hw,MASE)
```

```
##         seasonal damped      MASE      AIC      BIC
## 1 multiplicative   TRUE 0.6827737 1023.672 1073.391
## 2 multiplicative  FALSE 0.6979624 1020.011 1066.968
## 3       additive  FALSE 0.7083694 1070.551 1117.508
## 4       additive   TRUE 0.7247052 1070.795 1120.515
```

Based on the above results, it is clear that Holt-Winter's multiplicative seasonality model gives the best accuracy scores in terms of MASE and AIC. Let's evaluate all the possible Holt-Winter's multiplicative seasonality models as they are better than additive models in terms of accuracy measurements.

Fitting Holt-Winter's multiplicative seasonality models:

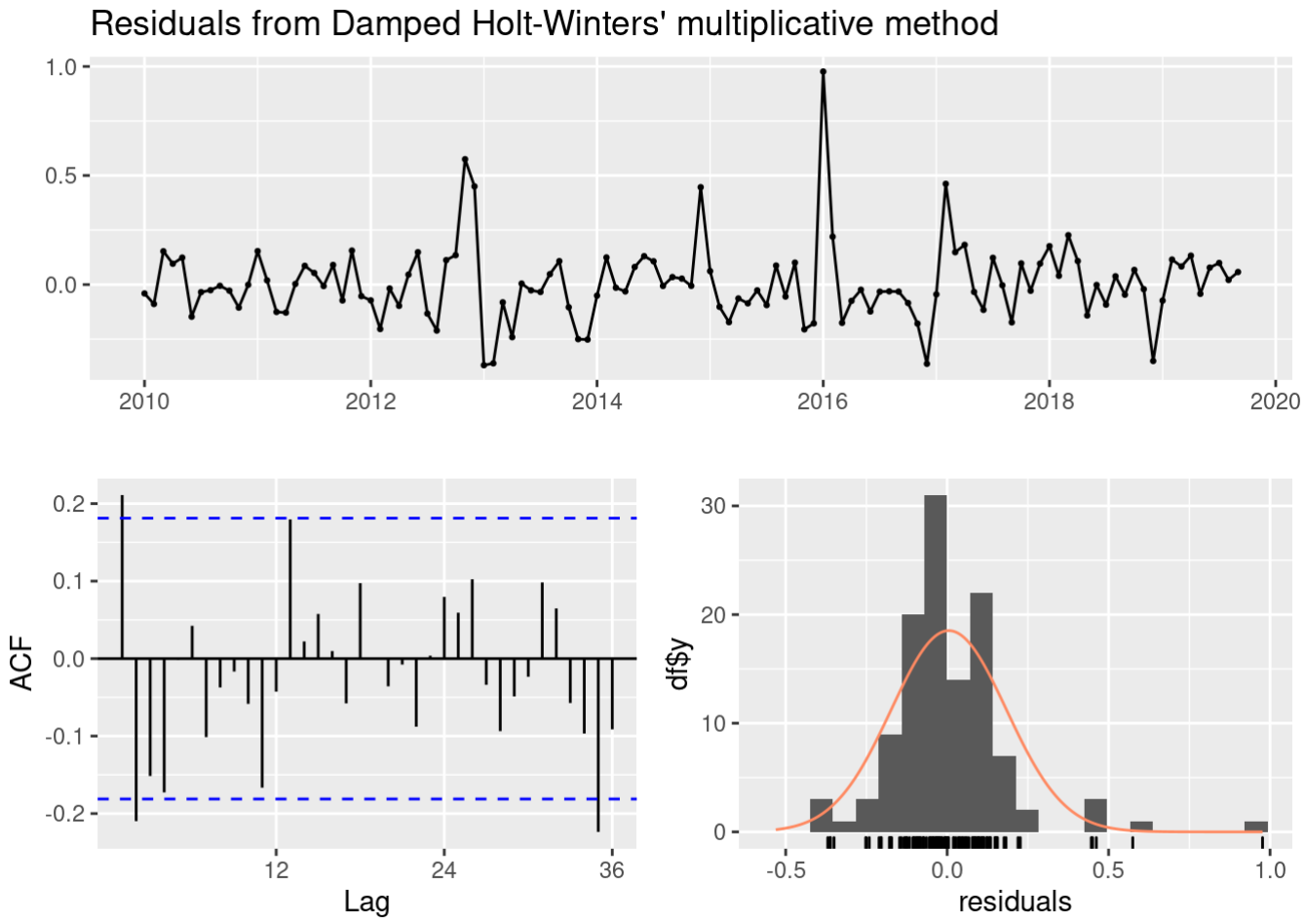
As per the previous results, let's evaluate Holt-Winter's multiplicative seasonality models with exponential components & damped components individually and together-

Fitting Holt-Winter's multiplicative seasonality model with only damped component:

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

# Checking summary & residual analysis

summary(hw1$model)
```

```
## Damped Holt-Winters' multiplicative method
##
## Call:
##  hw(y = mortalitymonthly, seasonal = "multiplicative", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.3414
##     beta  = 1e-04
##     gamma = 1e-04
##     phi   = 0.9735
##
##   Initial states:
##     l = 37.7671
##     b = 0.1035
##     s = 1.4191 1.2156 1.0678 0.818 0.8106 0.9399
##            0.7921 0.7871 0.987 0.8567 0.9355 1.3707
##
##   sigma:  0.1925
##
##      AIC     AICc      BIC
## 1023.672 1030.651 1073.391
##
## Training set error measures:
##                       ME     RMSE      MAE       MPE    MAPE      MASE      ACF1
## Training set 0.05352148 7.685394 4.706196 -2.313471 12.0604 0.6827737 0.1814885
```

```
checkresiduals(hw1)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt-Winters' multiplicative method
## Q* = 31.191, df = 6, p-value = 2.331e-05
##
## Model df: 17.   Total lags used: 23
```

From the above summary and residuals plots of hw1 model we can generate following insights-

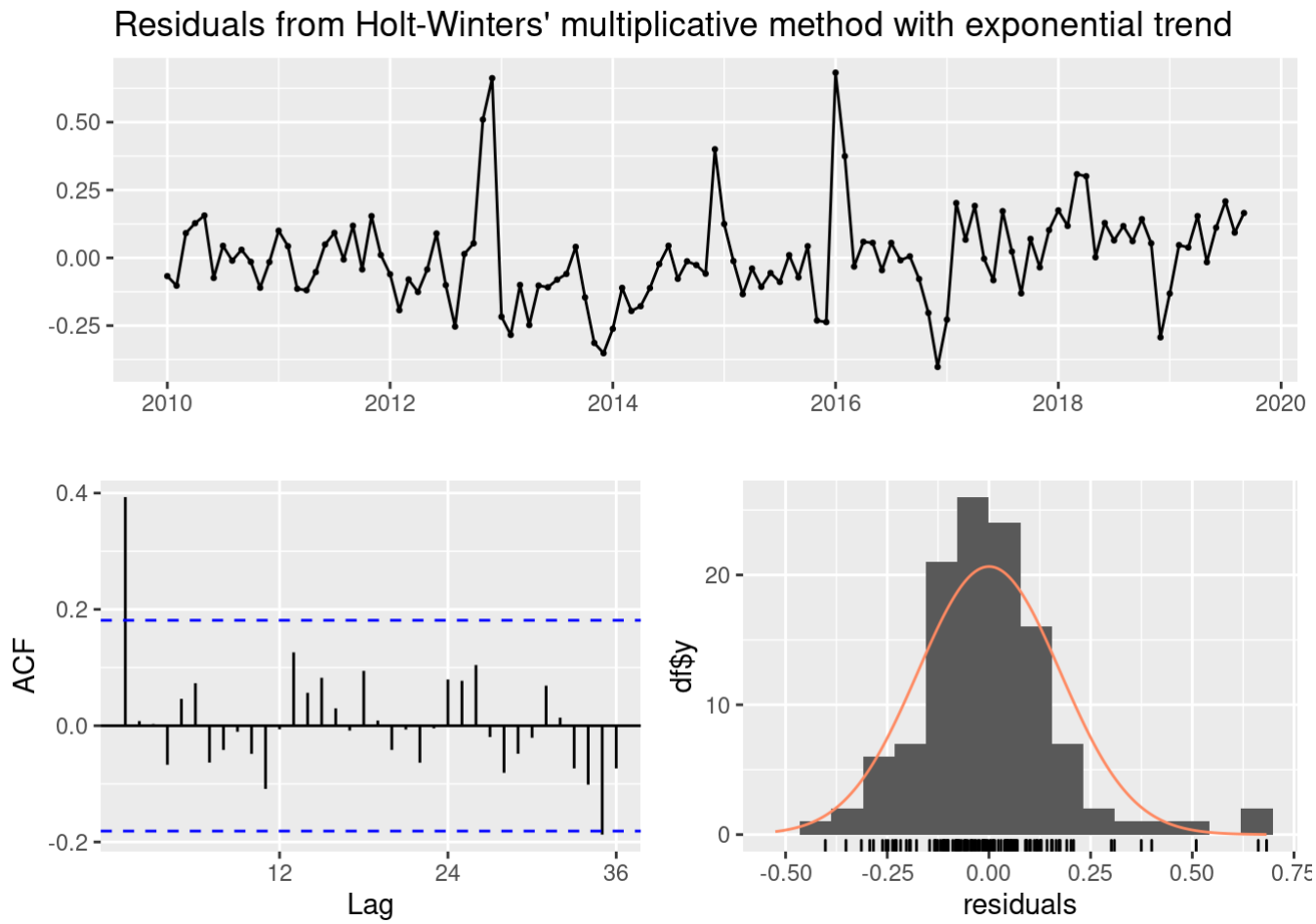
1. The MASE value of the model 0.6828 and other accuracy measures like AIC, AICc and BIC are in the range of 1020-1075.
2. The smoothing parameters are: alpha = 0.3414 ,beta = 1e-04 ,gamma = 1e-04 and phi = 0.9735.
3. As per the Ljung-Box test, p-value is less than 5% level of significance which indicates that autocorrelation is present in the residuals.
4. There are only three significant lags in the ACF plot which is quite a good result in comparison to previous models.
5. Normality problems are present in the model as residuals do not seem to follow normal distribution.

Fitting Holt-Winters' multiplicative seasonality model with exponential element:

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

# Checking summary & residual analysis

summary(hw2$model)
```

```
## Holt-Winters' multiplicative method with exponential trend
##
## Call:
##  hw(y = mortalitymonthly, seasonal = "multiplicative", exponential = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.0433
##     beta  = 1e-04
##     gamma = 1e-04
##
##   Initial states:
##     l = 37.9491
##     b = 1.0001
##     s = 1.4063 1.2298 1.0679 0.8017 0.8152 0.8983
##            0.7898 0.7986 0.9726 0.874 0.9392 1.4066
##
##   sigma:  0.1877
##
##      AIC     AICc      BIC
## 1018.193 1024.375 1065.150
##
## Training set error measures:
##                       ME     RMSE      MAE       MPE     MAPE     MASE
## Training set -0.03424976 7.723558 4.990229 -2.879857 12.86428 0.723981
##                   ACF1
## Training set 0.3494741
```

```
checkresiduals(hw2)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' multiplicative method with exponential trend
## Q* = 28.391, df = 7, p-value = 0.0001868
##
## Model df: 16.   Total lags used: 23
```

From the above summary and residuals plots of hw2 model we can generate following insights-

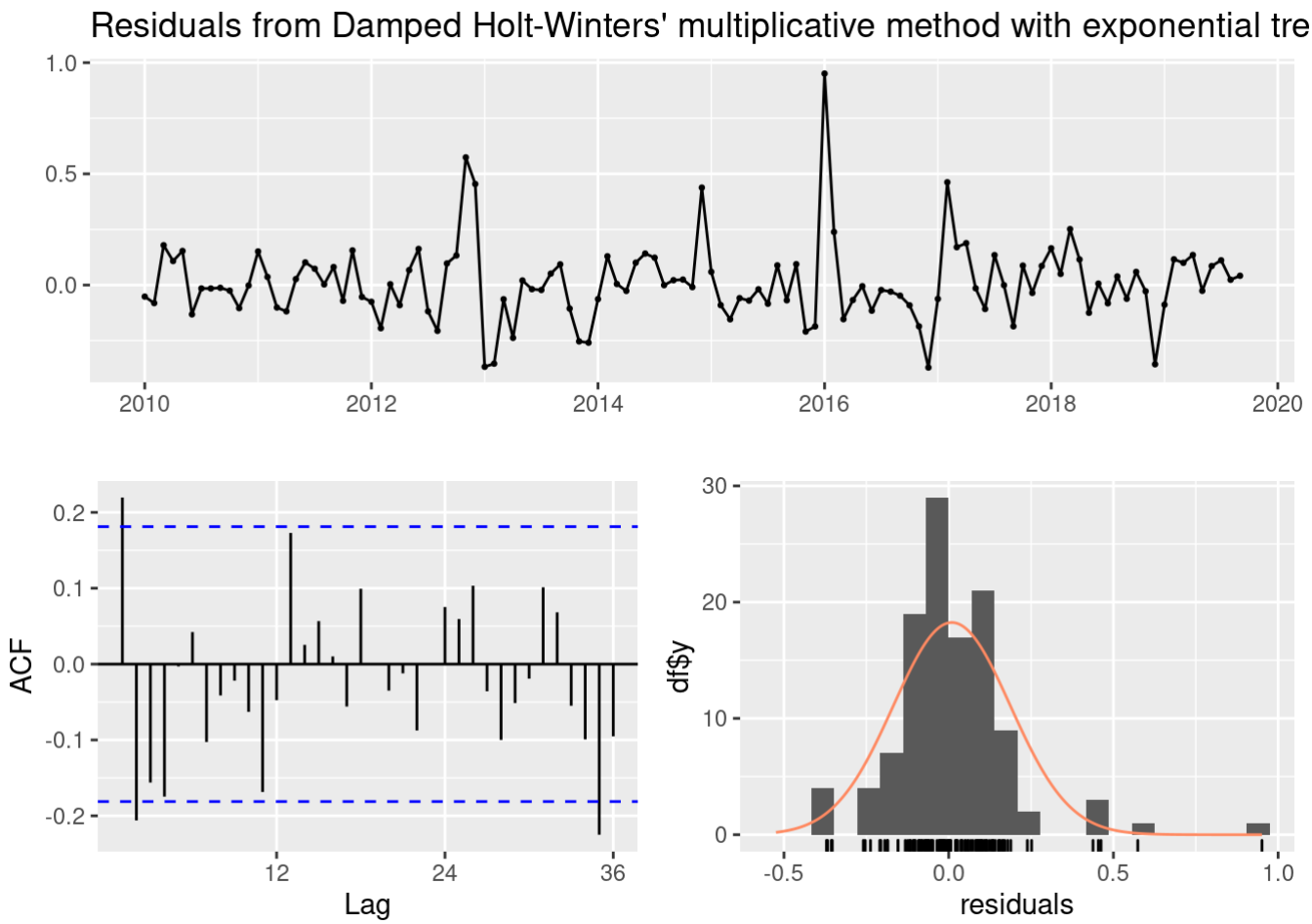
1. The MASE value of the model 0.723 and other accuracy measures like AIC, AICc and BIC are in the range of 1018-1065.
2. The smoothing parameters are: alpha = 0.0433, beta= 1e-04 and gamma = 1e-04.
3. As per the Ljung-Box test, p-value is less than 5% level of significance which indicates that autocorrelation is present in the residuals.
4. There is only one significant lags in the ACF plot which supports Ljung-Box test results.
5. Normality problems are present in the model as residuals do not seem to follow normal distribution as histogram is slightly right skewed.

Fitting Holt-Winters' multiplicative seasonality model with exponential and damped element:

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

# Checking summary & residual analysis

summary(hw3$model)
```

```
## Damped Holt-Winters' multiplicative method with exponential trend
##
## Call:
##  hw(y = mortalitymonthly, seasonal = "multiplicative", damped = TRUE,
##
##  Call:
##      exponential = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.3291
##     beta  = 1e-04
##     gamma = 1e-04
##     phi   = 0.979
##
##   Initial states:
##     l = 37.8041
##     b = 0.9981
##     s = 1.4344 1.2232 1.0712 0.8236 0.8022 0.9245
##            0.7831 0.7751 0.9853 0.8476 0.9379 1.3919
##
##   sigma:  0.192
##
##      AIC     AICc      BIC
## 1022.022 1029.002 1071.742
##
## Training set error measures:
##                     ME     RMSE     MAE       MPE     MAPE    MASE      ACF1
## Training set 0.1774934 7.680161 4.70369 -1.862828 12.00987 0.68241 0.1875796
```

```
checkresiduals(hw3)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt-Winters' multiplicative method with exponential trend
## Q* = 31.733, df = 6, p-value = 1.835e-05
##
## Model df: 17.   Total lags used: 23
```

From the above summary and residuals plots of hw3 model we can generate following insights-

1. The MASE value of the model 0.6824 and other accuracy measures like AIC, AICc and BIC are in the range of 1022-1072.
2. The smoothing parameters are: alpha = 0.3291, beta = 1e-04, gamma = 1e-04 and phi = 0.979.
3. As per the Ljung-Box test, p-value is less than 5% level of significance which indicates that autocorrelation is there in the residuals.
4. There are 3 slightly significant lags in the ACF plot which supports Ljung-Box test results.
5. Normality problems are present in the model as residuals do not seem to follow normal distribution.

Evaluating best Holt-Winter's model:

Out of the three models fitted, **Holt-Winter's model with damped component alone and Holt-Winter's model with damped component & exponential component** are the two best models computed in terms of MASE accuracy.

Modelling with State Space Models (ETS)

The parametric structure definition can be more flexible using state-space models. In the ETS model, the first term represents trend, the second term represents seasonality, and the third term represents error. Let's try to adapt appropriate models for the sr series, taking in mind that the series has a strong seasonal component.

Let's compute different variation of ETS models and compare them based on accuracy score-

```
# Computing different variation of ETS models for mortality

variations <- c("AAA", "MAA", "MAM", "MMM")
damped <- c(TRUE,FALSE)
ets_model <- expand.grid(variations, damped)
ets_aic <- array(NA, 8)
ets_bic <- array(NA,8)
ets_mase <- array(NA,8)
model <- array(NA, dim=c(8,2))
for (i in 1:8){
  ets_var<- ets(mortalitymonthly, model = toString(ets_model[i, 1]), damped = ets_model[i,2])
  ets_aic[i] <- ets_var$aic
  ets_bic[i] <- ets_var$bic
  ets_mase[i] <- accuracy(ets_var)[6]
  model[i,1] <- toString(ets_model[i,1])
  model[i,2] <- ets_model[i,2]
}

# Appending accuracy measurements

ets_cal <- data.frame(model, ets_mase, ets_aic, ets_bic)
ets_cal$X2 <- factor(ets_cal$X2, levels = c(T,F), labels = c("Damped","N"))
ets_cal <- unite(ets_cal, "ETS_Model", c("X1","X2"))
colnames(ets_cal) <- c("ETS_Model", "MASE", "AIC", "BIC")

accuracy_ets <- arrange(ets_cal, MASE)

# View all the computed models and compare their accuracy score

accuracy_ets
```

```
##   ETS_Model      MASE      AIC      BIC
## 1    MMM_NA 0.7012264 1017.597 1067.316
## 2    MAA_NA 0.7058651 1018.335 1068.054
## 3    MMM_NA 0.7081910 1013.029 1059.986
## 4    AAA_NA 0.7083694 1070.551 1117.508
## 5    MAA_NA 0.7091741 1017.251 1064.207
## 6    MAM_NA 0.7106809 1015.094 1064.813
## 7    AAA_NA 0.7247052 1070.795 1120.515
## 8    MAM_NA 0.7864997 1026.575 1073.532
```

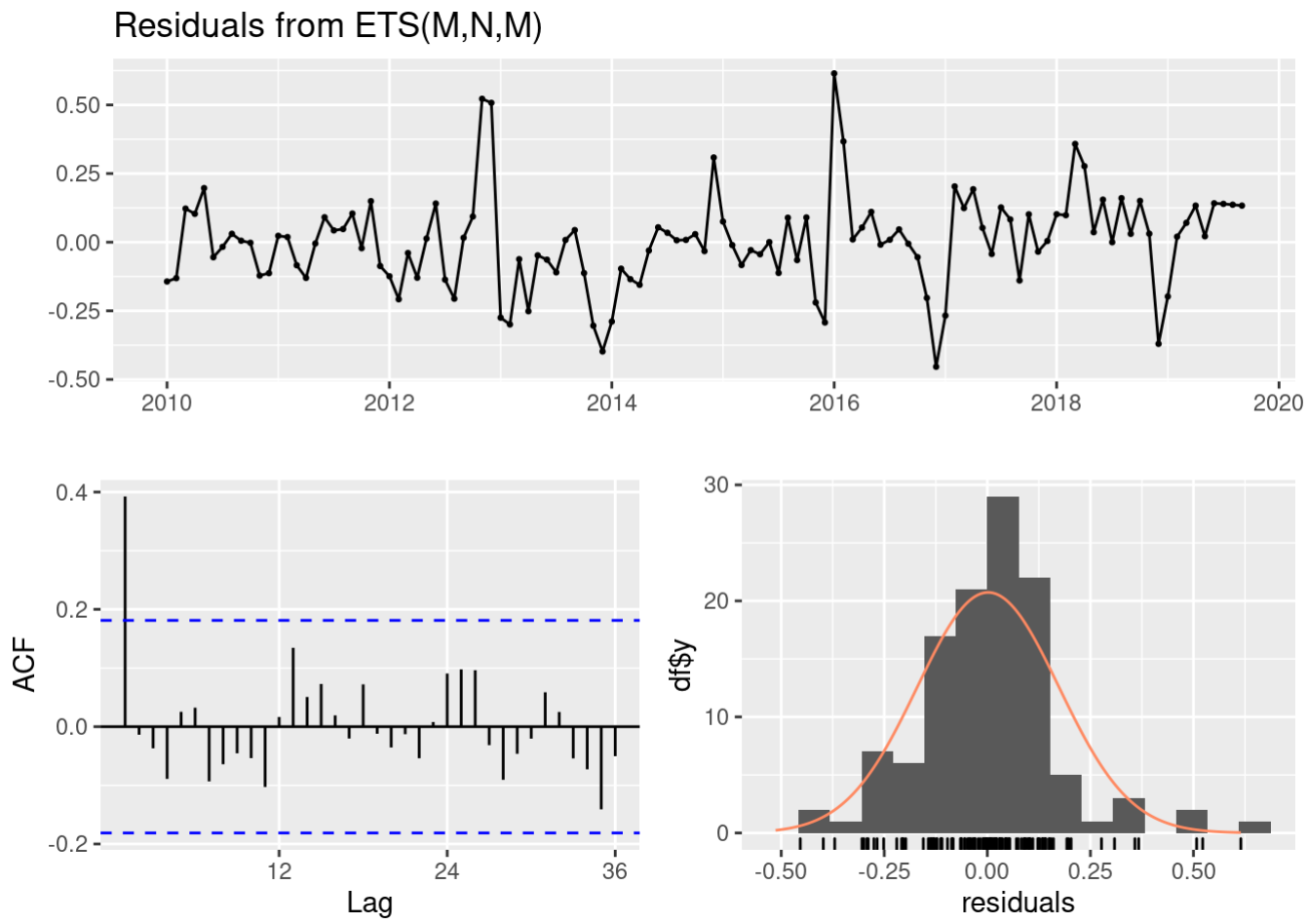
All the above models have a worse score in terms of MASE in comparison to HW models fitted before. Let's try and compute another model using auto ETS.

Fiting Auto-ETS model:

```
# Fitting ets model

ets_model <- ets(mortalitymonthly)
summary(ets_model)
```

```
## ETS(M,N,M)
##
## Call:
##  ets(y = mortalitymonthly)
##
##   Smoothing parameters:
##     alpha = 0.0653
##     gamma = 8e-04
##
##   Initial states:
##     l = 39.0751
##     s = 1.5243 1.2126 1.0267 0.7988 0.7631 0.9272
##            0.7516 0.7536 0.9723 0.8344 0.9483 1.4871
##
##   sigma:  0.1824
##
##      AIC     AICc      BIC
## 1009.427 1014.179 1050.859
##
## Training set error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE     ACF1
## Training set -0.3137847 7.938851 5.046938 -2.891975 12.99859 0.7322084 0.34495
```

```
checkresiduals(ets_model)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,M)
## Q* = 28.532, df = 9, p-value = 0.000777
##
## Model df: 14.   Total lags used: 23
```

From the above summary and residuals plots of ets_model we can generate following insights-

1. The MASE value of the model 0.7322 and other accuracy measures like AIC, AICc and BIC are in the range of 1009-1050. This is worse than previous models fitted. Hence, this model is a poor fit model.
2. The smoothing parameters are: alpha = 0.0653 and gamma = 8e-04.
3. As per the Ljung-Box test, p-value is less than 5% significance level which indicates that serial correlation is there in the residuals.
4. The residuals are randomly distributed and in the ACF plot there is one significant lag which supports Ljung-Box test results.
5. Normality problems are present in the model as residuals do not seem to follow normal distribution.

Summary Table Task 1

Generating a table for all models fitted so far-

Models fitted from DLM and dynamic linear method

```
mase_dlm <- rbind(cal_mase,cal_mase2)

arrange(mase_dlm,MASE)
```

```
##                n      MASE
## lm(dynlm1)   507 0.8558330
## lm(dynlm3)   507 0.8558596
## lm(dynlm7)   506 0.8678083
## lm(dynlm6)   506 0.8710927
## lm(dynlm8)   506 0.8720844
## lm(dynlm5)   506 0.8724418
## lm(dynlm2)   506 0.8796086
## model.reg    503 0.8823001
## model.koyck5 507 0.9194400
## lm(dynlm4)   507 0.9396163
## model.finite 498 1.0877112
## model.poly13 498 1.0916760
```

Here, it is clearly seen that **dynlm1** model has the best accuracy score in terms of MASE value (0.8558330) and is the best regression model fitted for mortality series.

Models fitted from Exponential Smoothing

```
# Accuracy score for all models with damped and seasonality

arrange(accuracy_hw,MASE)
```

```
##         seasonal damped      MASE      AIC      BIC
## 1 multiplicative   TRUE 0.6827737 1023.672 1073.391
## 2 multiplicative  FALSE 0.6979624 1020.011 1066.968
## 3       additive  FALSE 0.7083694 1070.551 1117.508
## 4       additive   TRUE 0.7247052 1070.795 1120.515
```

```
# Accuracy computed for fitted models
accuracy(hw2)
```

```
##                       ME     RMSE      MAE       MPE     MAPE     MASE
## Training set -0.03424976 7.723558 4.990229 -2.879857 12.86428 0.723981
##                    ACF1
## Training set 0.3494741
```

```
accuracy(hw3)
```

```
##                     ME     RMSE     MAE       MPE     MAPE    MASE      ACF1
## Training set 0.1774934 7.680161 4.70369 -1.862828 12.00987 0.68241 0.1875796
```

Here, the best model is **Holt Winter's model with both damped and exponential component** with MASE value as 0.682410.

Models fitted from ETS

```
# Accuracy score for all ETS models possible

arrange(accuracy_ets,MASE)
```

```
##   ETS_Model      MASE      AIC      BIC
## 1    MMM_NA 0.7012264 1017.597 1067.316
## 2    MAA_NA 0.7058651 1018.335 1068.054
## 3    MMM_NA 0.7081910 1013.029 1059.986
## 4    AAA_NA 0.7083694 1070.551 1117.508
## 5    MAA_NA 0.7091741 1017.251 1064.207
## 6    MAM_NA 0.7106809 1015.094 1064.813
## 7    AAA_NA 0.7247052 1070.795 1120.515
## 8    MAM_NA 0.7864997 1026.575 1073.532
```

```
#Accuracy for auto ETS model fitted

accuracy(ets_model)
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE    ACF1
## Training set -0.3137847 7.938851 5.046938 -2.891975 12.99859 0.7322084 0.34495
```

Out of all the ETS models fitted, **MMM_NA model** is the best model in terms of MASE accuracy score. The MASE accuracy score is: 0.7012264.

Best Model:

```
final_mase <- data.frame(Model = c("dynlm1 dynamic linear model","Holt-Winter's ES model","MM
M_NA ETS model"), MASE = c(0.8558330,0.682410,0.7012264))

arrange(final_mase, MASE)
```

```
##                         Model      MASE
## 1      Holt-Winter's ES model 0.6824100
## 2            MMM_NA ETS model 0.7012264
## 3 dynlm1 dynamic linear model 0.8558330
```

From the above result, the best model to fit the monthly mortality series is: Holt-Winter's ES model with exponential and damped component while for the weekly series it is dynlm1 dynamic linear model. Forecasting will be performed on both the weekly and monthly mor series using respective best models computed for them based on MASE, R2 & other accuracy measures.

Forecasting

Generating Monthly Forecasting using Damped & exponential HW multiplicative model:

The following are the mortality point prediction values for the next four weeks (1 month), along with their 95 percent confidence intervals using the best model-

```
# Fitting model

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

# Generating forecasts

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

predict_mor <- ts.intersect(ts(lower_limit, start = c(2019,1), frequency = 12), ts(fitting,st
art = c(2019,1), frequency = 12), ts(upper_limit,start = c(2019,1), frequency = 12))

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

# Displaying forecasts

predict_mor
```

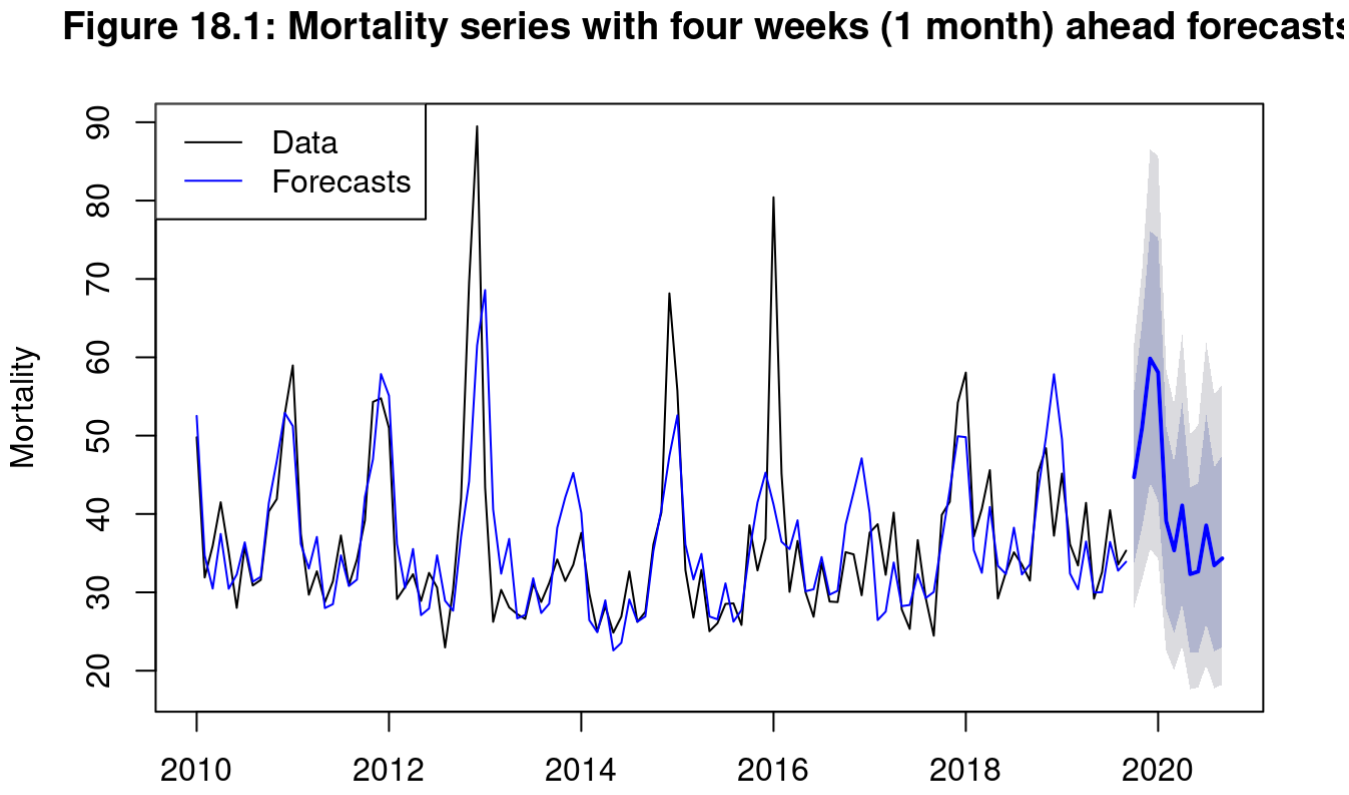
```
##          Lower Limit Prediction Points Upper limit
## Jan 2019    28.09490          44.69953    61.74898
## Feb 2019    31.66620          51.03637    71.11326
## Mar 2019    35.56330          59.84199    86.51699
## Apr 2019    34.30598          58.06964    85.67370
## May 2019    22.56765          39.12441    58.39489
## Jun 2019    20.05160          35.35513    53.93900
## Jul 2019    23.25291          41.09328    63.04171
## Aug 2019    17.63864          32.32433    50.10605
## Sep 2019    17.85774          32.65697    51.33413
## Oct 2019    20.72547          38.54653    61.90032
## Nov 2019    17.77951          33.44405    55.25885
## Dec 2019    18.22068          34.33628    56.50636
```

Using Damped Holt-Winters' multiplicative seasonality model with damped & exponential elements, projections for the next four weeks (1 month) are shown above. The four weeks ahead prediction is displayed here. As we can see, the month of October 2019, gives the four weeks ahead forecast which is 38.54653 and the 95% confidence interval for the same is (20.82952 & 62.68608).

However, the 95 percent confidence intervals for the estimates using the selected technique are extremely wide, suggesting that the forecasts are not completely accurate.

Plotting the forecast generated:

```
plot(fit.forecast, fcol = "white", main = "Figure 18.1: Mortality series with four weeks (1 m
onth) ahead forecasts", ylab = "Mortality")
lines(fitted(fit.forecast), col = "blue")
lines(fit.forecast$mean, col = "blue", lwd = 2)
legend("topleft", lty = 1, col = c("black", "blue"), c("Data", "Forecasts"))
```



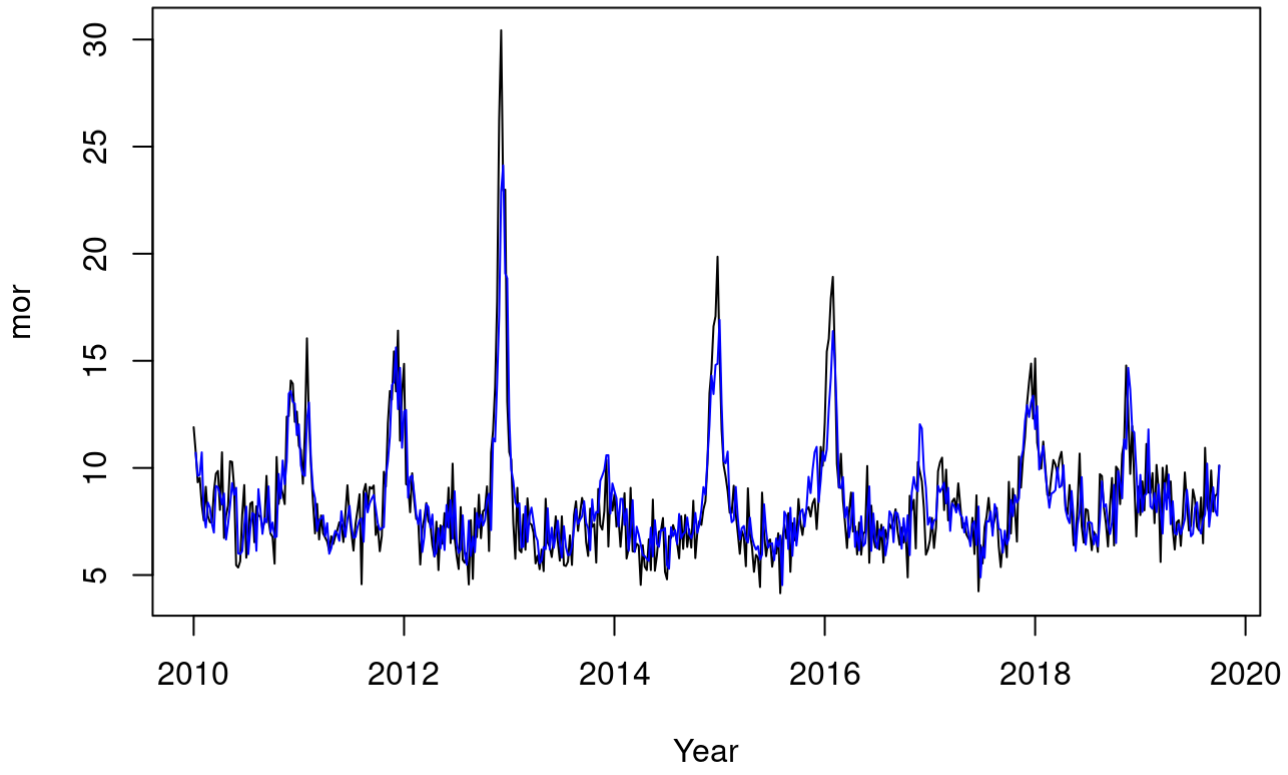
The forecast generated are graphically shown above. The model is not able to completely follow the original data and is missing big intervention points. However, Damped & exponential HW multiplicative model gives the predictions for monthly data (that was previously weekly). Hence, the predictions generated are not reliable.

Generating Weekly Forecasts using Dynamic linear model dynlm1:

Since the original series is weekly and we computed that dynlm1 was the best model in terms of different accuracy measures. Hence, using the best model(dynlm1) to predict the four weeks ahead forecasts-

```
par(mfrow=c(1,1))
plot(mor,ylab='mor',xlab='Year',main = "Figure 18.2: Fitting the dynlm1 model")
lines(dynlm1$fitted.values,col="blue")
```

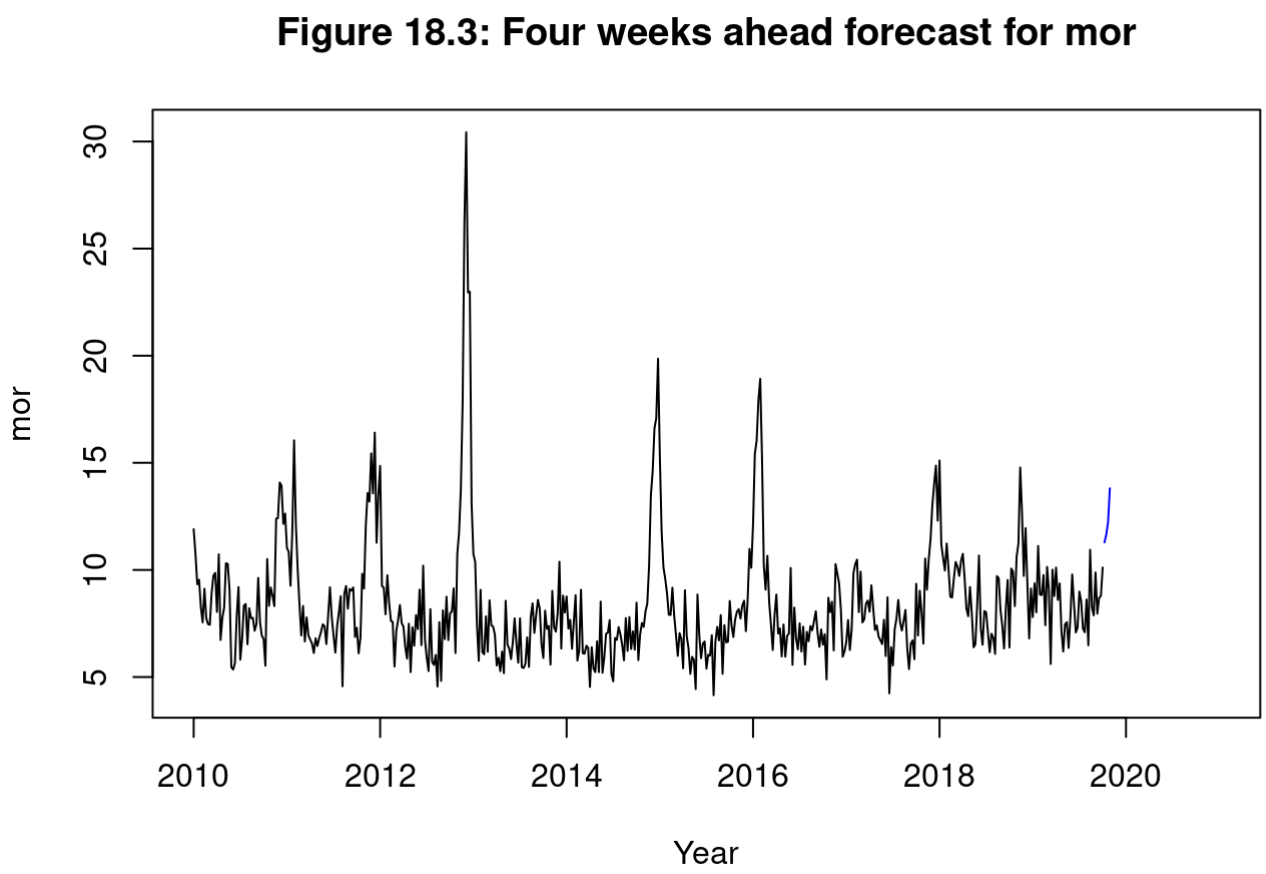
Figure 18.2: Fitting the dynlm1 model



As seen from above plot, the dynlm1 model is a better fit for the model in comparison to Holt Winter. However, it is not able to capture most of the seasonality in the mor series. Let’s generate the four weeks ahead forecasts using this model.

Plotting forecast for next four weeks:

```
q = 4
n = nrow(dynlm1$model)
m.frc = array(NA , (n + q))
m.frc[1:n] = Y.t[2:length(Y.t)]
trend = array(NA,q)
trend.start = dynlm1$model[n,"trend(Y.t)"]
trend = seq(trend.start , trend.start + q/12, 1/12)

for(i in 1:q){
weeks =array(0,51)
weeks[(i-12)%%52] = 1
data.new =c(1,m.frc[n-1+i],1,trend[i],weeks)
m.frc[n+i] = as.vector(dynlm1$coefficients) %*% data.new
}

plot(mor,
ylab='mor',xlab='Year',xlim = c(2010,2021),
main = "Figure 18.3: Four weeks ahead forecast for mor")
lines(ts(m.frc[(n+1):(n+q)],start=c(2019,41),frequency = 52),col="blue")
```



As per the above plot, we can say that, the mortality is going to be low in the next four weeks as indicated by the red line. This prediction is more reliable than the previous model as it is a better fit in comparison to it.

Forecast values:

```
forecastdynlm1 <- m.frc[(n+1):(n+q)]

round(forecastdynlm1,2)
```

```
## [1] 11.29 11.65 12.23 13.81
```

The prediction values for the next four weeks will lie between 11-13. Thus, the mortality rate is going to be low in the next four weeks than previous weeks.

Conclusion

The task 1 analysis was successfully done on all the five series present in mor.csv file. The five series were analyzed and different modelling methods(DLM,koyck,polyDLM,ARDL,dynlm,ES,State space models) were used to fit models. The weekly mortality data was converted to monthly for fitting the ES and State Space models as the mortality series had seasonality. Out of all those models, Damped & exponential HW multiplicative model was the best model fitted in terms of R2, AIC, BIC and MASE for the monthly series. For the weekly series, the dynlm1 model (dynamic linear model) was computed as the best model. Forecasting was performed on both the weekly and monthly mor series using HW & dynamic model respectively. The predictions generated by dynamic linear model were more reliable & accurate in comparison to HW model as the dynlm1 was a better fit and weekly values were predicted on the original series using it.

Task 2 - Time series analysis & forecast of FFD series

Data Description

Hudson & Keatley (2021) studied if climatic conditions such as rainfall (rain), temperature (temp), radiation level (rad), and relative humidity affect the day of a species' first blooming (first flowering day, FFD, a number between 1 -365). (RH). Hudson and Keatley's research focuses on the impact of long-term climate on the FFD of 81 plant species from 1984 to 2014.

The data 'FFD.csv' focuses on one species (of the 81) and contains 5 time series, the FFD time series of the given plant species and the contemporaneous yearly averaged climate variables measured from 1984 – 2014 (31 years). The climate variables are- temperature, rainfall,relhumidity and radiation.

Objective & Methodology

The main objective of task 2 is to analyze the FFD and give 4 years ahead forecast for FFD. Following steps will be performed-

1. Firstly, necessary data prepossessing steps will be performed on all the five series. The data file 'FFD.csv' will be checked for missing/unique/unknown values and will be dealt with accordingly. Each series from 'FFD.csv' will also be converted to time-series to demonstrate yearly values for all the five series to carry out a time series analysis.
2. The covariate file will be loaded and will be used during forecasting.
3. The next step will be to check and understand the nature/characteristics of all the time series by plotting respective time series plots and plotting them together. All the five series will also be checked for stationarity through ACF ,PACF plots and Dicker-Fuller Unit tests.
4. The correlation between each series will be checked by plotting a correlation matrix for all of them.
5. Further, different distributed lag models will be fitted on the series.
6. Different type of DLMs, dynlms, exponential smoothing & state-space models will be fitted depending upon the results generated from previous steps.

7. Forecasting will be done on all best models generated using different methods based on accuracy measures like R2, F-test, AIC, BIC and forecast generated.
8. The best model will be selected based on various accuracy measures.

Data Prepossessing

Before executing the task 2, we need to perform some necessary data prepossessing steps to make the datasets ready for analysis.

1. Importing dataset into Rstudio:

```
# Importing dataset `FFD' & `covariate x values for task 2`

FFD_series <- read_csv("/cloud/project/FFD .csv")
```

```
## 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.
```

```
task2_covariate <- read_csv("/cloud/project/Covariate 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.
```

```
# Checking they has been loaded successfully

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        93.9   310
## 2  1985        19.3     2.48      14.7        94.9   322
## 3  1986        18.6     2.42      14.5        94.1   306
## 4  1987        19.1     2.32      14.7        94.5   306
## 5  1988        20.4     2.47      14.7        94.1   306
## 6  1989        19.6     2.74      14.8        96.1   293
```

```
head(task2_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 of both series

class(FFD_series)
```

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

```
class(task2_covariate)
```

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

2. Checking for missing and special values:

Checking the datasets for missing values-

```
# Checking for missing values in both series

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
```

As per the above output there are no missing values in the FFD_series, however, there are two missing values in task2_covariate series. We will remove the missing values in task2_covariate dataset.

Now checking for special values-

```
#Checking for special values in both series

is.specialorNA <- function(x){
  if (is.numeric(x)) (is.infinite(x) | is.nan(x) | is.na(x))
}

sapply(FFD_series, function(x) sum( is.specialorNA(x) ))
```

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

```
sapply(task2_covariate, function(x) sum( is.specialorNA(x) ))
```

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

As per the above outputs, there are no missing and special values in FFD_series, however, there are missing/special values of NA in task2_covariate. Let's remove those-

3.A Removing NA values in task2_covariate

```
# 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        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
```

As per above output, missing/NA values have been removed from task2_covariate series.

4. Converting each variable to time series and storing separately:

The five different series will be extracted from FFD_series. Then they will converted to time series and stored in separate datasets.The dataset FFD_series will also be converted to 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 conversion has been successfully done for each series

class(climate)
```

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

```
class(temperature)
```

```
## [1] "ts"
```

```
class(rainfall)
```

```
## [1] "ts"
```

```
class(humidity)
```

```
## [1] "ts"
```

```
class(radiation)
```

```
## [1] "ts"
```

```
class(FFD)
```

```
## [1] "ts"
```

The conversion was successfully done.The FFD_series data was converted to time series and each of the five variables in the FFD_series (except Year) were individually converted to five different series containing time-series data of yearly values of temperature, radiation, rainfall, humidity and FFD respectively.

Understanding Nature of Series

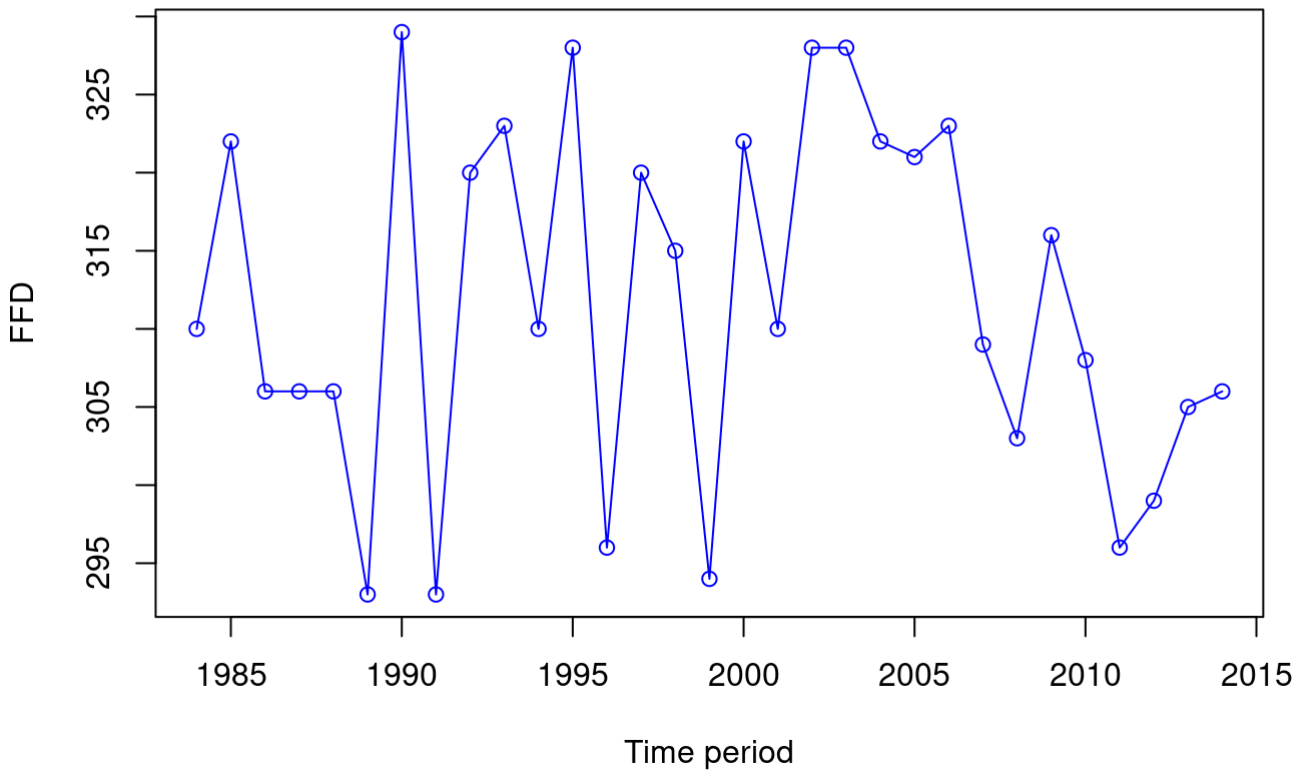
In order to understand the nature of each of the five series, we will plot a time-series plot for each of the series separately and together. We will also calculate correlation of the dataset `climate` .

Time-series plot of FFD:

A time-series plot of the yearly values of FFD was plotted using plot function.

```
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: Time-series plot of change in yearly values of FFD



Following observations were seen from the time series plot of the FFD series-

1.) Trend : There seems to be no presence of a particular trend in the series.

2.) Seasonality : No particular seasonality can be seen in the series.

3.) Changing variance : There is presence of change in variance in the series.

4.) Intervention : There are many intervention points in the series from 1984 to 2000.

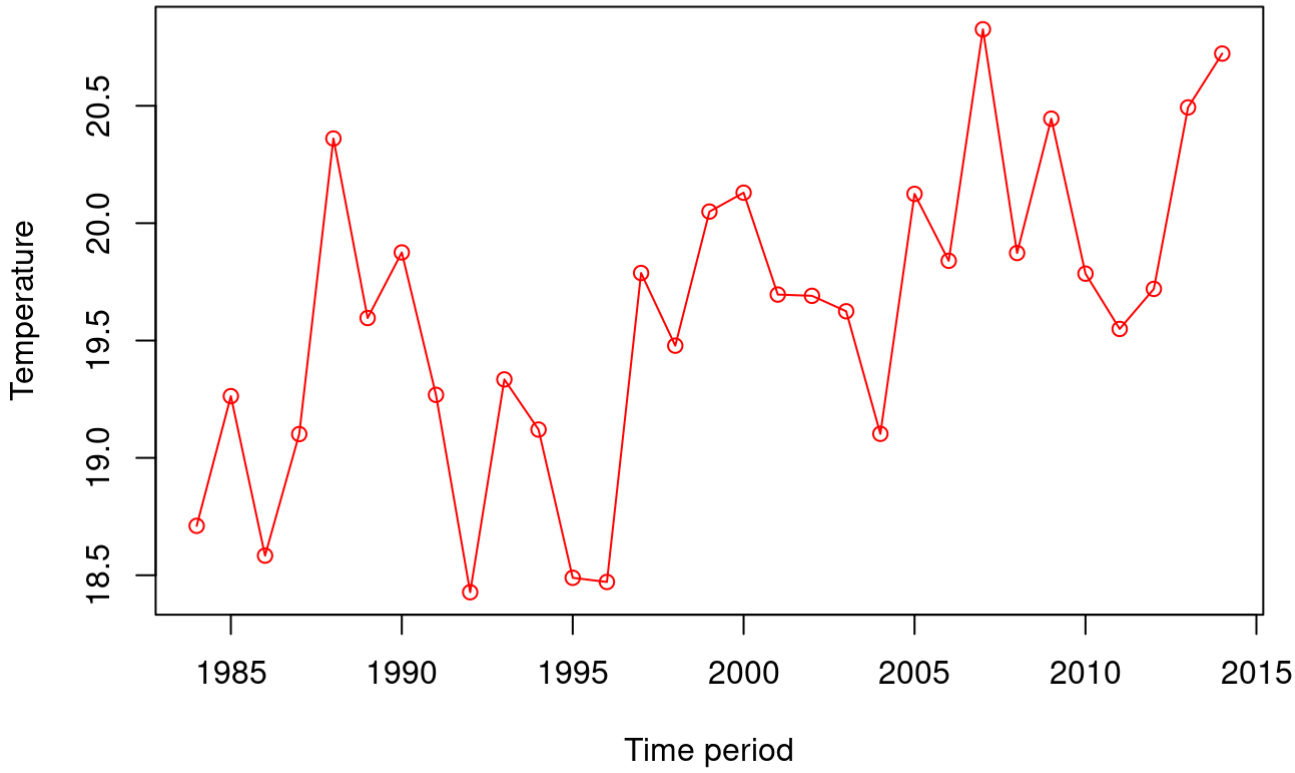
5.) Behavior : Several successive points and fluctuations can be observed along the mean level in the series therefore, a high order AR and MA behavior is seen.

Time-series plot of Temperature:

A time-series plot of the yearly values of temperature was plotted using plot function.

```
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: Time-series plot of change in yearly values of temperature



Following observations were seen from the time series plot of the temperature series-

1.) Trend : There seems to be presence of a downward trend in the series.

2.) Seasonality : No particular seasonality can be seen in the series.

3.) Changing variance : There is presence of change in variance in the series.

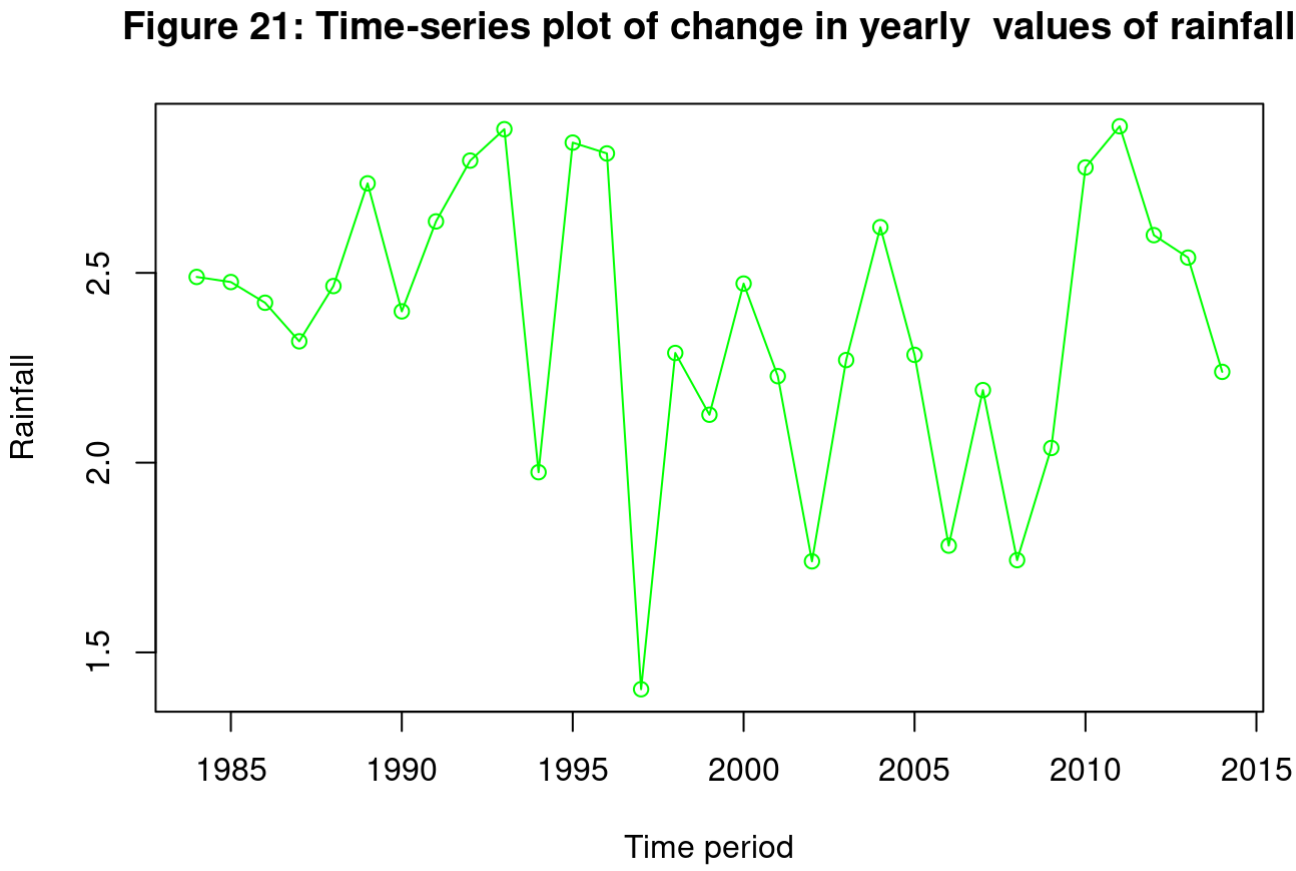
4.) Intervention : There are no particular intervention points in the series.

5.) Behavior :Several successive points and fluctuations can be observed along the mean level in the series therefore, a high order AR and MA behavior is seen.

Time-series plot of Rainfall:

A time-series plot of the yearly values of rainfall was plotted using plot function.

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



Following observations were seen from the time series plot of the rainfall series-

1.) Trend : There seems to be no presence of a particular trend.

2.) Seasonality : No particular seasonality can be seen in the series.

3.) Changing variance : There is presence of change in variance in the series.

4.) Intervention : There is one intervention point around the year 1996-1997.

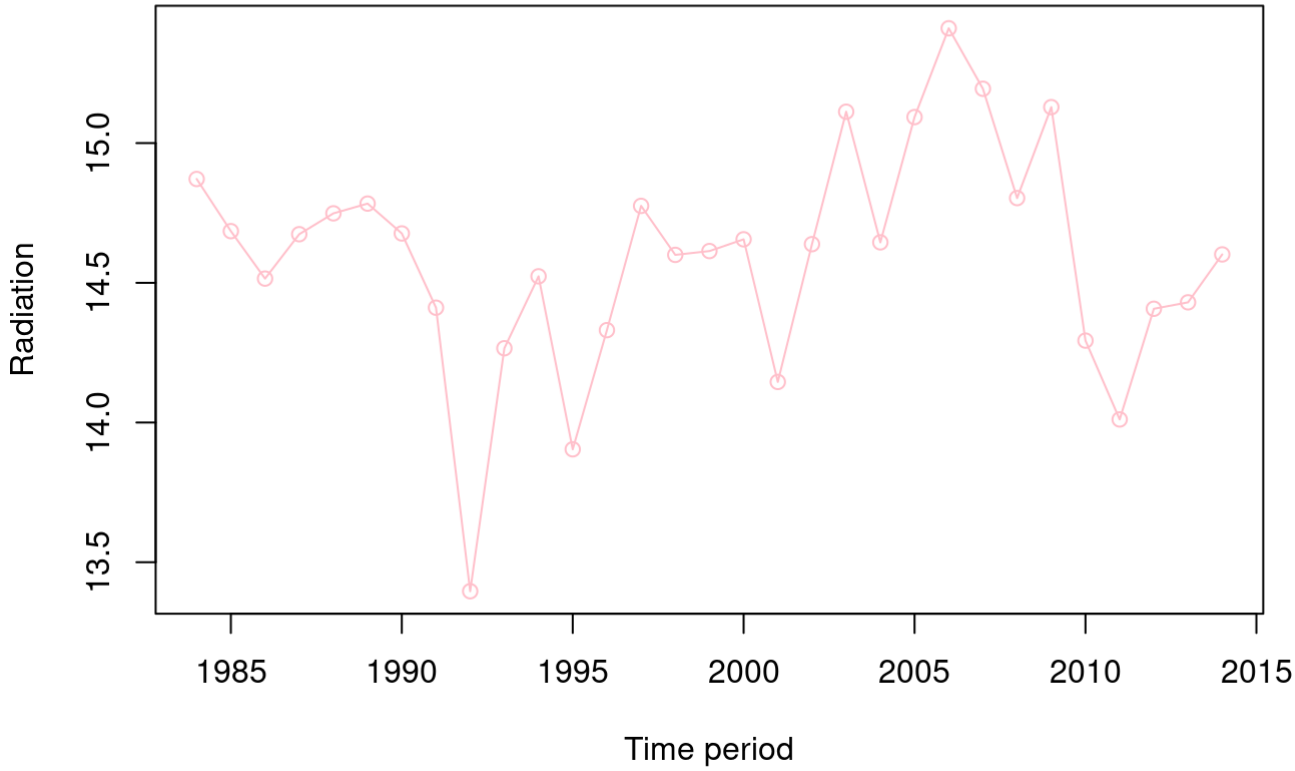
5.) Behavior : Several successive points and fluctuations can be observed along the mean level in the series therefore, a high order AR and MA behavior is seen.

Time-series plot of Radiation:

A time-series plot of the yearly values of radiation was plotted using plot function.

```
plot(radiation, ylab='Radiation', xlab='Time period', type='o', col='pink', main = 'Figure 2
2: Time-series plot of change in yearly  values of radiation')
```

Figure 22: Time-series plot of change in yearly values of radiation



Following observations were seen from the time series plot of the radiation series-

1.) Trend : There seems to be no presence of a particular trend.

2.) Seasonality : No particular seasonality can be seen in the series.

3.) Changing variance : There is presence of change in variance in the series.

4.) Intervention : There is one huge downward intervention point between the years 1990-1995.

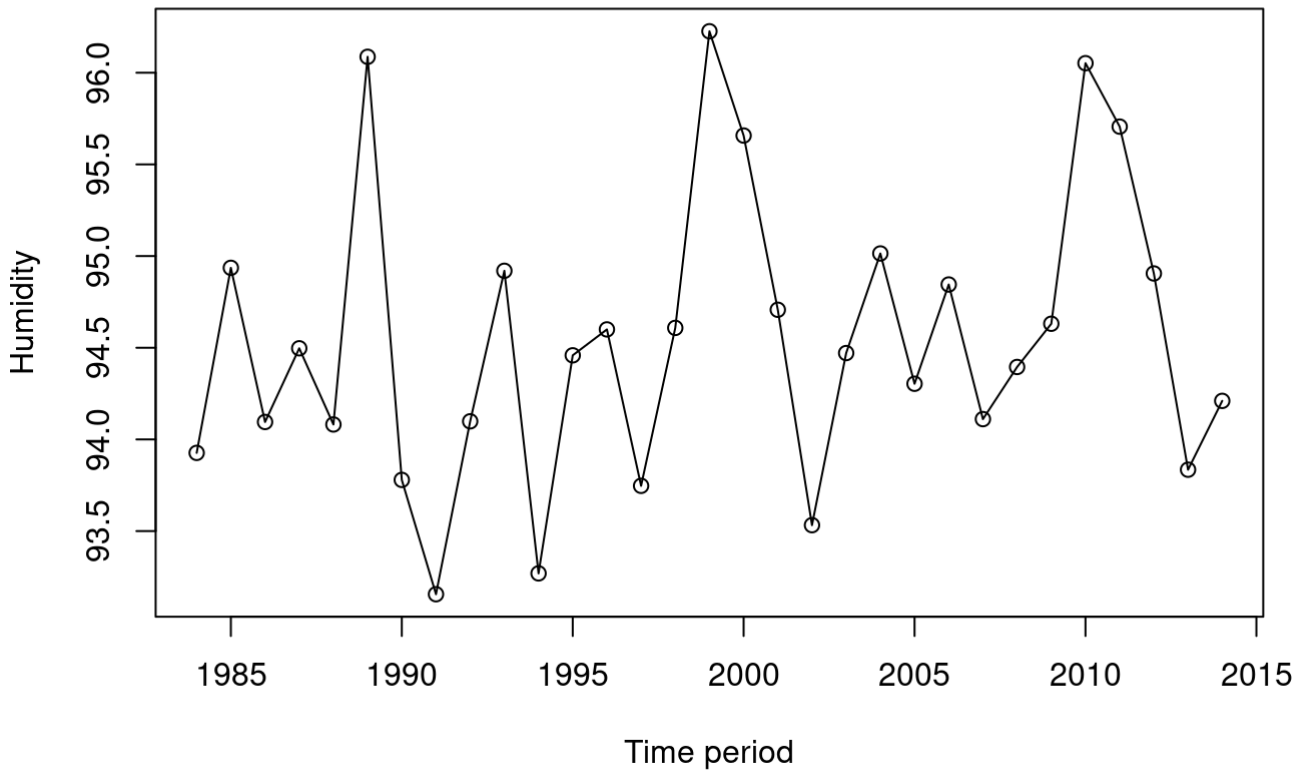
5.) Behavior : Several successive points and fluctuations can be observed along the mean level in the series therefore, a high order AR and MA behavior is seen.

Time-series plot of Humidity:

A time-series plot of the yearly values of humidity was plotted using plot function.

```
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: Time-series plot of change in yearly values of humidity



Following observations were seen from the time series plot of the humidity series-

1.) Trend : There seems to be no presence of a particular trend.

2.) Seasonality : No particular seasonality can be seen in the series.

3.) Changing variance : There is presence of change in variance in the series.

4.) Intervention : There are some mid-level intervention points in the series observed.

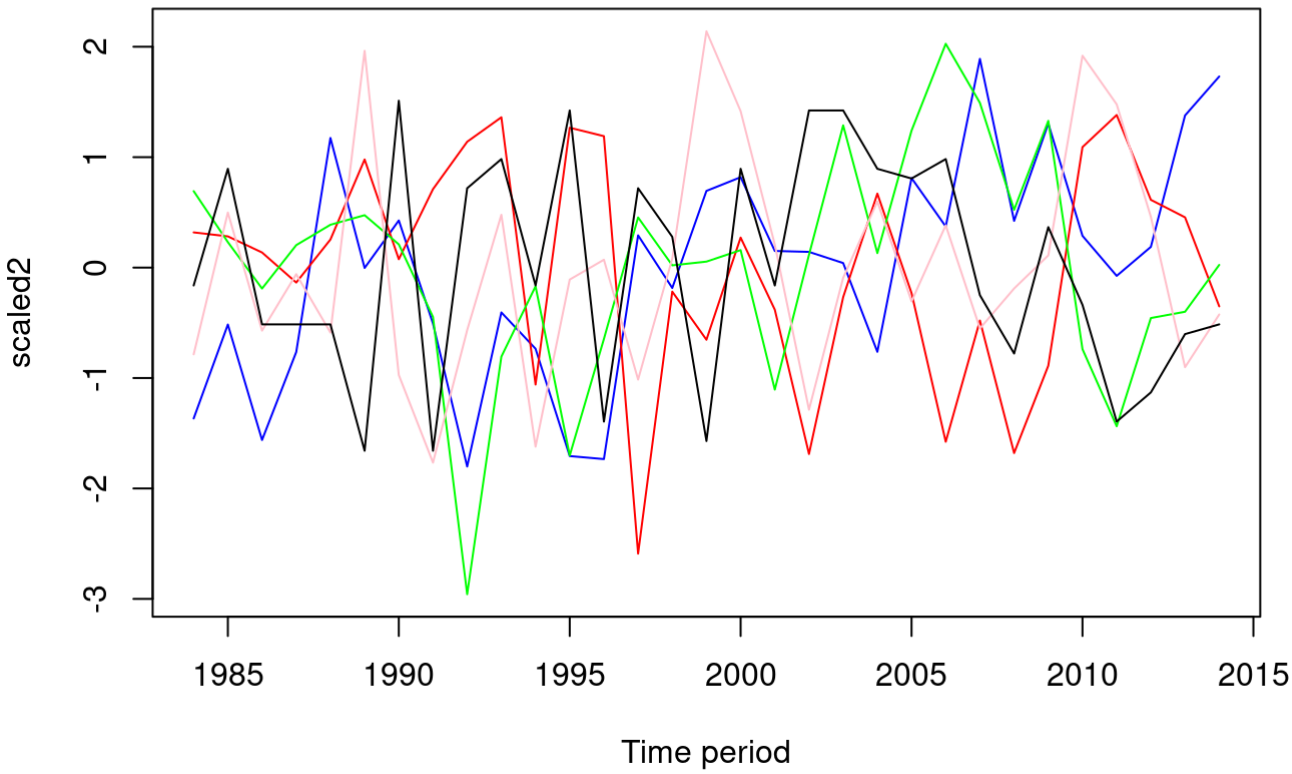
5.) Behavior : Several successive points and fluctuations can be observed along the mean level in the series therefore, a high order AR and MA behavior is seen.

Time-series plot of data set climate:

After analyzing each of the series independently, we will plot them together to have a visual comparison in the nature of the each of the series. In order to plot them together in the same plot, scaling will be done for the `climate` .

```
# Scaling data climate

scaled2  = scale(climate)

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

plot(scaled2, plot.type = "s", col = c("blue", "red","green","pink","black"), main = "Figure
 24: Time Series plot of  scaled climate",xlab="Time period")
```

Figure 24: Time Series plot of scaled climate



All the five series are likely to be correlated and are non-seasonal in nature. In the next step, correlation between the five series will be calculated to mathematically verify the above visual results-

Calculating correlation for `climate`

The correlation is calculated for each variable in climate using a user-defined function flattenCorrMatrix where correlation between each variable along with its significance is calculated (at 5% level of significance) and displayed-

```
# ++++++++++++++++++++++++++++
# flattenCorrMatrix
# ++++++++++++++++++++++++++++
# cormat : matrix of the correlation coefficients
# pmat : matrix of the correlation p-values

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 -0.39150719 0.029404244
## 2  Temperature   Radiation  0.51935760 0.002753070
## 3     Rainfall   Radiation -0.58131610 0.000604604
## 4  Temperature RelHumidity  0.09350562 0.616848779
## 5     Rainfall RelHumidity  0.33846101 0.062543181
## 6    Radiation RelHumidity -0.05520965 0.768003663
## 7  Temperature         FFD -0.05387003 0.773485332
## 8     Rainfall         FFD -0.22125196 0.231638165
## 9    Radiation         FFD  0.13759174 0.460442428
## 10 RelHumidity         FFD -0.22660091 0.220260446
```

As from the above results, we can say that rainfall & radiation have some moderate negative correlation which is significant (-0.58). Temperature & radiation also have significant moderate positive correlation (0.52). Other than that, no variable has any significant correlation with FFD. This result also suggests that we should only consider **fitting univariate models** as there is no particular correlation between variables we have to worry about while predicting FFD.

Checking non-stationarity

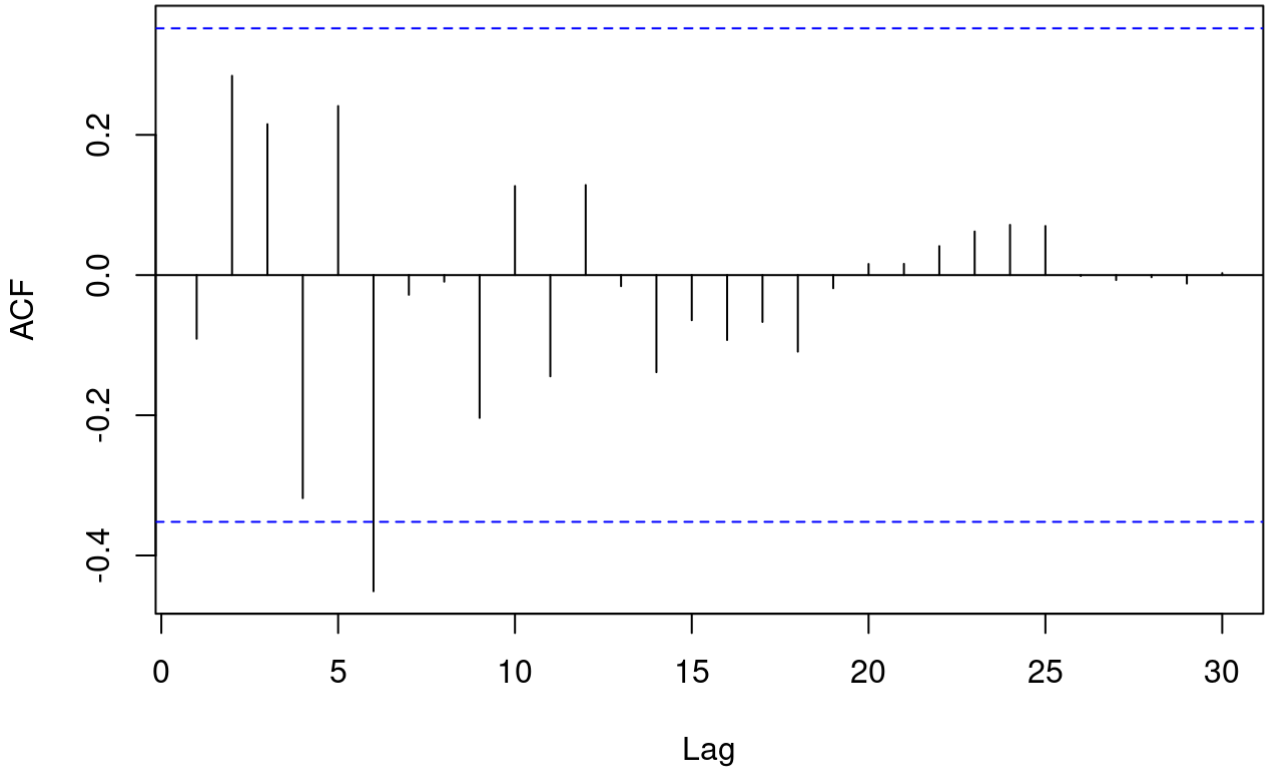
Let's now check whether there is existence of non-stationarity in the five series.The first check can be done through visualization by plotting ACF & PACF for each of the series. This can be further explored and verified by performing ADF tests for each series.

Plotting ACF and PACF:

Plotting ACF and PACF for FFD:

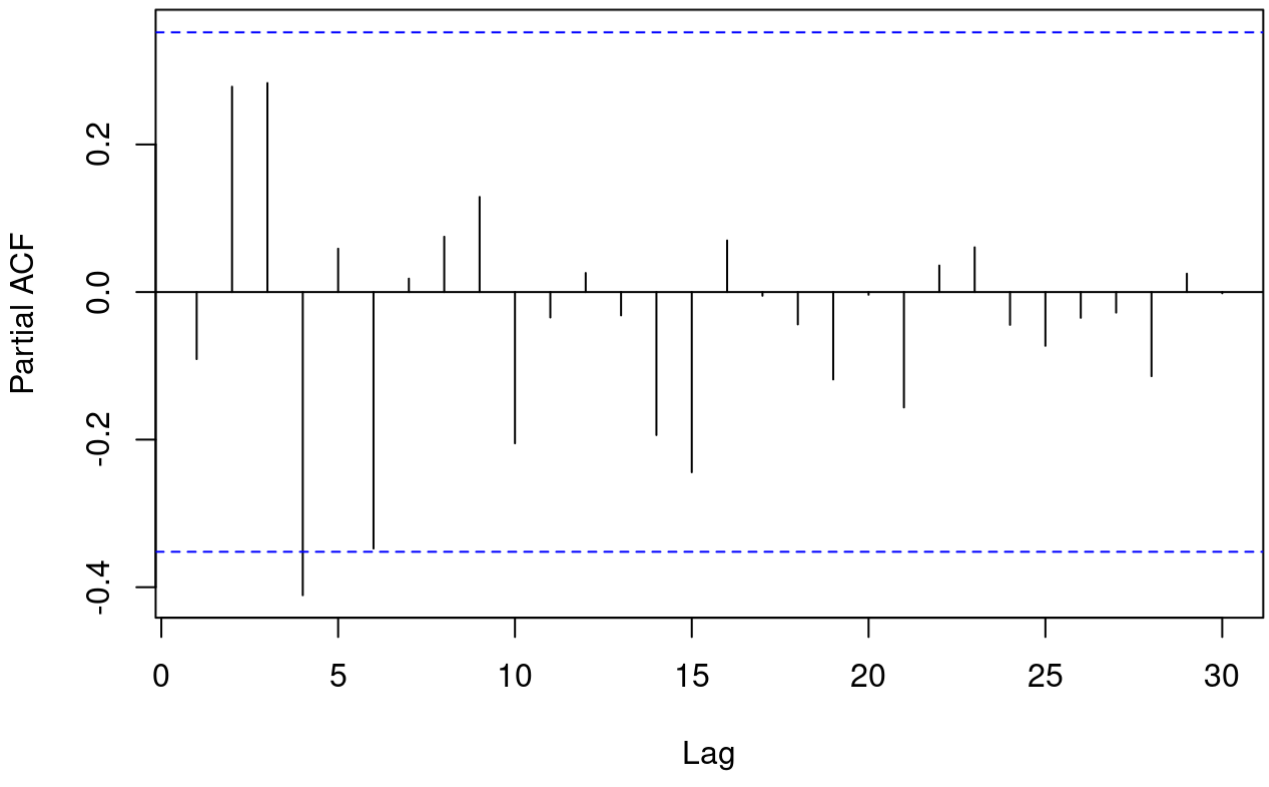
```
# Plotting ACF and PACF


acf(FFD, lag.max = 48, main = "Figure 25.1: FFD ACF ")
```

Figure 25.1: FFD ACF




```
pacf(FFD, lag.max = 48, main = "Figure 25.2: FFD PACF")
```

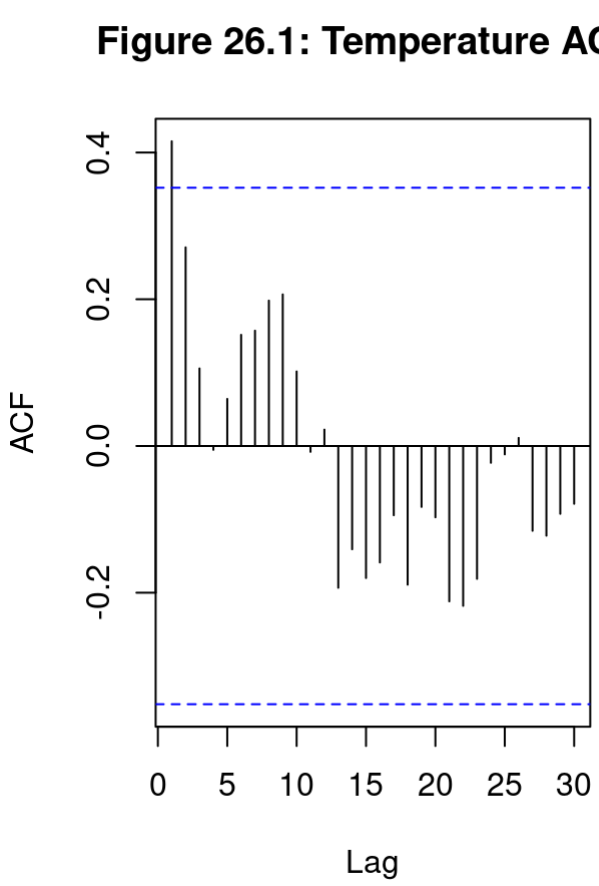
Figure 25.2: FFD PACF

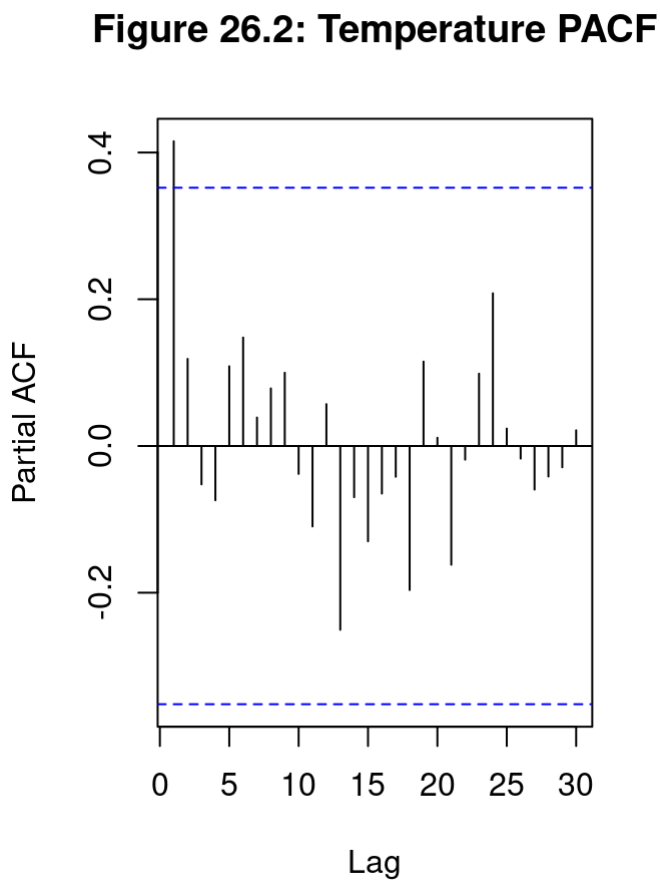


In the ACF plot, there is a slowly decaying pattern of positive & negative lags and we can also see a trend. In the PACF plot, there is only one lag which is significant which suggests that the FFD series is non-stationary.

Plotting ACF and PACF for temperature:

```
# Plotting ACF and PACF

par(mfrow=c(1,2))
acf(temperature, lag.max = 48, main = "Figure 26.1: Temperature ACF ")
pacf(temperature, lag.max = 48, main = "Figure 26.2: Temperature PACF")
```



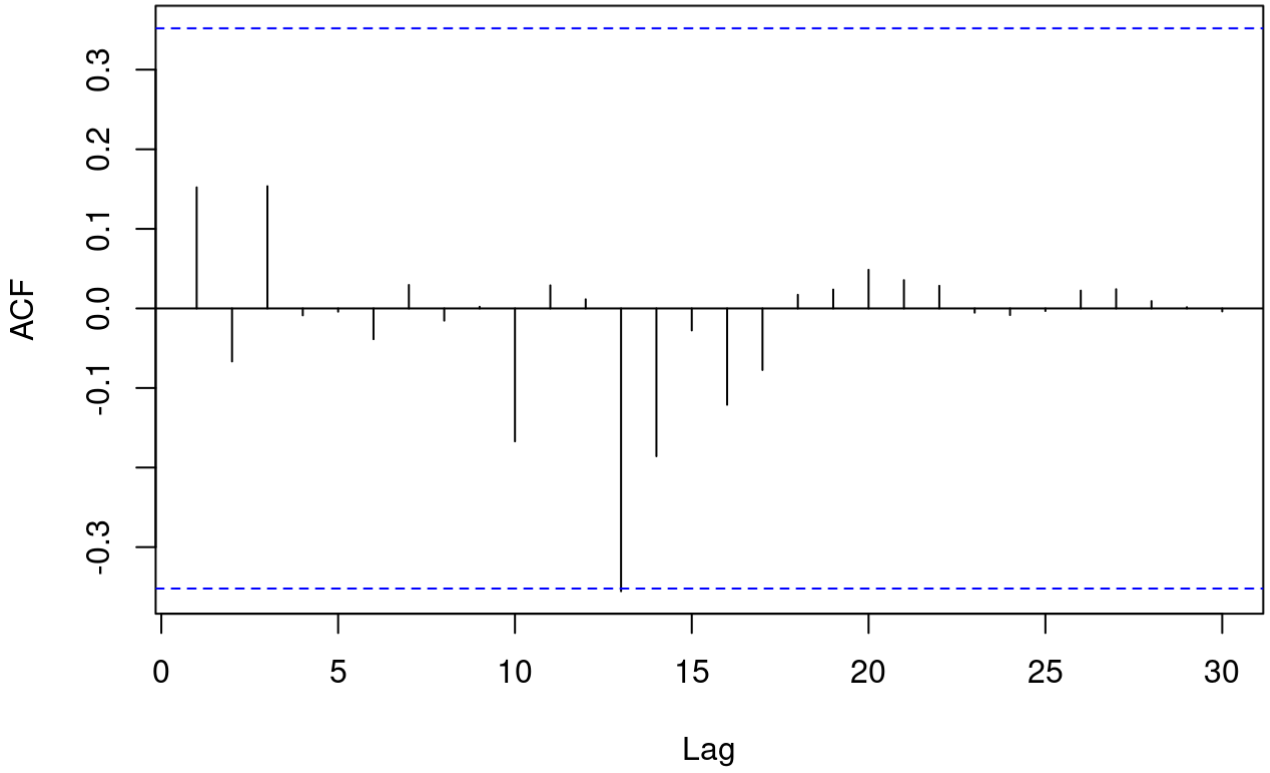


In the ACF plot, there is a slowly decaying pattern of lags and in the PACF plot, the first lag is significant which suggests non-stationarity in the series.

Plotting ACF and PACF for rainfall:

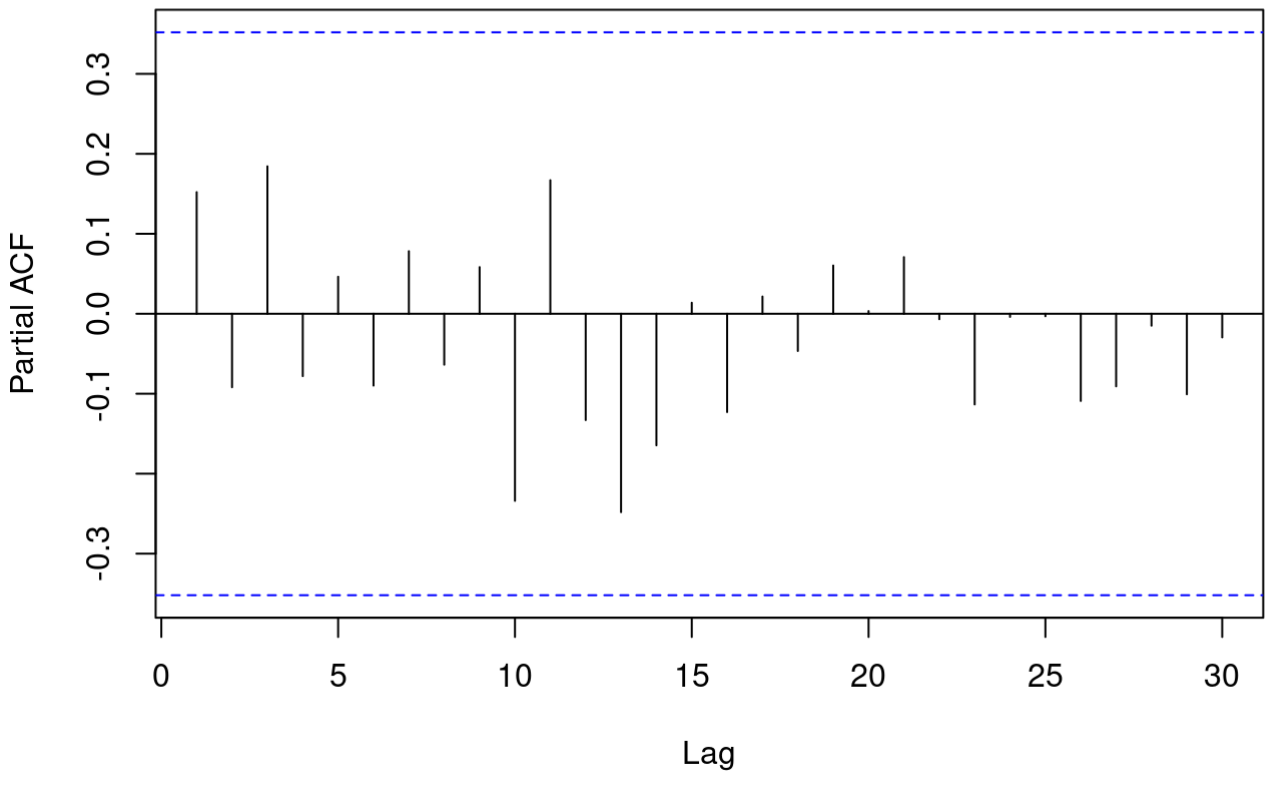
```
# Plotting ACF and PACF


acf(rainfall, lag.max = 48, main = "Figure 27.1: Rainfall ACF ")
```

Figure 27.1: Rainfall ACF




```
pacf(rainfall, lag.max = 48, main = "Figure 27.2: Rainfall PACF")
```

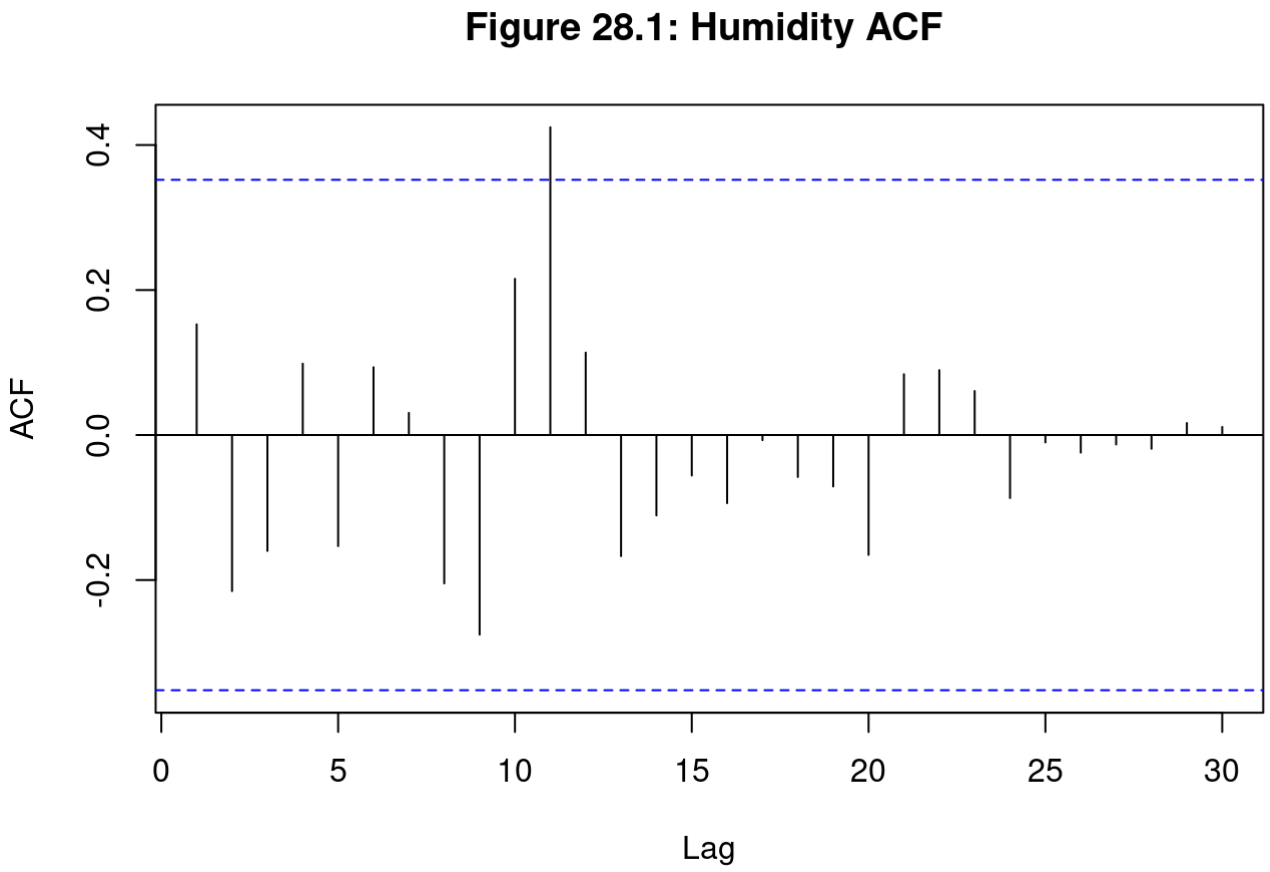
Figure 27.2: Rainfall PACF



As per the ACF plot, there IS a slowly decaying pattern while in PACF, no lag is significant which suggests series is stationary.

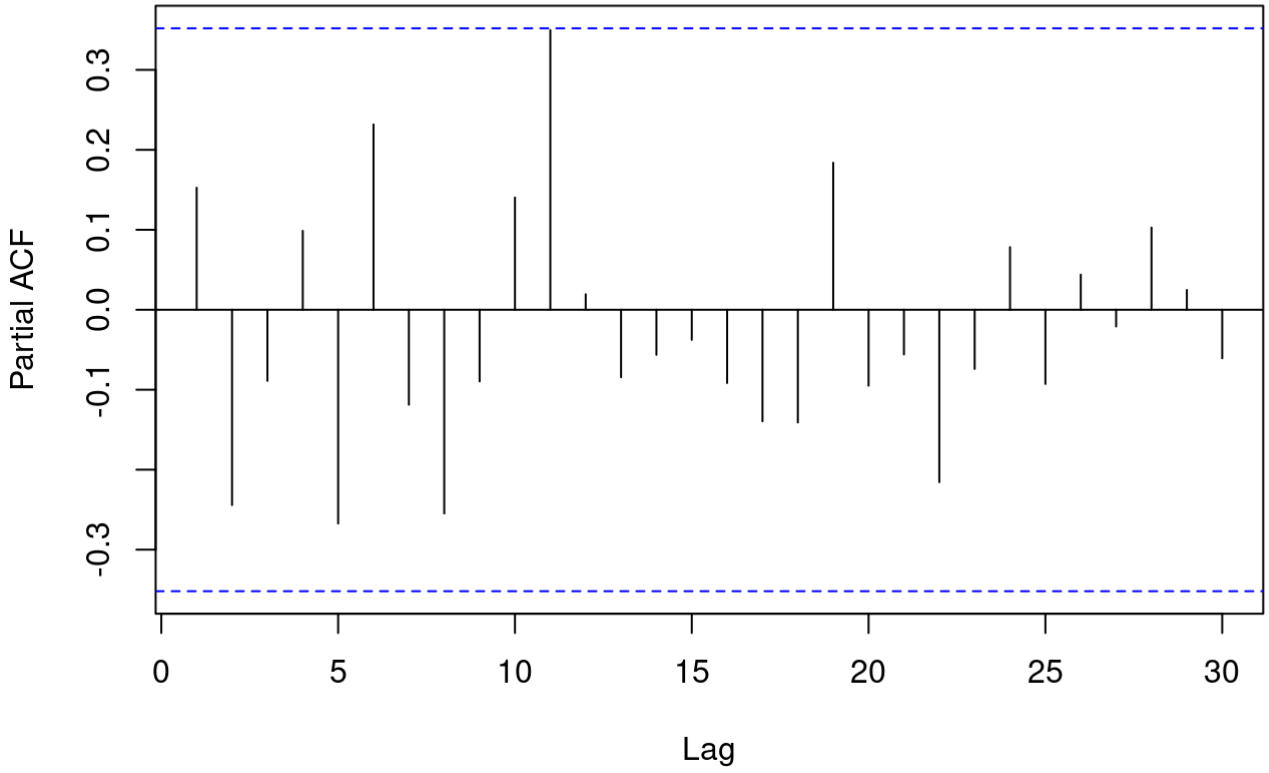
Plotting ACF and PACF for humidity:

```
# Plotting ACF and PACF


acf(humidity, lag.max = 48, main = "Figure 28.1: Humidity ACF ")
```




```
pacf(humidity, lag.max = 48, main = "Figure 28.2: Humidity PACF")
```

Figure 28.2: Humidity PACF

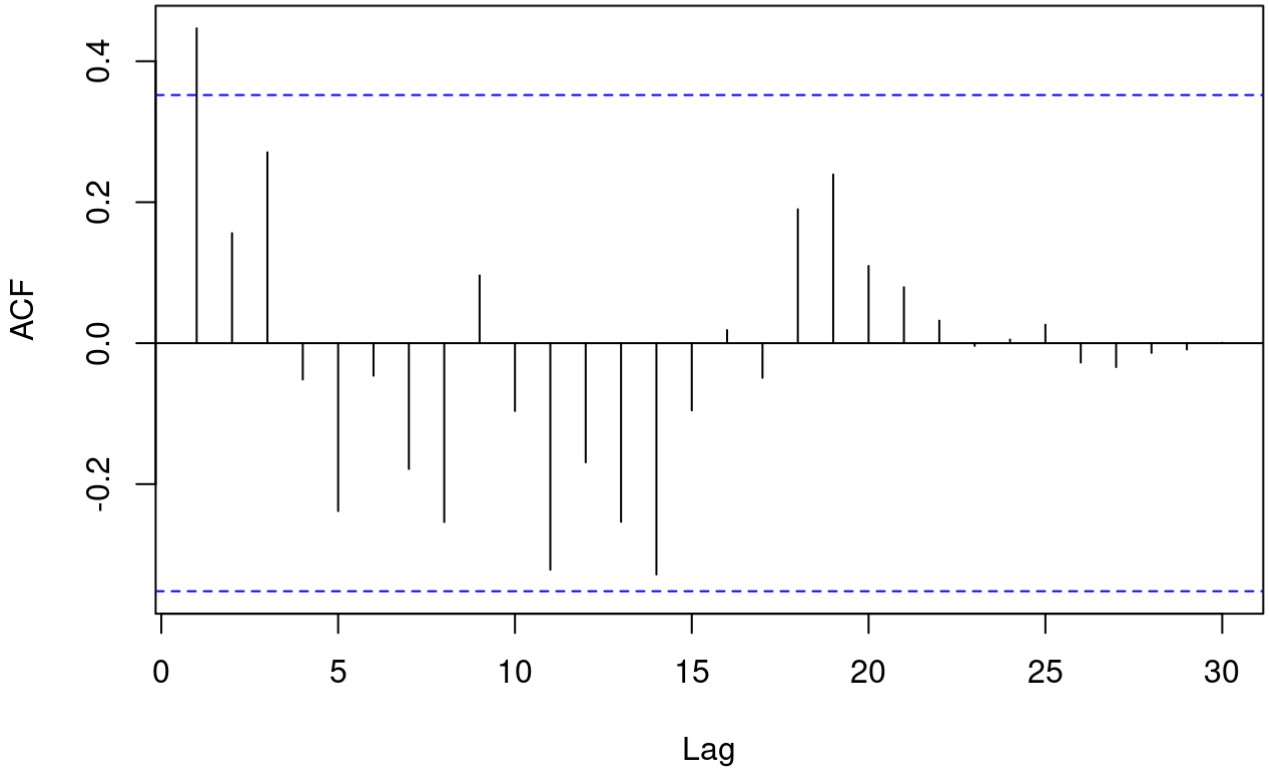


Just like ACF plots of the rest of the previous series, slowly declining behavior of the lags can be observed and in the PACF no lag is significant indicating the presence of stationarity in the humidity series.

Plotting ACF and PACF for radiation:

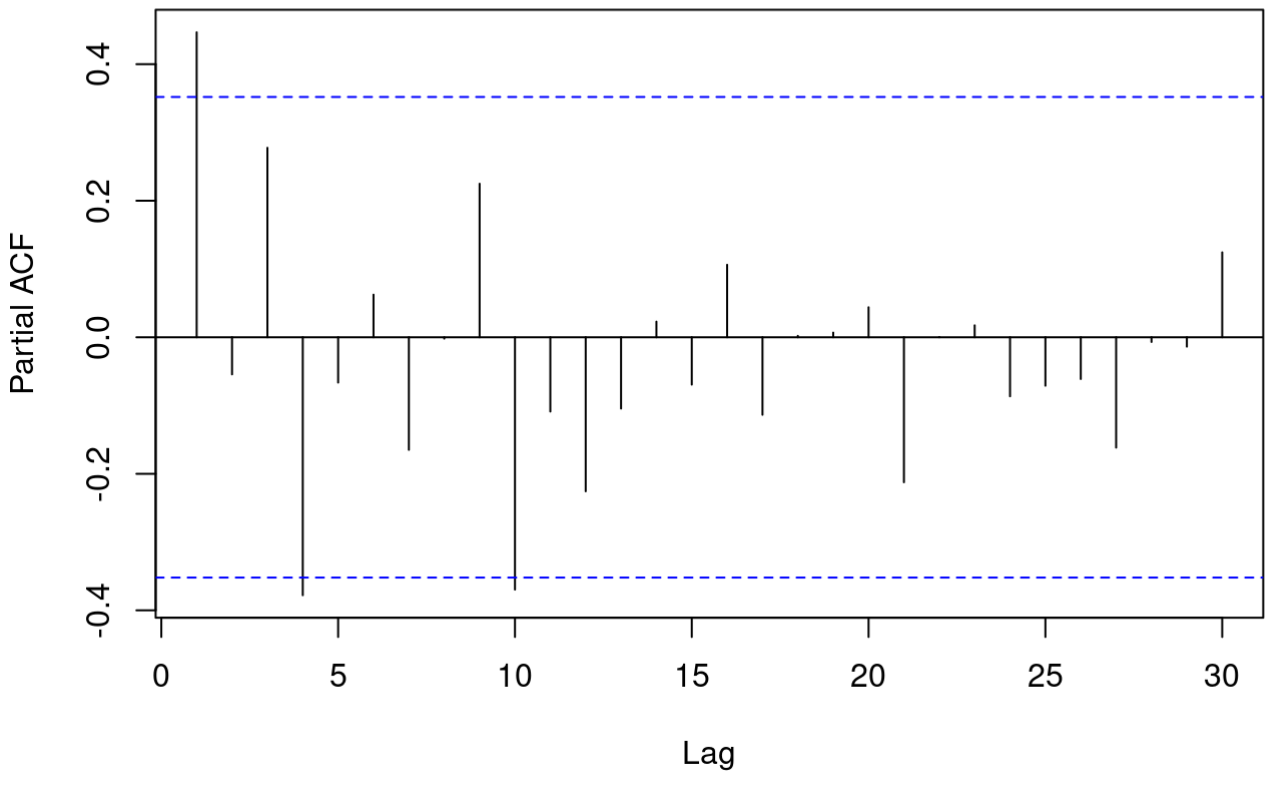
```
# Plotting ACF and PACF


acf(radiation, lag.max = 48, main = "Figure 29.1: Radiation ACF ")
```

Figure 29.1: Radiation ACF




```
pacf(radiation, lag.max = 48, main = "Figure 29.2: Radiation PACF")
```

Figure 29.2: Radiation PACF



In the ACF plot, there is a slowly decaying pattern of lags and in the PACF plot, the first lag is significant which suggests non-stationarity in the radiation series.

Intepretation of ACF and PACF for all five series:

From the ACF and PACF plots it is seen that series FFD, temperature and radiation are non-stationary while rainfall & humidity are stationary. However, ADF tests will be conducted next to verify this.

Dicker-Fuller Unit-Root test (ADF):

FFD series:

Performing ADF test on FFD series-

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  FFD
## Dickey-Fuller = -1.9353, Lag order = 6, p-value = 0.5977
## alternative hypothesis: stationary
```

The ADF test of 6 lag order for FFD indicates series is non-stationary as p-value is greater than 5% level of significance.

Temperature series:

Performing 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
```

The ADF test of 1 lag order supports the ACF & PACF plot indicating series is non-stationary at 5% significance level.

Rainfall series:

Performing 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
```

The ADF test of 0 lag order for rainfall indicates series is stationary as p-value is less than 5% level of significance supporting the ACF & PACF plots plotted for it.

Humidity series:

Performing ADF test on humidity series-

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

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

The ADF test of 0 lag order for humidity indicates series is stationary as p-value is less than 5% level of significance supporting the ACF & PACF plots plotted for it.

Radiation series:

Performing 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
```

The ADF test of 4 lag order supports the ACF & PACF plot indicating radiation series is non-stationary at 5% significance level.

Interpretation: The ADF test computed for all the series completely matches the visual interpretation of ACF & PACF plots as there are no contradicting results.

Transformation for non-stationary series

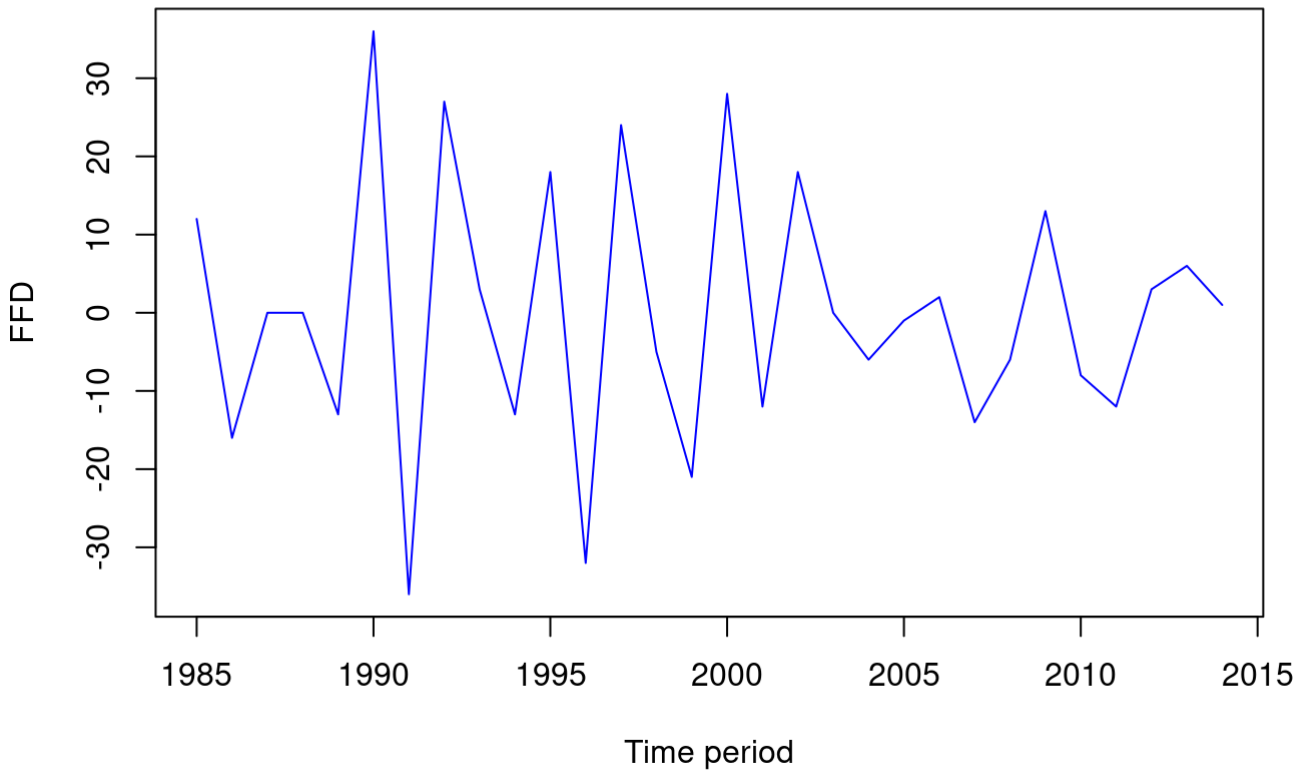
As verified from previous results, FFD, temperature & radiation are non-stationary series. Hence, they need to be converted to stationary in order to proceed with the analysis. Thus, first, second, third etc order of differencing will be applied on each of these series until they become stationary.

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: Time series plot of first differenced FFD



The FFD series after first differencing somwhat looks stationary. Let's confirm that by applying ADF test on the series.

Performing ADF test to check stationarity of first order differenced FFD:

For ADF tests, Ho indicates series is non-stationary and alternate hypothesis (Ha) is series is stationary.

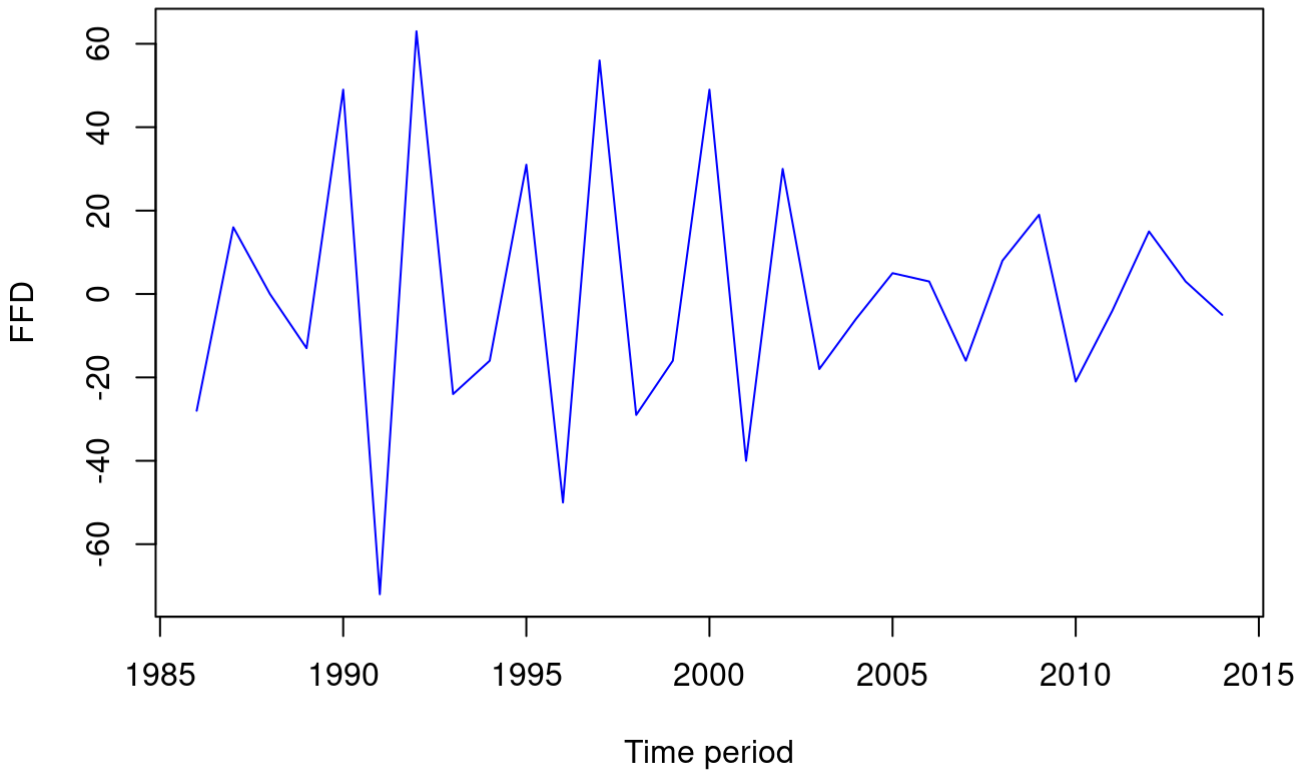
```
adf.test(FFDdiff)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  FFDdiff
## Dickey-Fuller = -3.486, Lag order = 3, p-value = 0.06385
## alternative hypothesis: stationary
```

The series has still not become completely stationary. Let's apply second order of differencing to make it stationary-

```
FFDdiff2 = diff(FFDdiff)
plot(FFDdiff2,ylab='FFD',xlab='Time period', col="blue", main = "Figure 31: Time series plot
 of second differenced FFD")
```

Figure 31: Time series plot of second differenced FFD



The FFD series after second differencing looks completely stationary. Let's confirm that by applying ADF test on the series-

Performing ADF test to check stationarity of second differenced FFD:

```
adf.test(FFDdiff2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  FFDdiff2
## Dickey-Fuller = -5.2016, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

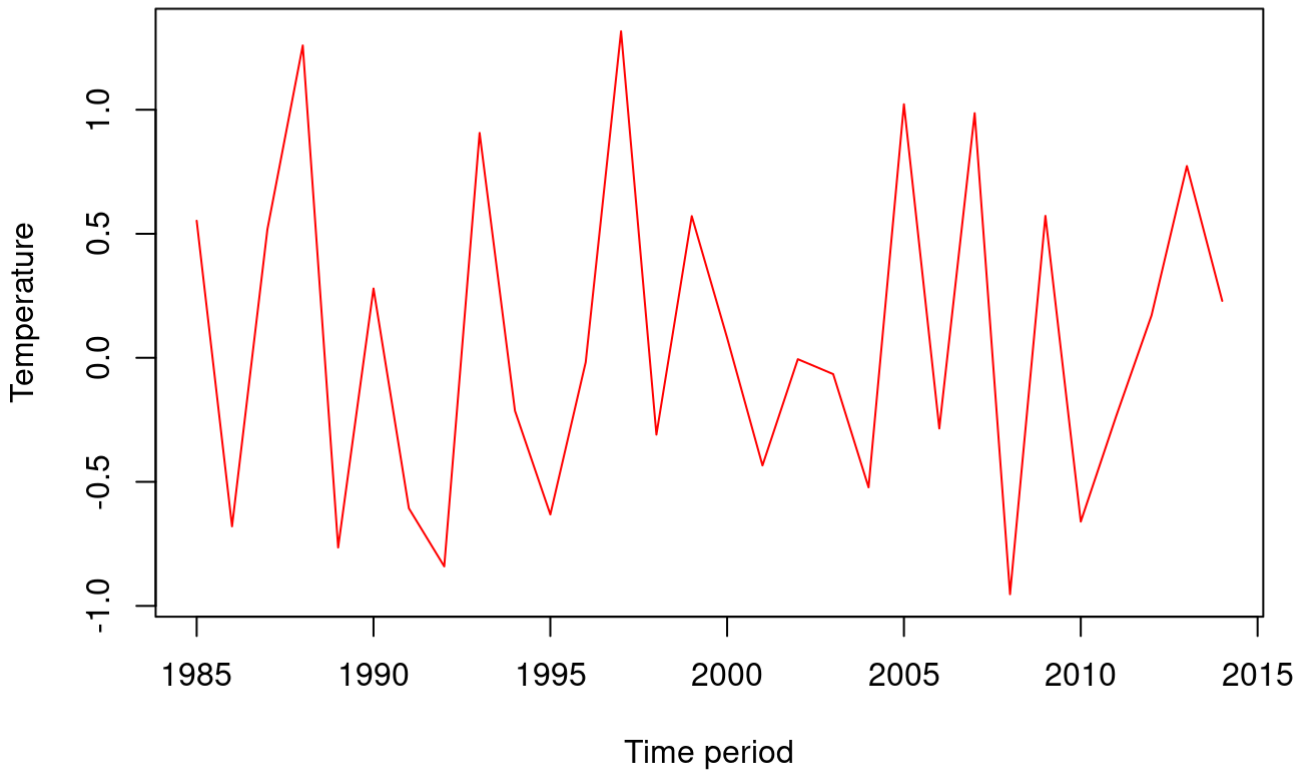
The series has finally become completely stationary as p-value is less than 5% significance level.

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 32: Tim
e series plot of first differenced temperature series")
```

Figure 32: Time series plot of first differenced temperature series



From the above plot, it is difficult to tell whether series has become stationary or not. Let's perform ADF test-

Performing ADF test to check stationarity of first differenced temperature series:

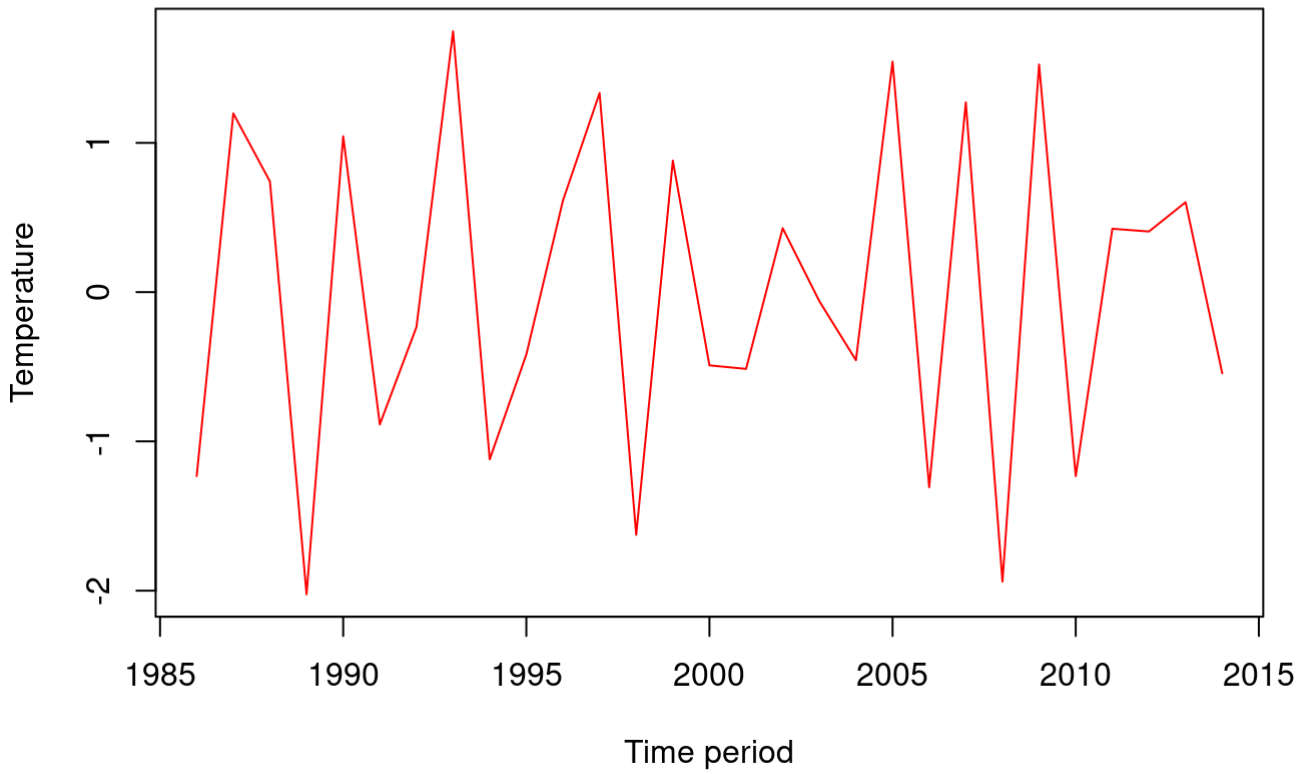
```
adf.test(temperaturediff)
```

```
##
##  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-

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

Figure 33: Time series plot of second differenced temperature series



From the above plot, it is difficult to tell whether series has become stationary or not. Let's perform ADF test-

Performing ADF test to check stationarity of second differenced temperature series:

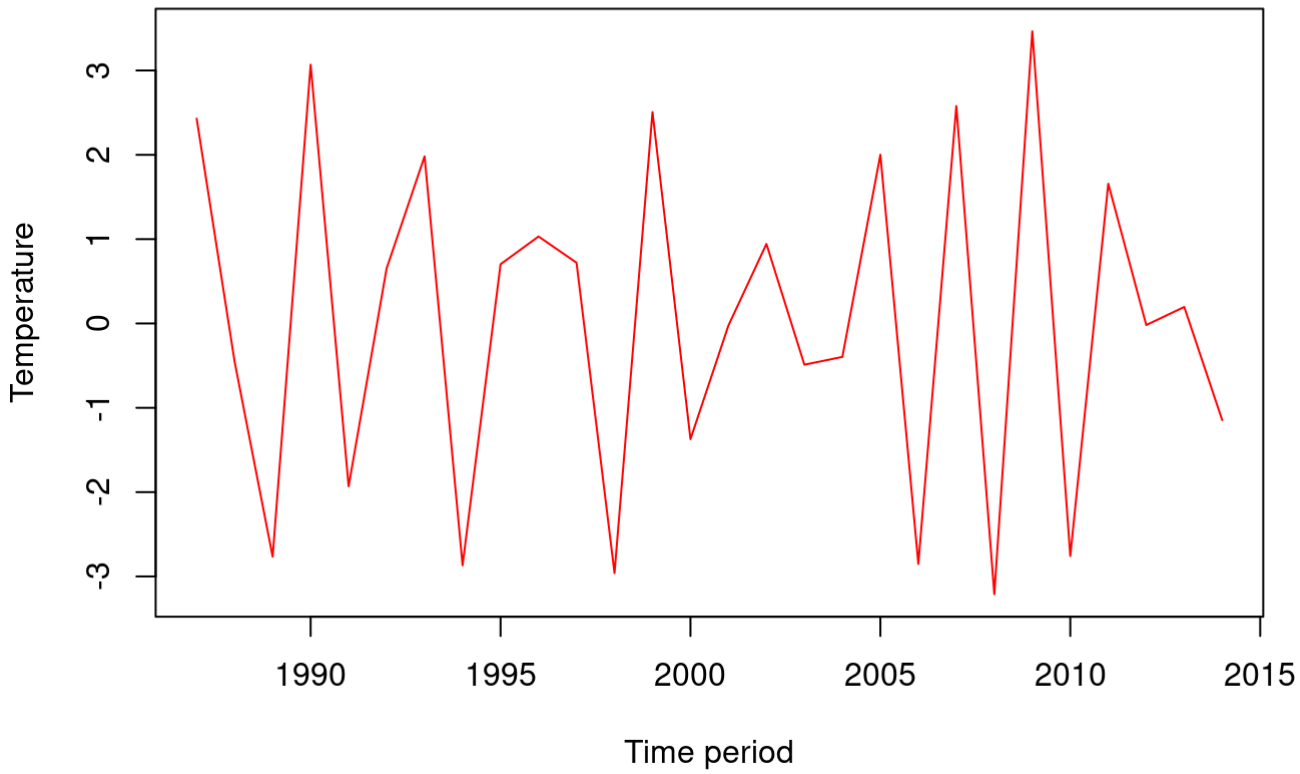
```
adf.test(temperaturediff2)
```

```
##
##  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-

```
temperaturediff3 = diff(temperaturediff2)
plot(temperaturediff3,ylab='Temperature',xlab='Time period', col="red", main = "Figure 34: Ti
me series plot of third order differenced temperature series")
```

Figure 34: Time series plot of third order differenced temperature serie



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 to check stationarity of 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
```

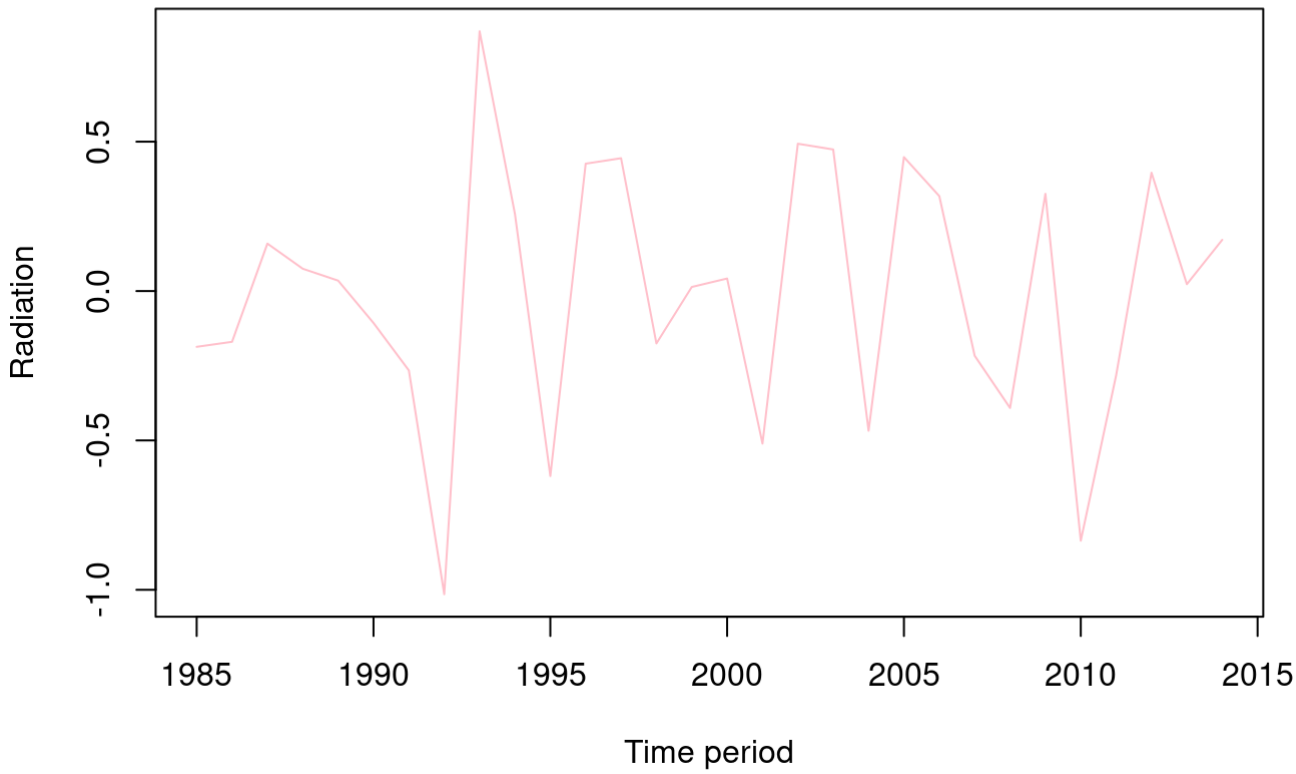
The temperature series has finally become stationary after third order differencing as p-value is less than 5% level of significance.

Transformation of radiation series:

Applying first order differencing on radiation series-

```
radiationdiff = diff(radiation)
plot(radiationdiff,ylab='Radiation',xlab='Time period', col="pink", main = "Figure 35: Time s
eries plot of first differenced radiation series")
```

Figure 35: Time series plot of first differenced radiation series



The series almost look similar to original series after first differencing. Let's perform ADF test-

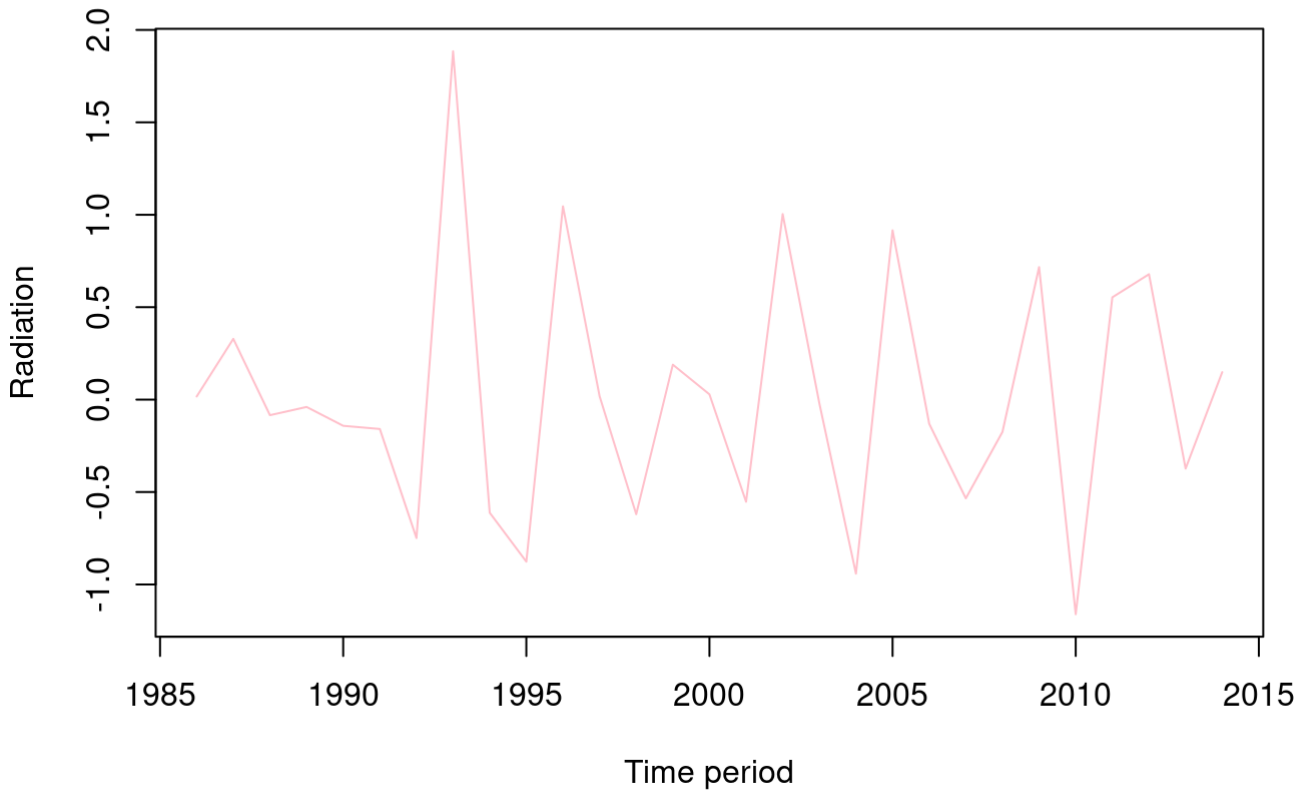
```
adf.test(radiationdiff)
```

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

As the series is still non-stationary (p-value > 0.05), let's apply second order differencing-

```
radiationdiff2 = diff(radiationdiff)
plot(radiationdiff2,ylab='Radiation',xlab='Time period', col="pink", main = "Figure 36: Time
 series plot of second differenced radiation series")
```

Figure 36: Time series plot of second differenced radiation series



The series looks transformed from the initial & end points after second differencing as per above plot. Lets' perform ADF-

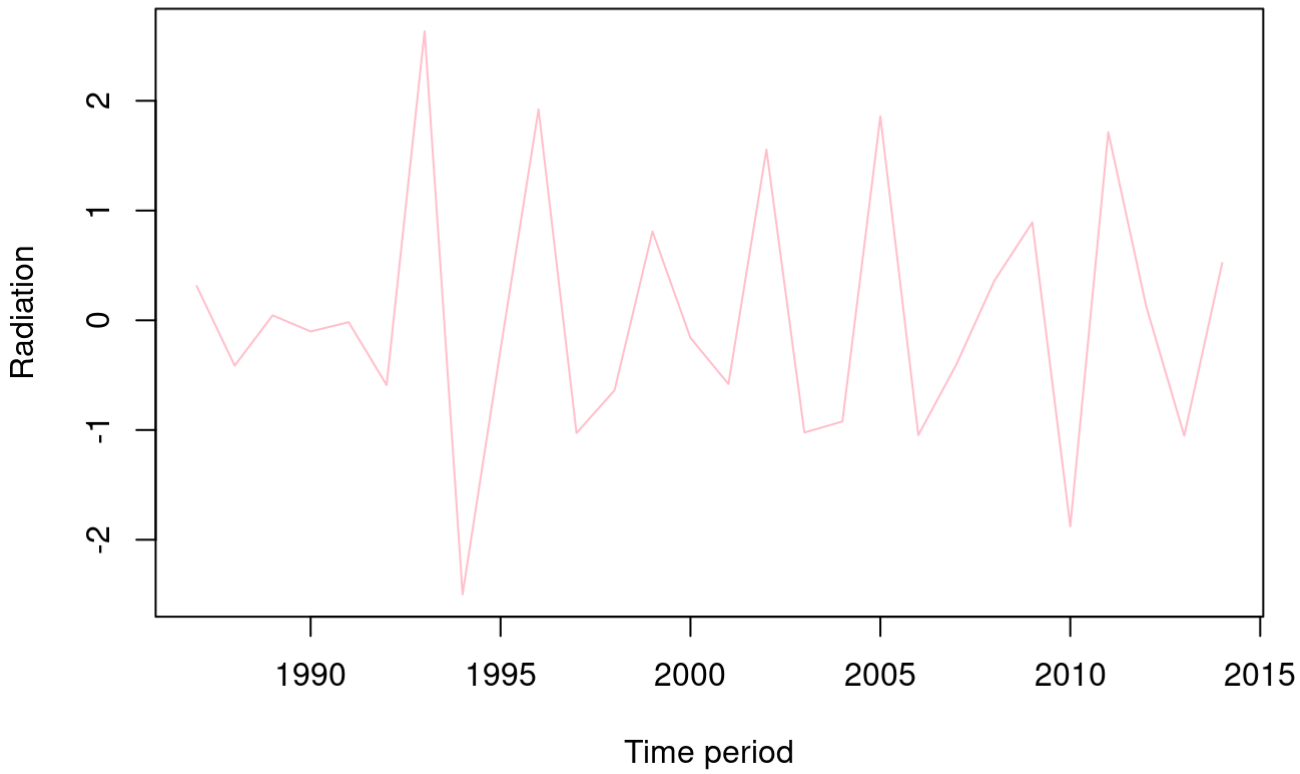
```
adf.test(radiationdiff2)
```

```
##
##  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 (p-value > 0.05. Let's apply third order differencing-

```
radiationdiff3 = diff(radiationdiff2)
plot(radiationdiff3 ,ylab='Radiation',xlab='Time period', col="pink", main = "Figure 37: Time
series plot of third differenced radiation series")
```

Figure 37: Time series plot of third differenced radiation series



The series looks transformed from the initial & end points after third order differencing as per above plot. Lets' perform ADF-

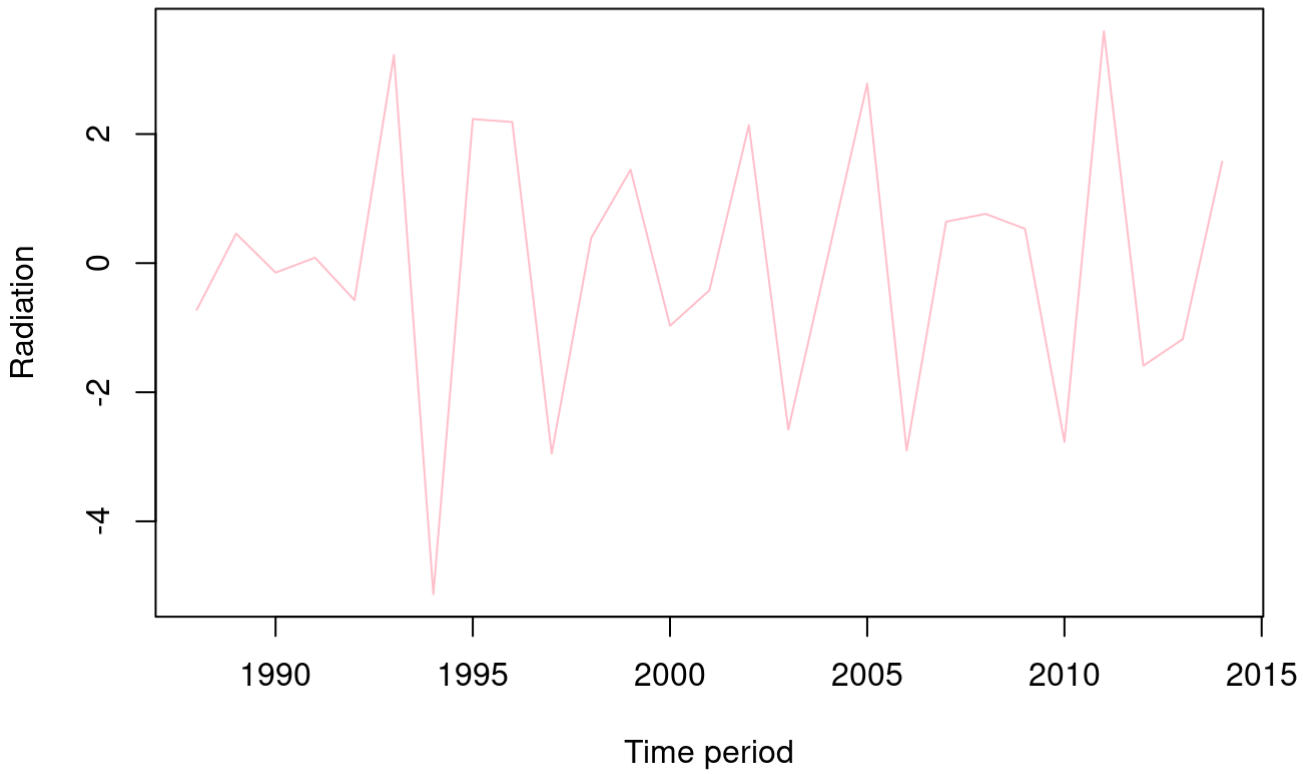
```
adf.test(radiationdiff3)
```

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

The third order differencing could not make the radiation series stationary. Let's apply fourth order differencing-

```
radiationdiff4 = diff(radiationdiff3)
plot(radiationdiff4 ,ylab='Radiation',xlab='Time period', col="pink", main = "Figure 38: Time
series plot of fourth differenced radiation series")
```

Figure 38: Time series plot of fourth differenced radiation series



The series looks transformed from the initial, intervention & end points after fourth order differencing as per above plot. Lets' perform ADF-

```
adf.test(radiationdiff4)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  radiationdiff4
## Dickey-Fuller = -9.6558, Lag order = 2, p-value = 0.01
## alternative hypothesis: stationary
```

Thus, the radiation series has been successfully transformed to a stationary series after fourth order differencing.

Interpretation: Thus, all the three series (FFD, temperature & radiation) have been successfully transformed to stationary after performing second, third & fourth order differencing respectively.

Decomposition

As we can't decompose an yearly time series using X12/STL decomposition, we will proceed to fitting different models for the purpose of predicting FFD.

Modelling using Distributed Lag Models (DLM)

As all the five series have become stationary, let's now fit different DLM models with FFD as dependent variable and other series as predictors one by one (univariate models).

Finite DLM:

For finding the best model using finite DLM, we will consider all the four series (except FFD) as predictor one by one and fit finite DLM based on appropriate lag length.

Computing lag length for the model with temperature as predictor and FFD as dependent variable by using finiteDLMauto function that will perform a comparison based on AIC, BIC and MASE values for lags ranging from 1 to 10. We will be choosing model having lowest of AIC,BIC and MASE values-

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

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

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

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 0.54957 170.8439 184.4227  1.60570 1.19325 -0.09888 0.4396766
## 9      9 0.56122 177.7726 190.8651  1.73681 0.14991 -0.03881 0.8512302
## 8      8 0.58091 184.1658 196.6562  1.42472 0.34114 -0.00400 0.5845755
## 7      7 0.70135 197.5201 209.3006  2.16062 0.78105 -0.16837 0.3527765
## 6      6 0.65042 204.9194 215.8893  1.42796 0.64040 -0.11466 0.9182029
## 5      5 0.68524 216.3322 226.3970  1.80756 1.21527 -0.21538 0.5164414
## 4      4 0.67329 221.3675 230.4383  1.70005 0.94513 -0.14692 0.5475327
## 3      3 0.71738 227.0429 235.0361  4.45925 1.70879 -0.11718 0.6334657
## 2      2 0.73353 231.9415 238.7780 10.43822 1.69886 -0.06547 0.7169714
## 1      1 0.73443 237.5505 243.1553  9.62277 0.67182 -0.02666 0.7318356
```

As per the above output, lag-length should be 10 based on the AIC, BIC and MASE values. Using the finiteDLMauto function, all the other predictors were replaced to check the lag length. For all the predictor variables in the dataset, it was found that optimum lag length is 10. **Thus, for all the finite DLMs, lag length will be taken as 10.**

Temperature vs FFD:

Fitting finite DLM model with temperature as predictor-

```
model.temperature = dlm(x=as.vector(temperature), y=as.vector(FFD), q=10)
```

Rainfall vs FFD:

Fitting finite DLM model with rainfall as predictor-

```
model.rainfall = dlm(x=as.vector(rainfall), y=as.vector(FFD), q=10)
```

Humidity vs FFD:

Fitting finite DLM model with humidity as predictor-

```
model.humidity = dlm(x=as.vector(humidity), y=as.vector(FFD), q=10)
```

Radiation vs FFD:

Fitting finite DLM model with radiation as predictor-

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

Temperature vs FFD (without intercept):

Fitting finite DLM model with temperature as predictor without the intercept-

```
model.nointercept = dlm(formula = FFD ~ temperature - 1, data=data.frame(climate), q=10)
```

Rainfall vs FFD (without intercept):

Fitting finite DLM model with rainfall as predictor without the intercept-

```
model.nointercept2 = dlm(formula = FFD ~ rainfall - 1, data=data.frame(climate), q=10)
```

Humidity vs FFD (without intercept):

Fitting finite DLM model with humidity as predictor without the intercept-

```
model.nointercept3 = dlm(formula = FFD ~ humidity - 1, data=data.frame(climate), q=10)
```

Radiation vs FFD (without intercept):

Fitting finite DLM model with radiation as predictor without the intercept-

```
model.nointercept4 = dlm(formula = FFD ~ radiation - 1, data=data.frame(climate), q=10)
```

Sorting based on AIC,BIC and MASE:

```
# Sorting based on AIC

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.radiation$model    13 134.8487
## model.rainfall$model     13 159.5287
## model.temperature$model  13 170.8439
## model.humidity$model     13 172.2492
## model.nointercept4$model 12 174.0916
## model.nointercept3$model 12 174.1069
## model.nointercept$model  12 189.0983
## model.nointercept2$model 12 202.8882
```

```
# Sorting based on BIC

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.radiation$model    13 148.4275
## model.rainfall$model     13 173.1074
## model.temperature$model  13 184.4227
## model.humidity$model     13 185.8280
## model.nointercept4$model 12 186.6259
## model.nointercept3$model 12 186.6411
## model.nointercept$model  12 201.6326
## model.nointercept2$model 12 215.4225
```

```
# Sorting based on MASE

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$model    21 0.2319041
## model.rainfall$model     21 0.4554174
## model.temperature$model  21 0.5495688
## model.humidity$model     21 0.5702439
## model.nointercept4$model 21 0.6007618
## model.nointercept3$model 21 0.6153355
## model.nointercept$model  21 0.9011990
## model.nointercept2$model 21 1.2496951
```

From the above results, the best model in terms of AIC, BIC and MASE is model.radiation. Let's analyze this model further-

Analysing Radiation vs FFD finite DLM:

Using summary and checkresiduals function, analysing the model.radiation finite DLM-

```
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
## -6.6469 -2.4028  0.3735  1.6333  5.7969
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 736.33179   99.15460   7.426 3.99e-05 ***
## x.t          -0.08165    5.96616  -0.014 0.989379
## x.1          17.47280    7.12240   2.453 0.036563 *
## x.2          -7.46471    4.27694  -1.745 0.114883
## x.3         -13.78589    4.08428  -3.375 0.008187 **
## x.4           6.92619    4.27792   1.619 0.139888
## x.5         -12.07987    4.06633  -2.971 0.015684 *
## x.6           7.77051    4.21777   1.842 0.098546 .
## x.7           5.73213    4.13563   1.386 0.199112
## x.8         -12.61493    4.20313  -3.001 0.014924 *
## x.9           7.47535    5.19309   1.439 0.183874
## x.10        -28.58673    5.21824  -5.478 0.000391 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.935 on 9 degrees of freedom
## Multiple R-squared:  0.9109, Adjusted R-squared:  0.8021
## F-statistic: 8.367 on 11 and 9 DF,  p-value: 0.001773
##
## AIC and BIC values for the model:
##        AIC      BIC
## 1 134.8487 148.4275
```

From the above summary, we can interpret the following things about model.finite2-

1. The model model.finite2 is a good fitted model. It contains a mixture of significant & insignificant lags.
2. For lags x.t, x.2,x.3,x.5,x.8,x.10, the flowering is earlier as FFD values are lower(negative coefficients) while for rest of the lags (including intercept) flowering is later as they have positive coefficient values.
3. The value of adjusted R2 is 80.2%. However, since there is only one independent variable, we will look at the R2 value which is 91.1%. This means that model.finite2 is able to explain 91.1% variation in the dependent variable (FFD).
4. The residuals have maximum value of 5.7969 and minimum value of -6.6469 with residual standard error (RSE) as 4.935.
5. The AIC & BIC values of the model are 134.8487 & 148.4275 respectively.
6. As per the F-test of the overall significance of model, the model.finite2 is a significant model at 5% level of significance.

Calculating VIF for model.finite2 finite DLM:

Calculating VIF for the model.finite2 DLM to check for presence of multicollinearity in the model-

```
vif(model.finite2$model)
```

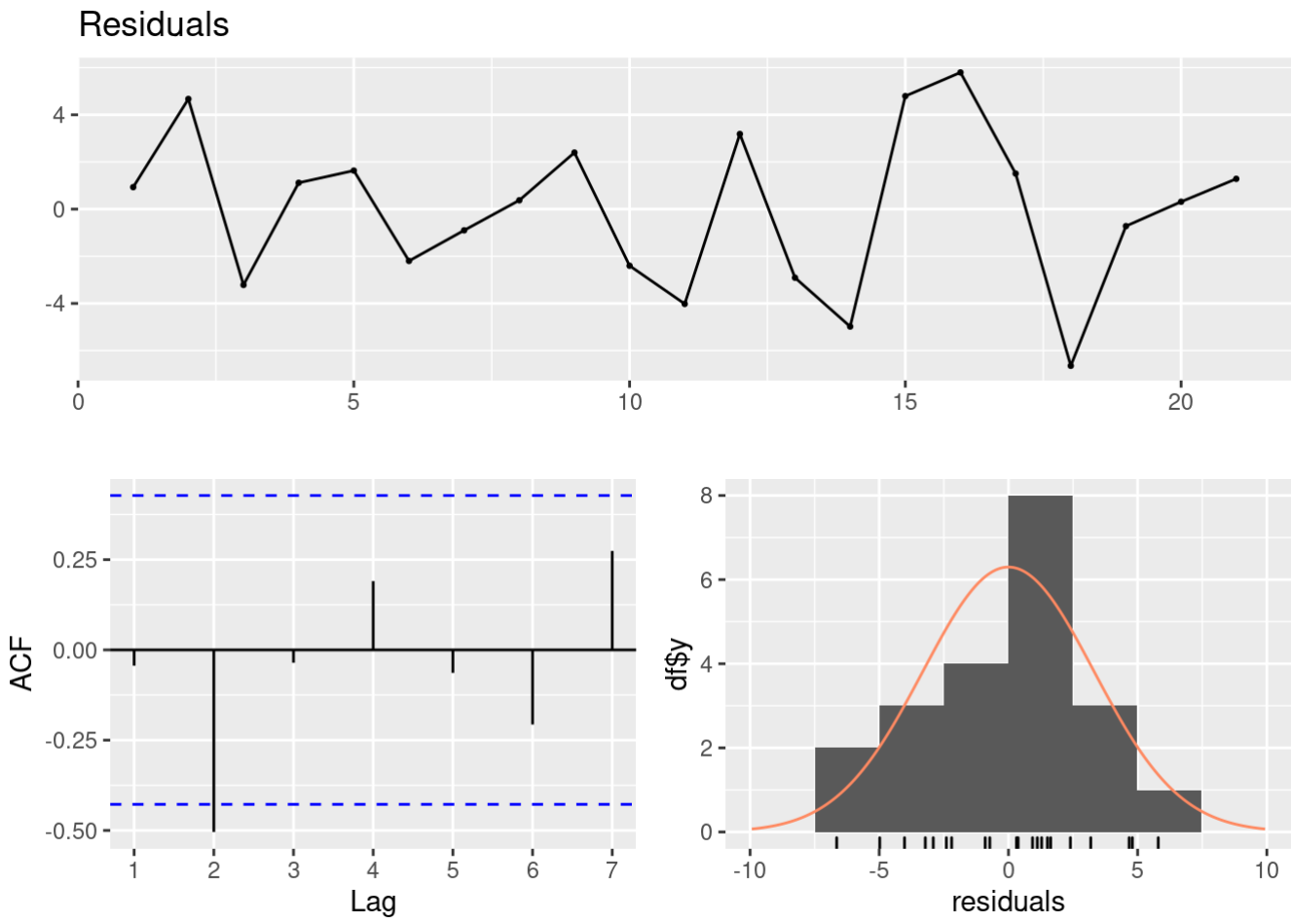
```
##      x.t      x.1      x.2      x.3      x.4      x.5      x.6      x.7
## 4.571066 6.783792 3.502952 3.193621 3.262380 2.898270 2.938461 2.800495
##      x.8      x.9     x.10
## 2.625446 3.207576 3.013446
```

As seen above, the VIF <10 for all the lags in the model. Hence, there is no issue of multicollinearity in the model.finite2 model.

Residual Analysis for model.finite2 DLM:

The residual analysis will be performed on the model.finite2 using checkresiduals function.

```
#Residual analysis of model.finite2 model


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



As per the above plots generated, we can see that residuals seem to be spread randomly without following any particular pattern. In ACF plot, one lag is significant which shows the presence of serial correlation in the residuals. The residuals do not seem to follow normal distribution as the histogram obtained is not symmetric.

Overall, we can conclude that the finite DLM model.finite2 is a good fit significant finite DLM as it explains 91.1% of the variation in FFD. The model has the best MASE value of 0.2319.

Forecasting using model.finite2

For generating the forecasts for the next 4 years, we will use the values of radiation for the next four years from the `task2_covariate` data. The next four year values for radiation are-

```
# Store the radiation values after 2014 in x


 x <- task2_covariate[,4]



# Display radiation values after 2014

x
```

```
## # A tibble: 4 × 1
##   Radiation
##       <dbl>
## 1      14.6
## 2      14.6
## 3      14.8
## 4      14.8
```

Using these values in the best model, let's compute predictions for FFD for the next four years-

Computing Prediction intervals & point forecasts:

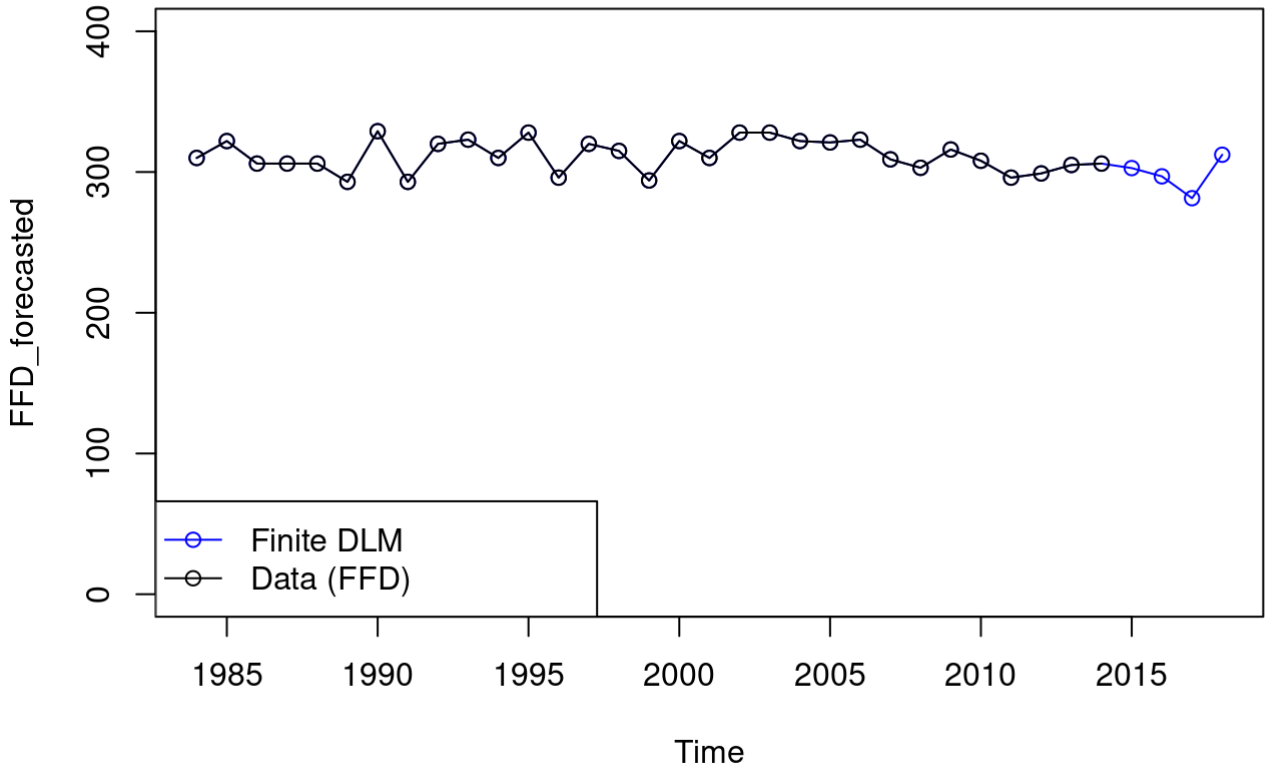
```
# Generating prediction intervals & point forecasts

 forecast(model = model.finite2, x = x, h = 4, interval = TRUE)$forecasts

# Generating point forecasts

forecastFFDfinite <- dLagM::forecast(model.finite2,x = c(14.60,14.56,14.79,14.79) ,h = 4)

forecastFFDfinite <- round(forecastFFDfinite$forecasts,2)
```

```
##   95% LB Forecast 95% UB
## 1 296.09   302.80 310.80
## 2 289.74   296.93 304.93
## 3 275.08   281.41 289.41
## 4 304.72   312.28 318.28
```

Plotting the four years ahead forecasts:

```
plot(ts(c(as.vector(FFD),forecastFFDfinite),start=1984),type="o",col="blue",ylim=c(0,400), yl
ab="FFD_forecasted",
main="Figure 39: FFD four year ahead predicted values from 2015-2018",)
lines(ts(as.vector(FFD),start=1984),col="black",type="o")
legend("bottomleft", lty = 1,pch=1,text.width=11, col = c("blue","black"), c("Finite DLM","Da
ta (FFD)"))
```

Figure 39: FFD four year ahead predicted values from 2015-2018



As seen from the above plot, the model.finite2 (with radiation as predictor) is able to exactly fit the series so it seems to be almost perfect. It shows that in the four years ahead, there will be few fluctuations in the FFD values. The FFD series will follow a short decreasing pattern from 2015-2017 (early flowering) and will then increase in 2018 by approximately 30 (later flowering). The predictions generated are reliable as the model is exactly matching the previous values for FFD series. The lines for both FFD series & model are interlining with each other. Thus, the model.finite2 gives almost accurate four year ahead predictions for the FFD series from 2015-2018.

Polynomial DLM:

As the finite DLM model was a good fit , we will try to fit a polynomial model.Before, we fit the polynomial model, using finiteDLMauto, we will find the best lag length based on fitted models AIC,BIC and MASE values-

```
# 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 0.67967 163.7328 168.9554  2.34477  0.44779  0.11167 0.1156320
## 9   9 - 2 0.68557 171.5568 177.0120  2.45666  0.31608  0.09568 0.4443567
## 8   8 - 2 0.70137 179.1711 184.8486  2.44451  0.27253  0.06846 0.7823894
## 7   7 - 2 0.77579 193.1759 199.0662  2.06403  0.44213 -0.10914 0.9905490
## 6   6 - 2 0.70753 201.1191 207.2135  1.85570  0.64601 -0.06741 0.7049536
## 5   5 - 2 0.73071 211.4676 217.7581  2.15713  0.92511 -0.09650 0.4365501
## 4   4 - 2 0.71100 218.4577 224.9369  2.00645  0.77646 -0.09034 0.4362252
## 3   3 - 2 0.74285 225.8207 232.4817  4.91268 -1.41813 -0.10079 0.4893547
## 2   2 - 2 0.73353 231.9415 238.7780 10.43822  1.69886 -0.06547 0.7169714
## 1   1 - 2 0.73443 237.5505 243.1553  9.62277  0.67182 -0.02666 0.7318356
```

As per the above output, lag-length should be 10 based on the AIC, BIC and MASE values. Using the finiteDLMauto function, all the other predictors were replaced to check the lag length. For all the predictor variables in the dataset, it was found that optimum lag length is 10. **Thus, for all the polynomial DLMs, lag length will be taken as 10 & order as 2.**

Fitting Polynomial DLM:

As models without intercept and models with intercept were giving exactly same model, no models without intercept was fitted for polynomial method. Let's fit the possible combination of possible models-

```
# Fitting all combination of polynomial models

model.temperaturepoly = polyDlm(x = as.vector(temperature), y = as.vector(FFD), q = 10, k = 2
)

model.rainfallpoly = polyDlm(x = as.vector(rainfall), y = as.vector(FFD), q = 10, k = 2)

model.humiditypoly = polyDlm(x = as.vector(humidity), y = as.vector(FFD), q = 10, k = 2)

model.radiationpoly = polyDlm(x = as.vector(radiation), y = as.vector(FFD), q = 10, k = 2)
```

In the above code, we fitted 4 different polynomial models. Let's generate the accuracy score for each of them-

Comparison based on AIC,BIC and MASE:

```
# Sorting based on AIC, BIC and MASE

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

```
##                             df      AIC
## model.radiationpoly$model    5 163.2937
## model.humiditypoly$model     5 163.5124
## model.temperaturepoly$model  5 163.7328
## model.rainfallpoly$model     5 164.9582
```

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

```
##                              df      BIC
## model.radiationpoly$model    5 168.5163
## model.humiditypoly$model     5 168.7350
## model.temperaturepoly$model  5 168.9554
## model.rainfallpoly$model     5 170.1808
```

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

```
##                               n      MASE
## model.radiationpoly$model    21 0.6355363
## model.rainfallpoly$model     21 0.6551896
## model.temperaturepoly$model 21 0.6796679
## model.humiditypoly$model     21 0.7312748
```

As model.radiationpoly have lowest AIC,BIC and MASE scores, lets' analyze its other parameters-

Analyzing model.radiationpoly:

```
summary(model.radiationpoly$model)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -23.051  -3.964   2.555   5.979  18.374
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 589.3348   194.2721   3.034   0.0075 **
## z.t0          1.8875     3.3075   0.571   0.5757
## z.t1         -0.8977     1.5925  -0.564   0.5803
## z.t2          0.0249     0.1612   0.154   0.8791
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.35 on 17 degrees of freedom
## Multiple R-squared:  0.2605, Adjusted R-squared:  0.1301
## F-statistic: 1.997 on 3 and 17 DF,  p-value: 0.1527
```

From the above summary, we can generate following insights about the model.radiationpoly-

1.The model model.radiationpoly is a poorly fitted model and almost all lags are insignificant except the intercept.

2. For lags z.t1 & z.t2, the flowering is earlier as FFD values are lower(negative coefficients) while for intercept & z.t0 lag flowering is later as they have positive coefficient values.
3. The value of R2 is 0.2605.This means that model.radiationpoly is able to explain only (26%) variation in the dependent variable (FFD).
4. The residuals have maximum value of 18.374 and minimum value of -23.051 with residual standard error (RSE) as 10.35.

5. As per the F-test of the overall significance of model, the model.radiationpoly is an insignificant model at 5% level of significance despite having many insignificant number of lags and poor R2 value.

Calculating VIF for model.radiationpoly DLM:

Calculating VIF for the model.radiationpoly DLM to check for presence of multicollinearity in the model-

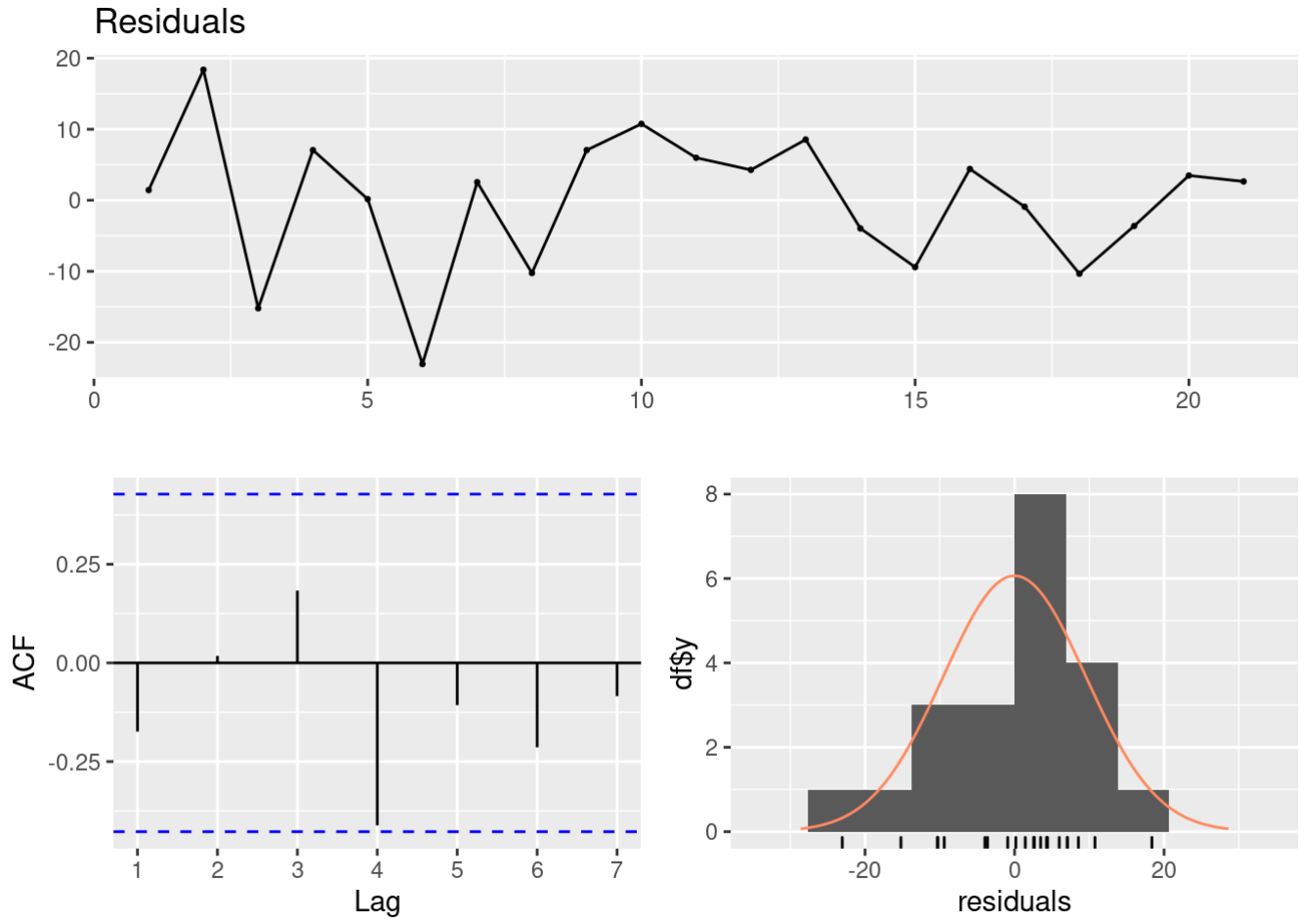
```
vif(model.radiationpoly$model)
```

```
##     z.t0     z.t1     z.t2
##  9.16423 69.99257 40.54527
```

As for not all lags VIF <10, hence, there is presence of multicollinearity in the model.radiationpoly model.

Residual Analysis for model.radiationpoly2:

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



As per the above plots generated, we can see that residuals seem to be spread randomly without following any particular pattern. In ACF plot, no lag is significant which shows there is no presence of serial correlation in the residuals. The residuals do not seem to follow normal distribution as the histogram obtained is not symmetric.

The model.radiationpoly is an insignificant model and is a poor fit in comparison to model.finite2.

Forecasting using model.radiationpoly

Lets' generate the forecast point intervals and plot the forecast for the next four years-

```
# Generating point forecasts using task2_covariate data for radiation

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

forecastFFDpoint <- round(forecastFFDpoint$forecasts,2)

# Generating prediction intervals

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

round(forecastFFDinterval$forecasts,2)
```

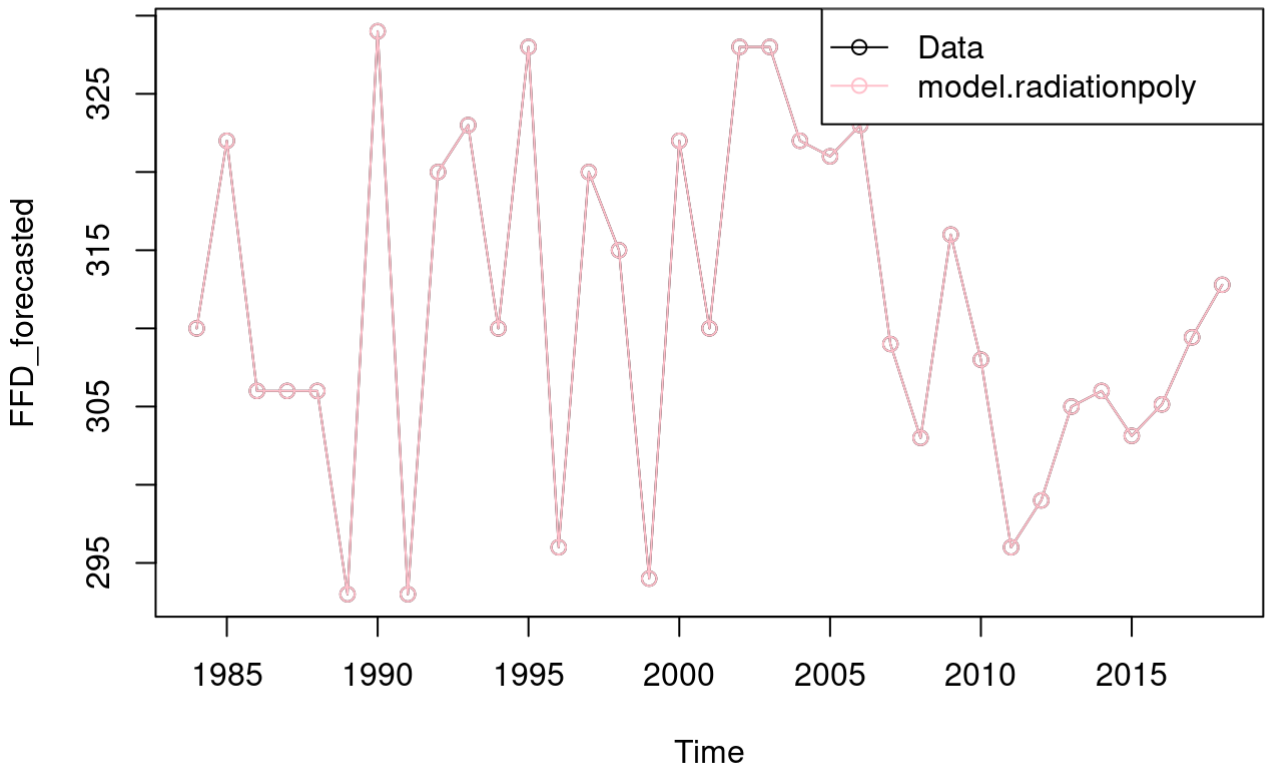
```
##    Lower Estimate  Upper
## 1 282.82   303.13 323.41
## 2 282.50   305.14 322.71
## 3 288.59   309.43 331.12
## 4 292.87   312.79 331.84
```

The point forecast and confidence intervals generated from the model.poly are quite good. As per the forecast, the FFD values will follow an increasing trend from 2015-2018 (later flowering).

Plotting forecast

```
plot(ts(c(as.vector(FFD),forecastFFDpoint),start=1984),type="o",col="black", ylab="FFD_foreca
sted",
main="Figure 40: FFD four year ahead predicted values from 2015-2018")
lines(ts(c(as.vector(FFD),forecastFFDpoint),start=1984),type="o",col="pink")
legend("topright", lty = 1,pch=1,text.width=11, col = c("black","pink"), c("Data","model.radi
ationpoly"))
```

Figure 40: FFD four year ahead predicted values from 2015-2018



The above plot shows four year predicted values for FFD using model.radiationpoly. The forecast generated are reliable as the model fits the FFD data perfectly and the prediction intervals are also good (not wide) for the model.

Koyck DLM:

Fittng all possible Koyck models:

As models without intercept and models with intercept were giving exactly same model, no models without intercept was fitted for koyck method.

```
model.temperaturekoyck = koyckDlm(x = as.vector(temperature), y = as.vector(FFD))
model.rainfallkoyck = koyckDlm(x = as.vector(rainfall), y = as.vector(FFD))
model.humiditykoyck = koyckDlm(x = as.vector(humidity), y = as.vector(FFD))
model.radiationkoyck = koyckDlm(x = as.vector(radiation), y = as.vector(FFD))
```

Comparison based on AIC,BIC and MASE:

```
# Sorting based on  MASE


MASE_koyck <- MASE(model.temperaturekoyck,model.rainfallkoyck,model.humiditykoyck,model.radia
tionkoyck)


# Sorting in ascending order

arrange(MASE_koyck,MASE)
```

```
##                          n      MASE
## model.radiationkoyck    30 0.7540611
## model.temperaturekoyck 30 0.7968756
## model.humiditykoyck     30 0.8390184
## model.rainfallkoyck     30 1.1546848
```

From the above results, model.radiationkoyck(with radiation as predictor) is the best model based on MASE value. Let's analyze this model further.

Analyzing model.radiationkoyck:

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

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -20.3177  -7.7263   0.2059   9.4379  18.2406
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  297.541    165.304   1.800   0.0831 .
## Y.1           -0.118      0.212  -0.557   0.5824
## X.t            3.509     12.595   0.279   0.7827
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.76 on 27 degrees of freedom
## Multiple R-Squared: 0.03382, Adjusted R-squared: -0.03775
## Wald test: 0.1555 on 2 and 27 DF,  p-value: 0.8567
##
## Diagnostic tests:
##                  df1 df2  statistic    p-value
## Weak instruments   1  27 6.30587988 0.01832201
## Wu-Hausman         1  26 0.01250576 0.91181777
##
##                             alpha    beta        phi
## Geometric coefficients:  266.1391 3.50942 -0.1179891
```

From the above summary, we can generate following insights about the model.radiationkoyck-

1.The model model.radiationkoyck is a poorly fitted model and all of the lags are insignificant.

2. The value of R2 is 0.033. This means that model.radiationkoyck is able to explain only 3% variation in the dependent variable (FFD).
3. For lags Y.1, the flowering is earlier as FFD values are lower(negative coefficient) while for intercept & x.t lag flowering is later as they have positive coefficient values.
4. The residuals have maximum value of 18.2406 and minimum value of -20.3177with residual standard error (RSE) as 11.76.
5. As per the F-test of the overall significance of model, the model.radiatonkoyck is a pretty insignificant model at 5% level of significance.

6. The model in the first stage of least-squares estimation is moderately significant at the 5% level, according to the Weak instruments test.
7. We may infer that there is an insignificant connection between the explanatory variable and the error term at the 5% level based on the Wu-Hausman test.
8. The geometric coefficients alpha, beta and phi are 266.1391,3.50942 and -0.1179891 respectively.
9. This model is a poorer fit in comparison to other previous models in terms of R2 and F-test p-value.

Calculating VIF for model.radiationkoyck DLM:

Calculating VIF for the model.radiationkoyck DLM to check for presence of multicollinearity in the model-

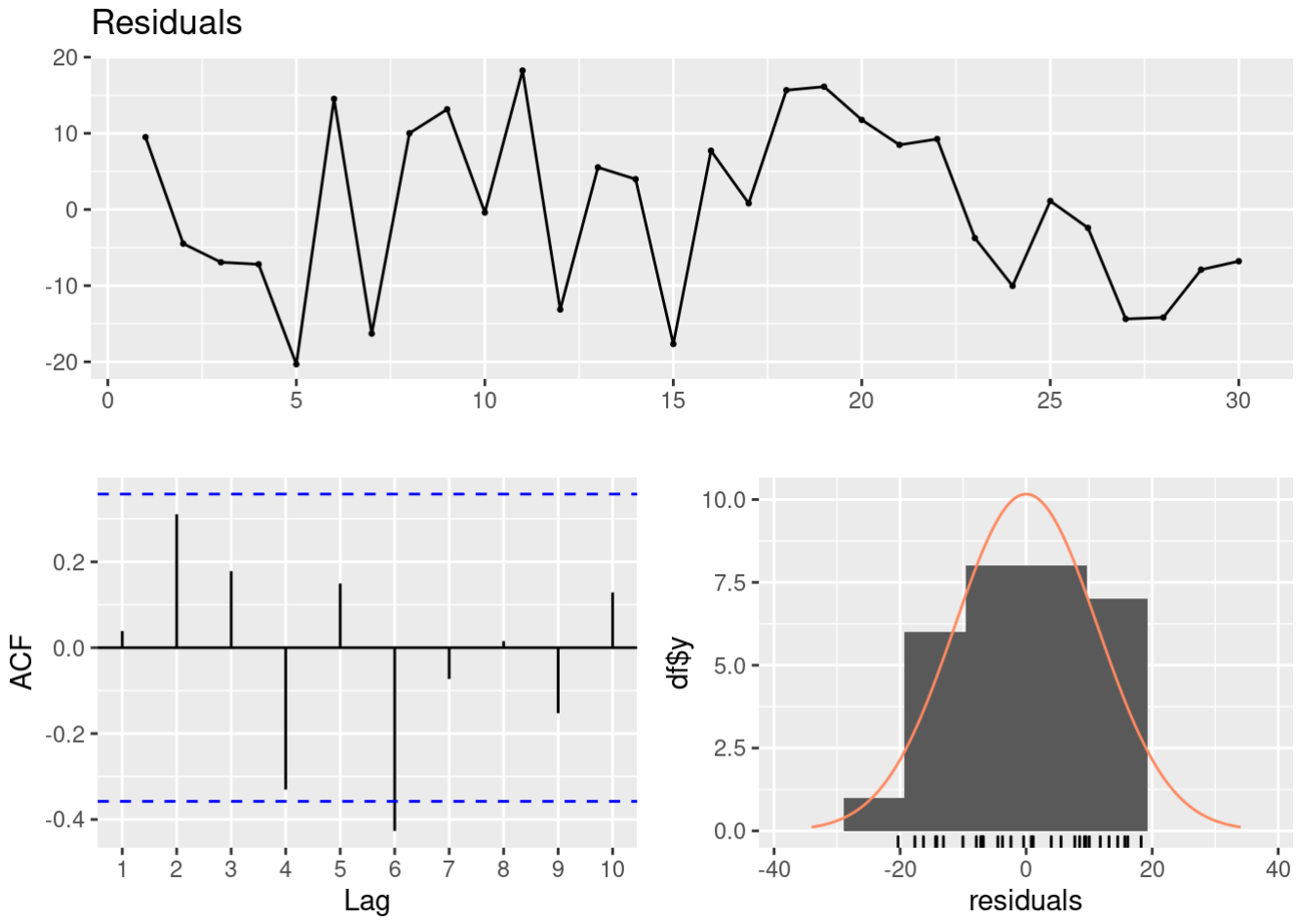
```
vif(model.radiationkoyck$model)
```

```
##      Y.1      X.t
## 1.245589 1.245589
```

As seen above, the VIF < 10 for all the lags in the model. Hence, there is no issue of multicollinearity in the model.radiationkoyck model.

Residual Analysis for model.radiationkoyck DLM:

```
# Residual analysis of model

checkresiduals(model.radiationkoyck$model)
```



As per the above residual plots generated, we can see that residuals seem to be spread randomly without any particular pattern. In ACF plot, no particular lag is significant (except 1) which shows there is no serious presence of autocorrelation in the residuals. The residuals do not have normal distribution as per the histogram plot.

As per the diagnostic checking (residual analysis and other tests), Koyck DLM seems to be a poorer fit in comparison to the previous two models as the model is insignificant even though it does not face any issue of multicollinearity. Let's generate its forecasts-

Forecasting using model.radiationkoyck

Lets' generate the forecast point intervals and plot the forecast for the next four years-

```
# Generating point forecasts using task2_covariate radiation values

forecastFFDpoint2 <- dLagM::forecast(model.radiationkoyck,x = c(14.60,14.56,14.79,14.79) ,h =
4)

forecastFFDpoint2 <- round(forecastFFDpoint2$forecasts,2)

# Generating prediction intervals

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

round(forecastFFDinterval2$forecasts,2)
```

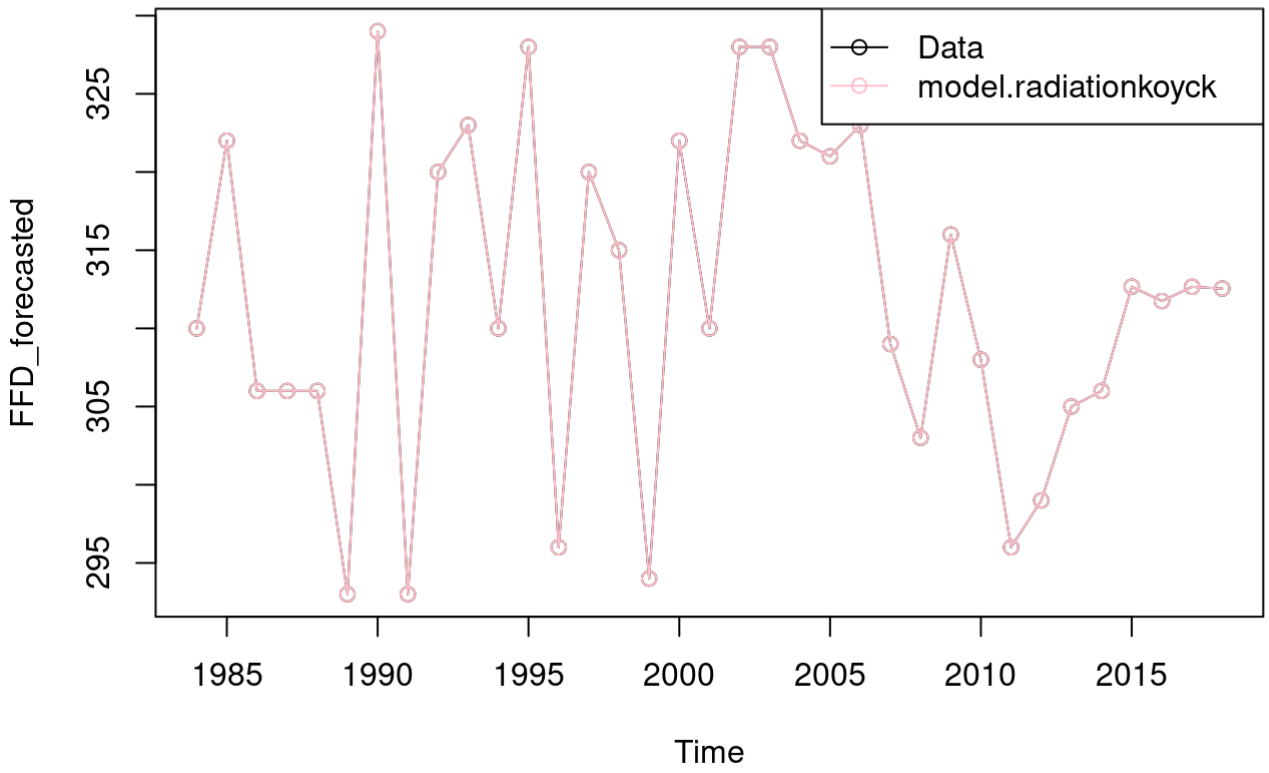
```
##    Lower Estimate  Upper
## 1 288.96   312.67 335.15
## 2 291.16   311.75 337.93
## 3 290.88   312.66 335.37
## 4 286.82   312.55 337.56
```

The forecast generated from the Koyck model are not very good like the poly model. The FFD predicted values for the next four years will almost remain constant in the range of 311-312 and will not change much as per the koyck model. The prediction intervals are a little wider for the koyck model in comparison to Poly model.

Plotting forecast for model.radiationkoyck:

```
plot(ts(c(as.vector(FFD),forecastFFDpoint2),start=1984),type="o",col="black", ylab="FFD_forec
asted",
main="Figure 41: FFD four year ahead predicted values from 2015-2018")
lines(ts(c(as.vector(FFD),forecastFFDpoint2),start=1984),type="o",col="pink")
legend("topright", lty = 1,pch=1,text.width=11, col = c("black","pink"), c("Data","model.radi
ationkoyck"))
```

Figure 41: FFD four year ahead predicted values from 2015-2018



The koyck model fits the previous FFD data perfectly. As per the model, in the next four years, the FFD values will not change muuch and will remain between 311-312. The predictions generated seem to be reliable as prediction intervals are not very wide.

Autoregressive DLM:

Computing autoregressive DLMs for a range of lag lengths and AR process orders using the loop, and fitting the model with the lowest information criterion-

```
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 = 239.2912 BIC = 246.2972 MASE = 0.7282071
## p = 1 q = 2 AIC = 232.1732 BIC = 240.377 MASE = 0.718391
## p = 1 q = 3 AIC = 224.8471 BIC = 234.1725 MASE = 0.6592621
## p = 1 q = 4 AIC = 214.1608 BIC = 224.5275 MASE = 0.5854783
## p = 1 q = 5 AIC = 208.9515 BIC = 220.2744 MASE = 0.5812989
## p = 2 q = 1 AIC = 233.8023 BIC = 242.006 MASE = 0.7298609
## p = 2 q = 2 AIC = 233.8822 BIC = 243.4533 MASE = 0.7025069
## p = 2 q = 3 AIC = 226.4394 BIC = 237.097 MASE = 0.6628448
## p = 2 q = 4 AIC = 216.1331 BIC = 227.7956 MASE = 0.5839391
## p = 2 q = 5 AIC = 210.9272 BIC = 223.5082 MASE = 0.5820008
## p = 3 q = 1 AIC = 228.9066 BIC = 238.232 MASE = 0.7135432
## p = 3 q = 2 AIC = 228.9529 BIC = 239.6106 MASE = 0.6884519
## p = 3 q = 3 AIC = 228.2994 BIC = 240.2892 MASE = 0.6559889
## p = 3 q = 4 AIC = 218.111 BIC = 231.0693 MASE = 0.5849322
## p = 3 q = 5 AIC = 212.9123 BIC = 226.7513 MASE = 0.582732
## p = 4 q = 1 AIC = 223.1523 BIC = 233.519 MASE = 0.6644883
## p = 4 q = 2 AIC = 223.1783 BIC = 234.8409 MASE = 0.6501574
## p = 4 q = 3 AIC = 222.318 BIC = 235.2764 MASE = 0.6113809
## p = 4 q = 4 AIC = 219.5892 BIC = 233.8434 MASE = 0.5623477
## p = 4 q = 5 AIC = 214.5814 BIC = 229.6786 MASE = 0.5641539
## p = 5 q = 1 AIC = 218.0365 BIC = 229.3593 MASE = 0.6807474
## p = 5 q = 2 AIC = 218.2282 BIC = 230.8091 MASE = 0.6655148
## p = 5 q = 3 AIC = 217.4182 BIC = 231.2573 MASE = 0.6199229
## p = 5 q = 4 AIC = 214.3612 BIC = 229.4584 MASE = 0.5551226
## p = 5 q = 5 AIC = 216.3587 BIC = 232.7139 MASE = 0.5540506
```

As per the above output, ARDL(5,5) is the best based on the AIC, BIC and MASE values. Using the finiteDLMauto function, all the other predictors were replaced to check the optimum p & q values. For all the predictor variables, it was found that optimum output is given by ARDL(5,5) only. **Thus, for all the autoregressive DLMs, p & q values will be taken as (5,5).**

Fitting ARDL(5,5) for all combination of predictors:

As models without intercept and models with intercept were giving exactly same model, no models without intercept was fitted for ARDL method.

```
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)
```

Comparison based on AIC,BIC and MASE:

```
# Sorting based on AIC, BIC and MASE

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

```
##                              df      AIC
## model.radiationardl$model   13 206.4543
## model.rainfallardl$model    13 207.6224
## model.humidityardl$model    13 208.0520
## model.temperatureardl$model 13 216.3587
```

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

```
##                              df      BIC
## model.radiationardl$model   13 222.8096
## model.rainfallardl$model    13 223.9776
## model.humidityardl$model    13 224.4073
## model.temperatureardl$model 13 232.7139
```

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

arrange(Maseardl,MASE)
```

```
##                          n      MASE
## model.rainfallardl    26 0.4143799
## model.radiationardl   26 0.4516374
## model.humidityardl    26 0.4860361
## model.temperatureardl 26 0.5540506
```

From the AIC and BIC values, model with only radiation as predictor is the best model. While, based on MASE, the model with rainfall as predictor is the best model. Let’s analyze the latter model as it is based on MASE.

Analyzing ARDL(5,5) with rainfall as predictor:

```
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
## -17.8733  -3.1317   0.5946   2.4862  19.9177
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 520.66772  217.05224   2.399   0.0309 *
## X.t         -10.85300    7.54032  -1.439   0.1720
## X.1          -9.97784    8.28432  -1.204   0.2484
## X.2           4.04690    7.32843   0.552   0.5895
## X.3          -1.78521    7.54794  -0.237   0.8165
## X.4          -3.47024    7.24015  -0.479   0.6391
## X.5           1.15055    6.98861   0.165   0.8716
## Y.1          -0.08942    0.25641  -0.349   0.7325
## Y.2           0.47510    0.22326   2.128   0.0516 .
## Y.3           0.09625    0.25339   0.380   0.7098
## Y.4          -0.68063    0.29039  -2.344   0.0344 *
## Y.5          -0.30712    0.33475  -0.917   0.3744
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.84 on 14 degrees of freedom
## Multiple R-squared:  0.5502, Adjusted R-squared:  0.1968
## F-statistic: 1.557 on 11 and 14 DF,  p-value: 0.2153
```

From the above summary, we can generate following insights about the model.rainfallardl-

1.The model model.rainfallardl is a poorly fitted model and almost all lags are insignificant except X.t and Y.4 and the intercept.

2. For lags X.t,X.1,X.3,X.4,Y.1,Y.4 & Y.5, the flowering is earlier as FFD values are lower(negative coefficients) while for intercept & other remaining lags flowering is later as they have positive coefficient values.
3. The value of R2 is 0.5502. This means that model.rainfallardl is able to explain (55%) variation in the dependent variable (FFD).
4. The residuals have maximum value of 19.9177 and minimum value of -17.8733 with residual standard error (RSE) as 10.84.
5. As per the F-test of the overall significance of model, the model.rainfallardl is an insignificant model at 5% level of significance despite having a moderate R2 value.

VIF for model.rainfallardl:

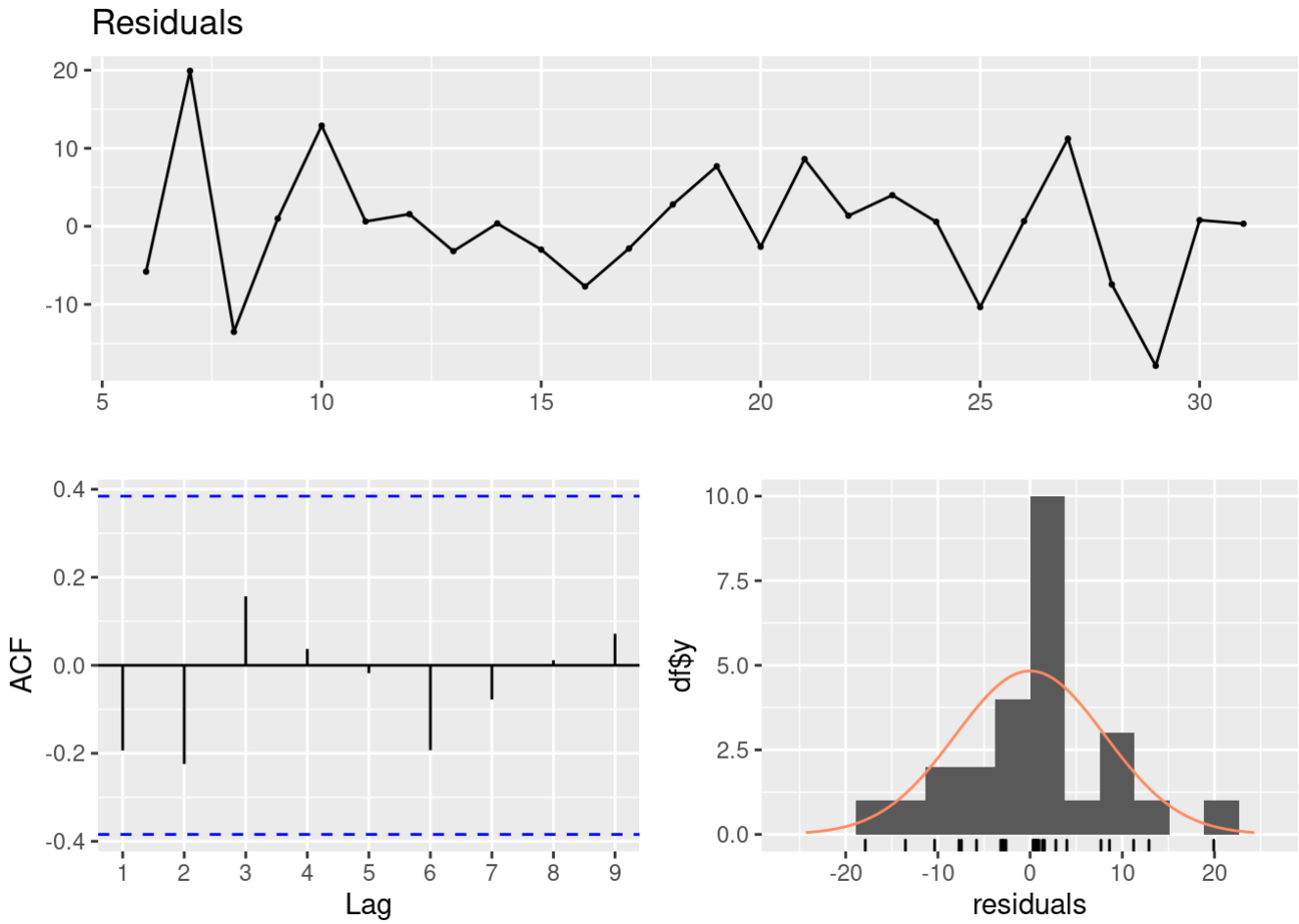
```
vif(model.rainfallardl$model)
```

```
##       X.t L(X.t, 1) L(X.t, 2) L(X.t, 3) L(X.t, 4) L(X.t, 5) L(y.t, 1) L(y.t, 2)
##  2.003301  2.415482  1.876605  1.963839  1.683333  1.495623  2.046387  1.545780
## L(y.t, 3) L(y.t, 4) L(y.t, 5)
##  1.915730  2.366747  3.127862
```

As VIF is less than 10 for all elements in the model, there is no presence of multicollinearity in the model.

Residual Analysis (Diagnostic checking) of model.rainfallardl:

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



From the above residual plots generated, we can see that residuals seem to be spread randomly without any particular pattern. In ACF plot, no particular lag is significant which shows there is no presence of autocorrelation in the residuals. The residuals do not have normal distribution as per the histogram plot.

As per the diagnostic checking (residual analysis and other tests), autoregressive DLM seems to be a decent fit in comparison to the previous Koyck model. However, the model is insignificant even though it does not face any issue of multicollinearity and explains a decent variation in the dependent variable FFD.

Forecasting using model.rainfallardl

Lets' generate the forecast point intervals and plot the forecast for the next four years using rainfall values-

```
# Generating point forecasts using rainfall values from task2_covariate

forecastFFDpoint3 <- dLagM::forecast(model.rainfallardl,x = c(2.27,2.38,2.26,2.27) ,h = 4)

forecastFFDpoint3 <- round(forecastFFDpoint3$forecasts,2)

# Generating prediction intervals

forecastFFDinterval3 <- forecast(model = model.rainfallardl, x = c(2.27,2.38,2.26,2.27) ,
                h = 4 , interval = TRUE)

round(forecastFFDinterval3$forecasts,2)
```

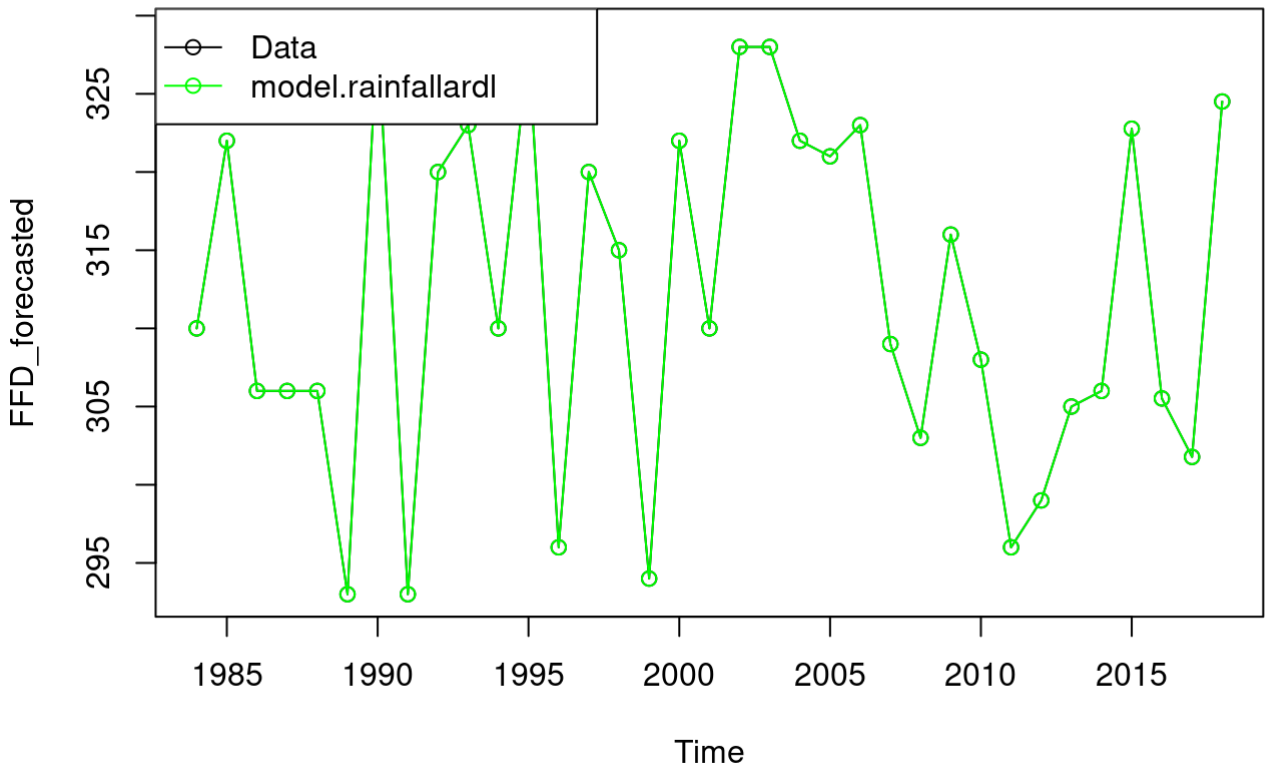
```
##   95% LB Forecast 95% UB
## 1 302.90   322.77 343.64
## 2 283.30   305.52 328.90
## 3 279.31   301.78 326.30
## 4 294.11   324.51 352.55
```

The 95% prediction intervals and predictions generated for the ARDL model show that FFD values will fluctuate a lot and will follow an increasing and decreasing pattern for the next four years. It will increase to 322.77 in 2015 and will have a sharp decrease in the next two years. It will again regain its upward pattern by reaching the value of 324.51 by 2018. The prediction intervals are also not very wide indicating the forecast values are reliable.

Plotting forecast for model.rainfallardl:

```
plot(ts(c(as.vector(FFD),forecastFFDpoint3),start=1984),type="o",col="black", ylab="FFD_forec
asted",
main="Figure 42: FFD four year ahead predicted values from 2015-2018")
lines(ts(c(as.vector(FFD),forecastFFDpoint3),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 42: FFD four year ahead predicted values from 2015-2018



The model model.rainfallardl is perfectly in line with the FFD values from 1984-2014. Thus, the predictions generated by the ARDL model are reliable and accurate even though they are a little unrealistic in terms of the high fluctuations predicted for the next four years.

Modelling using Dynamic Linear Models

The Dynamic Linear models are often considered suitable for forecasting. As there are no trend or seasonality in the FFD series, we will only fit models with univariate independent variables (with & without intercept). Let's fit a few dynlm models-

Fitting different 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))
```

Let’s now compare all these models in terms of MASE-

Comparing models based on MASE:

```
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(dynamicmodel3)      28 0.6615623
## lm(rainfalldynlm7)     28 0.6840155
## lm(humiditydynlm7)     28 0.6936275
## lm(rainfalldynlm6)     28 0.6948640
## lm(temperaturedynlm6) 28 0.7084813
## lm(rainfalldynlm1)     29 0.7104169
## lm(radiationdynlm1)    29 0.7208351
## lm(dynamicmodel2)      29 0.7219351
## lm(radiationdynlm7)    28 0.7228539
## lm(temperaturedynlm1) 29 0.7243203
## lm(rainfalldynlm2)     31 0.7283036
## lm(humiditydynlm1)     29 0.7304124
## lm(temperaturedynlm4) 30 0.7344272
## lm(rainfalldynlm4)     30 0.7345684
## lm(rainfalldynlm5)     29 0.7355722
## lm(radiationdynlm6)    28 0.7372073
## lm(rainfalldynlm)      30 0.7402171
## lm(radiationdynlm2)    31 0.7448205
## lm(humiditydynlm6)     28 0.7501637
## lm(radiationdynlm)     30 0.7508961
## lm(humiditydynlm5)     29 0.7511569
## lm(radiationdynlm5)    29 0.7513022
## lm(humiditydynlm2)     31 0.7538879
## lm(temperaturedynlm7) 28 0.7547584
## lm(temperaturedynlm2) 31 0.7548499
## lm(humiditydynlm4)     30 0.7554706
## lm(temperaturedynlm)  30 0.7571374
## lm(radiationdynlm4)    30 0.7582353
## lm(humiditydynlm)      30 0.7591338
## lm(dynamicmodel1)      30 0.7677606
## lm(temperaturedynlm5) 29 0.7706397
## lm(humiditydynlm3)     31 0.7815254
## lm(radiationdynlm3)    31 0.8083308
## lm(temperaturedynlm3) 31 1.0326703
## lm(rainfalldynlm3)     31 3.1418931
```

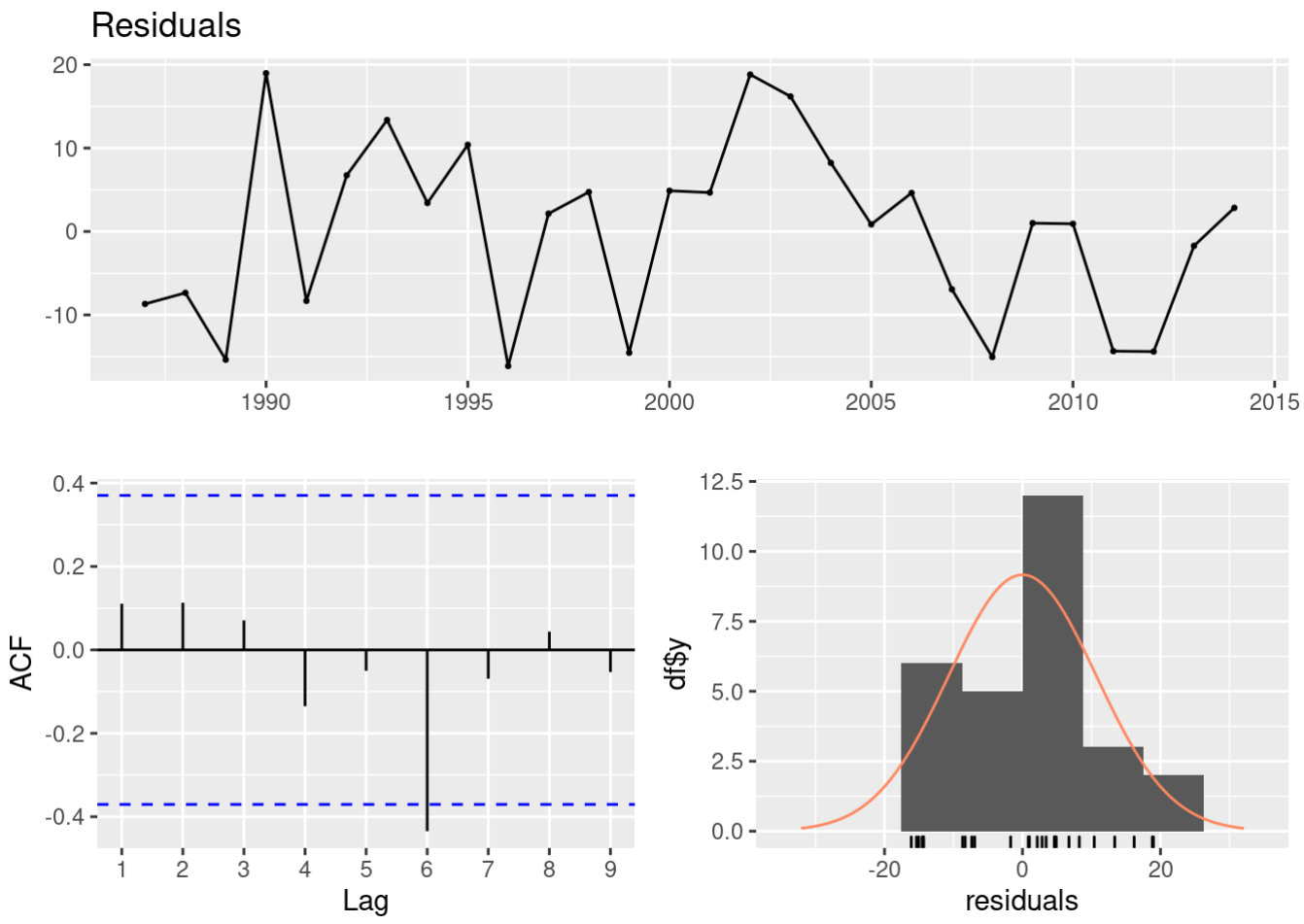
From the above results, let's analyze the dynamicmodel3 model as it has the lowest MASE value out of all the models fitted-

Analyzing dynamicmodel3 using summary & residual analysis:

```
# Generating summary of model

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
## -16.123  -8.405   1.571   5.350  18.972
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   155.1715   100.6201   1.542    0.136
## L(FFD, k = 1)  -0.1277     0.1939  -0.659    0.516
## L(FFD, k = 2)   0.3167     0.1866   1.697    0.103
## L(FFD, k = 3)   0.3117     0.1982   1.573    0.129
##
## Residual standard error: 11.34 on 24 degrees of freedom
## Multiple R-squared:  0.1718, Adjusted R-squared:  0.06826
## F-statistic: 1.659 on 3 and 24 DF,  p-value: 0.2023
```

```
checkresiduals(dynamicmodel3)
```



```
##
##  Breusch-Godfrey test for serial correlation of order up to 7
##
## data:  Residuals
## LM test = 10.394, df = 7, p-value = 0.1673
```

From the above summary of dynamicmodel3 model, we can generate following insights-

1. All the terms in the model are insignificant at 5% level of significance.
2. The value of R2 is very poor (17.18%). This means that dynamic model fitted is able to explain only 17.18% variation in the dependent variable (FFD).
3. The residuals have maximum value of 18.972 and minimum value of -16.123 with residual standard error (RSE) as 11.34 which is not better in comparison to previous models.
4. As per the F-test of the overall significance of model, the dynamicmodel3 is an insignificant model at 5% level of significance.
5. In comparison to other DLM models, the dynamic model generated poor summary statistics.

Residual Analysis:The BG test indicate no serial correlation is there in the residuals.The residuals plot indicate that residuals are randomly distributed. In the ACF plot, there is one not very obvious significant lag. As a result, no obvious autocorrelation found in the residuals. The histogram shows residuals do not seem to follow normal distribution.

VIF for dynamicmodel3:

```
vif(dynamicmodel3)
```

```
## L(FFD, k = 1) L(FFD, k = 2) L(FFD, k = 3)
##      1.089835      1.022428      1.098747
```

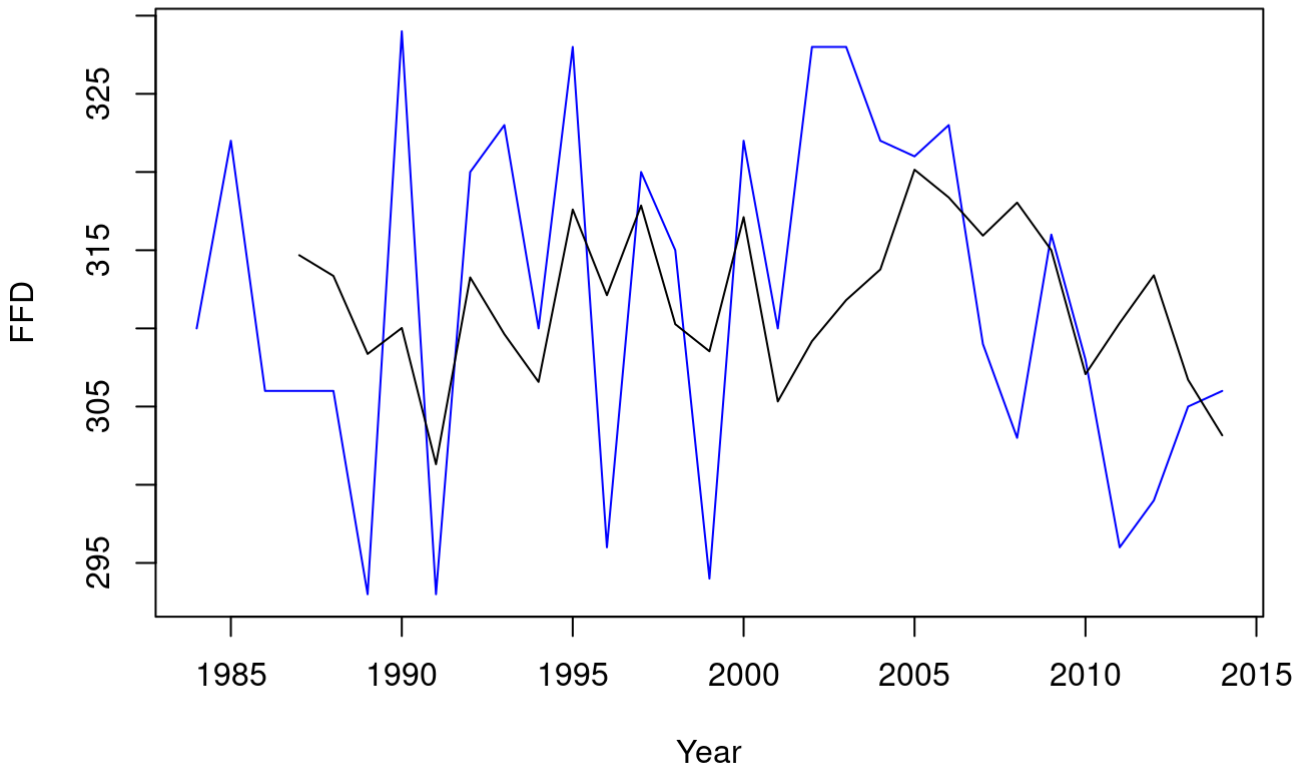
As VIF is less than 10 for all elements in the model, there is no presence of multicollinearity in the model

Interpretation: The model dynamicmodel3 obtained using Dynamic linear method is not a significant model in terms of R2 & F-test even though it has best MASE accuracy score than other dynlm models fitted.

Forecasting using dynaminmodel3

```
# Checking fit of dynamicmodel3 with FFD series

par(mfrow=c(1,1))
plot(FFD,ylab='FFD',xlab='Year',main = "Figure 61: Fit with original FFD", col="blue")
lines(dynamicmodel3$fitted.values,col="black")
```

Figure 61: Fit with original FFD

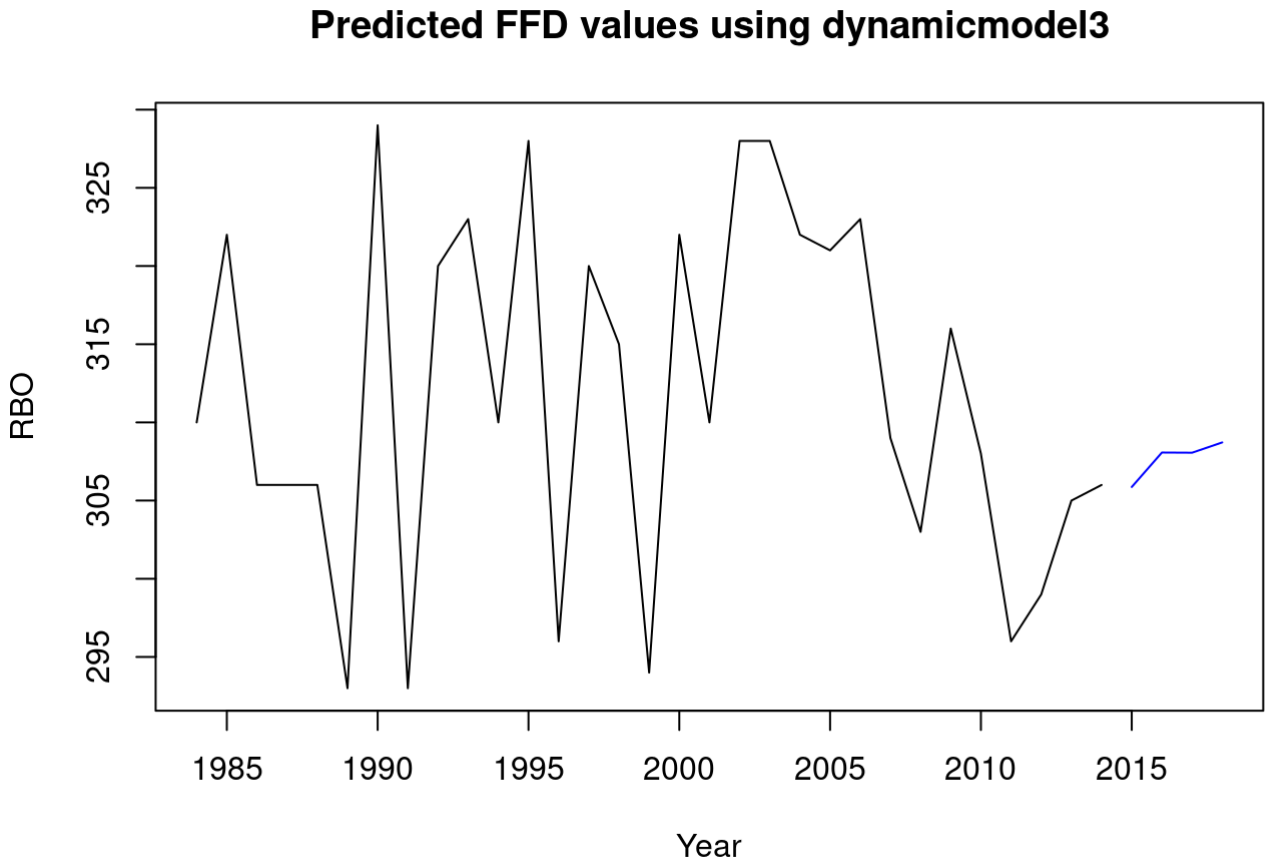


The dynamicmodel3 does not seem to be a good fit for the FFD series prediction as the fitted values deviate a lot from the original FFD series.

Plotting forecasts:

Let's plot the forecast using 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 = "Predicted FFD values using dynamicmodel3")
lines(ts(FFD.frc[(n+1):(n+q)],start=c(2015)),col="blue")
```




```
# Checking forecast

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

```
## [1] 305.8690 308.0724 308.0612 308.7196
```

As per the predicted values plotted & computed, we see that in the next four years, the forecast values will not change much and will lie between 305-308. However, as we know that the dynamic model did not fit the original series well, thus, these values predicted are not reliable.

Modelling using Exponential Smoothing

As FFD series have no trend or seasonality present, we can only fit a selective of models using exponential smoothing methods.

Fitting Simple ES model:

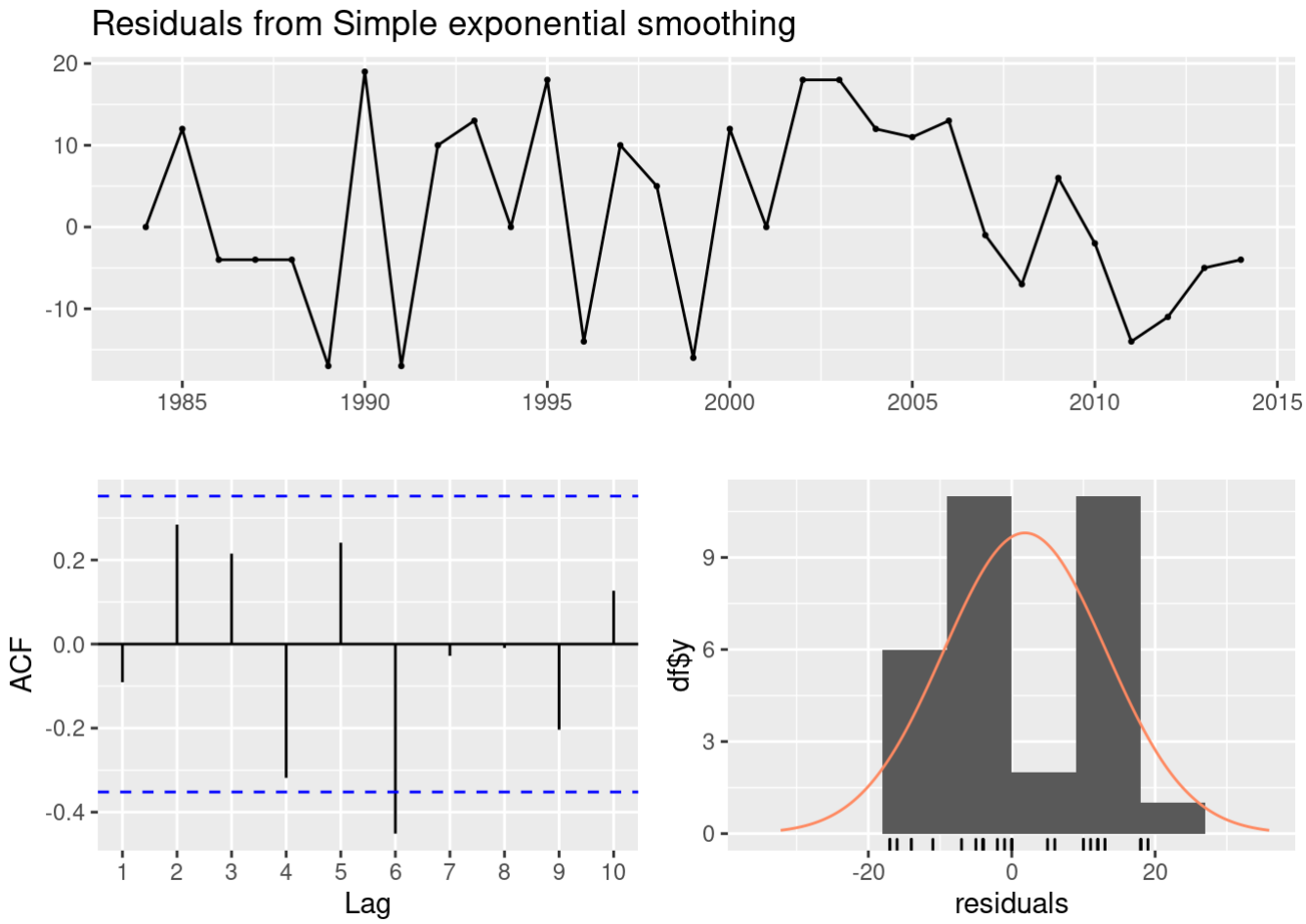
```
# Let the R software compute alpha & beta values

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

Summary & Residual Analysis of model.FFDses:

```
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
##
##   Initial states:
##     l = 310
##
##   sigma:  11.3237
## Error measures:
##                    ME     RMSE      MAE       MPE     MAPE      MASE        ACF1
## Training set 1.83871 11.32368 9.580645 0.4611293 3.06345 0.7446097 -0.09083322
##
## Forecasts:
##      Point Forecast    Lo 80    Hi 80   Lo 95   Hi 95
## 2015            310 295.4881 324.5119 287.806 332.194
## 2016            310 295.4881 324.5119 287.806 332.194
## 2017            310 295.4881 324.5119 287.806 332.194
## 2018            310 295.4881 324.5119 287.806 332.194
```

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



```
##
##  Ljung-Box test
##
## data:  Residuals from Simple exponential smoothing
## Q* = 19.27, df = 4, p-value = 0.0006954
##
## Model df: 2.   Total lags used: 6
```

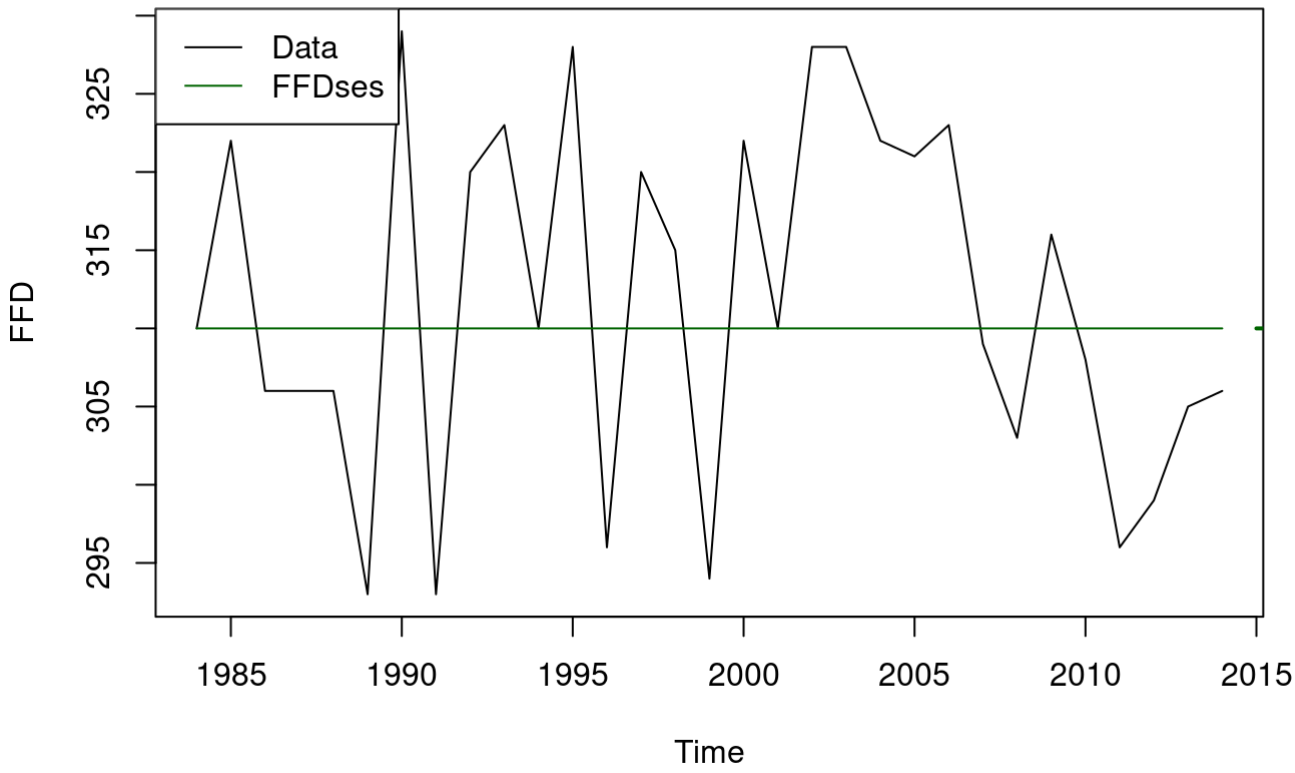
From the above summary and residuals plots of model.FFDses model we can generate following insights-

1. The MASE value of the model 0.7446097 and other accuracy measures like RMSE, MAPE are 11.32368 & 3.06345 respectively.
2. The smoothing parameters are: alpha = 0.
3. As per the Ljung-Box test, p-value is less than 5% level of significance which indicates that autocorrelation is present in the residuals.
4. There is only one significant lag in the ACF plot which indicate some kind of autocorrelation is present in the residuals.
5. The residuals do not seem to follow any trend or pattern and are randomly spread.
6. Normality problems are present in the model as residuals do not seem to follow normal distribution.
7. The point forecasts are also displayed for the model along with 80% and 95% confidence intervals.

Forecasting using FFDses

```
plot(FFD, fcol = "black", main = "Figure 43: FFD series with four years ahead forecasts", yla
b = "FFD")
lines(fitted(model.FFDses), col = "darkgreen")
lines(model.FFDses$mean, col = "darkgreen", lwd = 2)
legend("topleft", lty = 1, col = c("black", "darkgreen"), c("Data", "FFDses"))
```

Figure 43: FFD series with four years ahead forecasts




```
# Fitted values

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     287.806            310     332.194
## 2016     287.806            310     332.194
## 2017     287.806            310     332.194
## 2018     287.806            310     332.194
```

Interpretation: The Simple Exponential Smoothing model is not a good fit model as seen from the fitted & predicted values. The model is following a constant line & generate very constant predictions for the next 4 years. Thus, this model is not at all a good fit model for predicting the FFD values for the next 4 years.

Modelling with State Space Models

As there is no trend or seasonality in the model,we can only fit models with multiplicative and additive error. Let’s fit the models and compare.

Fitting ETS(M,N,N) with multiplicative error:

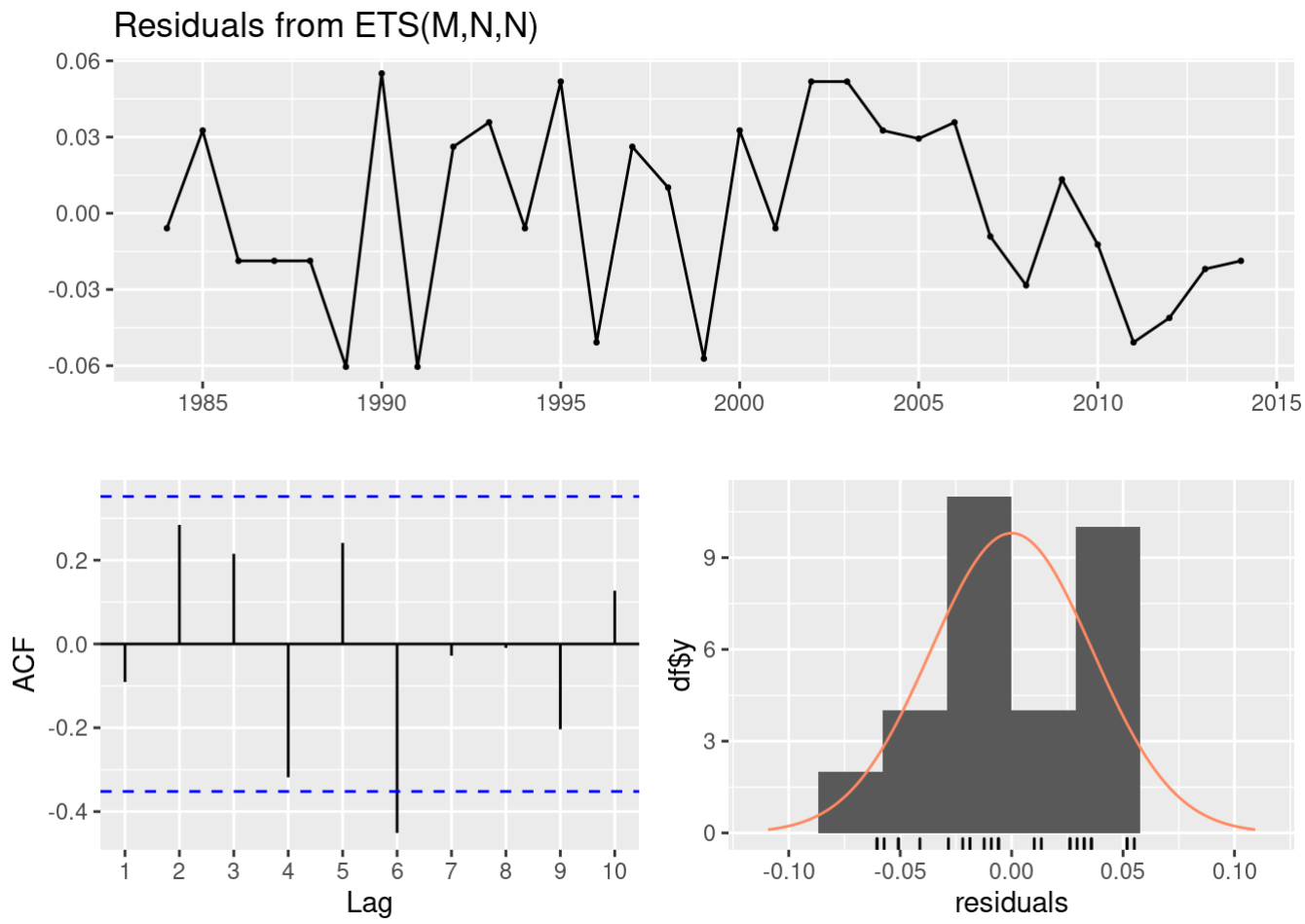
```
FFDets1 <- ets(FFD, model="MNN")
```

Analyzing Model ETS(M,N,N):

```
summary(FFDets1)
```

```
## ETS(M,N,N)
##
## Call:
##  ets(y = FFD, model = "MNN")
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 311.8397
##
##   sigma:  0.037
##
##      AIC     AICc      BIC
## 262.0961 262.9850 266.3980
##
## Training set error measures:
##                        ME     RMSE      MAE         MPE    MAPE      MASE
## Training set -0.001648532 11.17396 9.759322 -0.1298033 3.13927 0.7584965
##                      ACF1
## Training set -0.09083193
```

```
checkresiduals(FFDets1)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,N)
## Q* = 19.27, df = 4, p-value = 0.0006955
##
## Model df: 2.   Total lags used: 6
```

From the above summary and residuals plots of FFDets1 we can generate following insights-

1. The MASE value of the model 0.7584965 and other accuracy measures like AIC, AICc and BIC are in the range of 262-266. This is not much better than previous models fitted. Hence, this model is a decent fit model but not good enough.
2. The smoothing parameters are: alpha = 1e-04.
3. As per the Ljung-Box test, p-value is less than 5% significance level which indicates that serial correlation is there in the residuals.
4. The residuals are randomly distributed and in the ACF plot there is one significant lag which supports Ljung-Box test results.
5. Normality problems are present in the model as residuals do not seem to follow normal distribution.

Fitting ETS(A,N,N) with additive error:

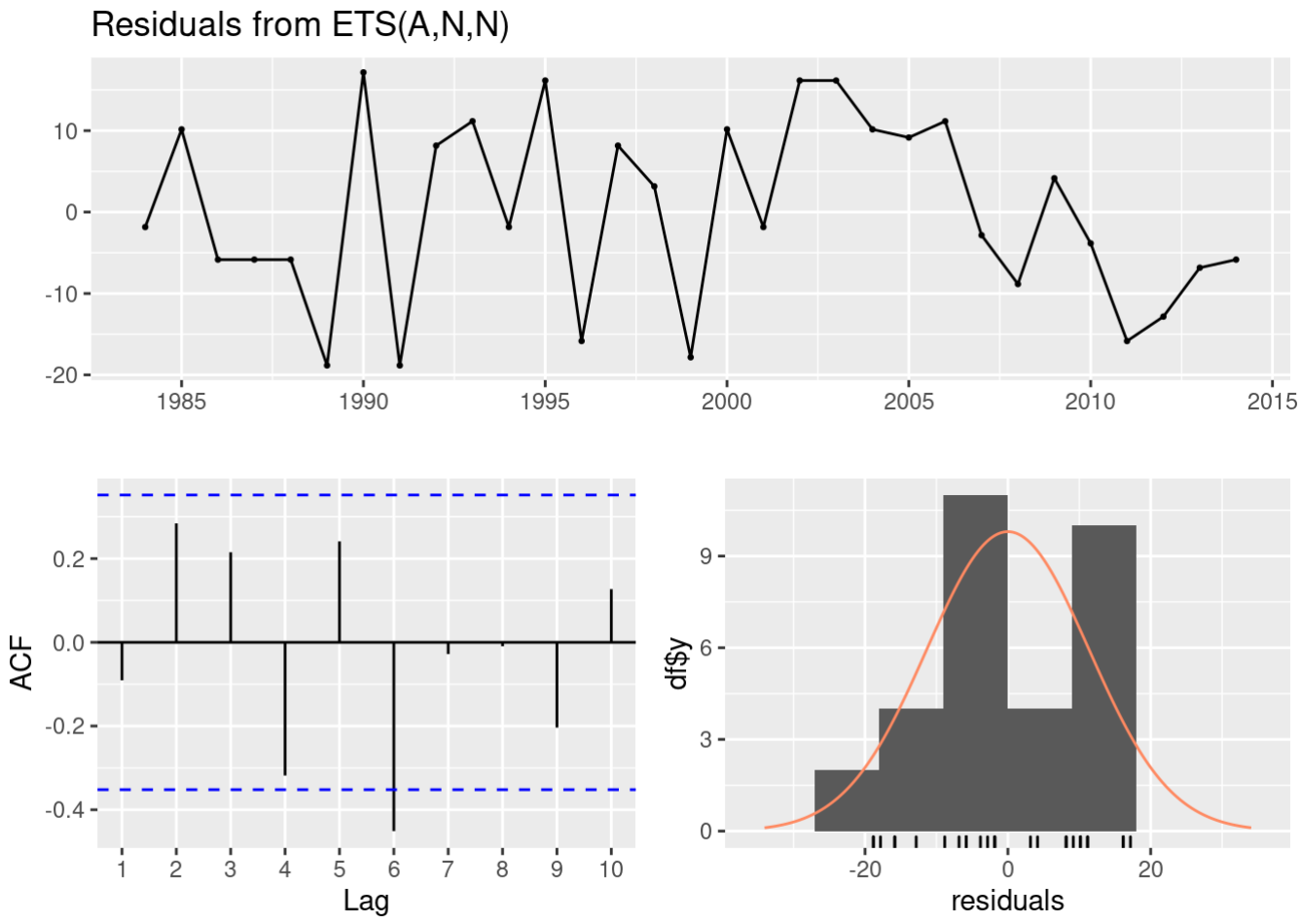
```
FFDets2 <- ets(FFD, model="ANN")
```

Analyzing Model ETS(A,N,N):

```
summary(FFDets2)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = FFD, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 311.8397
##
##   sigma:  11.5528
##
##      AIC     AICc      BIC
## 262.0960 262.9848 266.3979
##
## Training set error measures:
##                       ME     RMSE      MAE         MPE     MAPE      MASE
## Training set -0.001648532 11.17396 9.759322 -0.1298033 3.13927 0.7584965
##                     ACF1
## Training set -0.09083193
```

```
checkresiduals(FFDets2)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,N)
## Q* = 19.27, df = 4, p-value = 0.0006955
##
## Model df: 2.   Total lags used: 6
```

From the above summary and residuals plots of FFDets2 we can generate following insights-

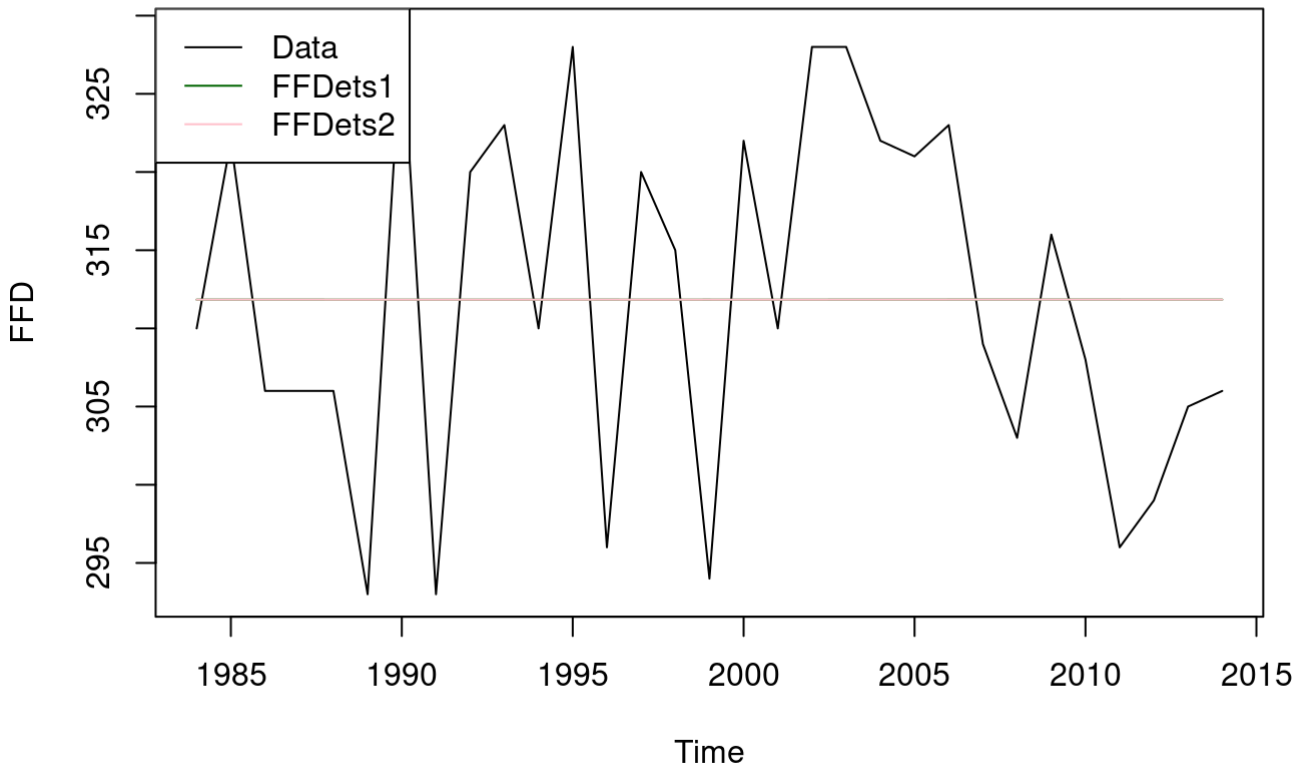
1. The MASE value of the model 0.7584965 and other accuracy measures like AIC, AICc and BIC are in the range of 262-266.This is exactly similar to previous model fitted. Hence, this model is a decent fit model but not good enough.
2. The smoothing parameters are: alpha = 1e-04.
3. As per the Ljung-Box test, p-value is less than 5% significance level which indicates that serial correlation is there in the residuals.
4. The residuals are randomly distributed and in the ACF plot there is one significant lag which supports Ljung-Box test results.
5. Normality problems are present in the model as residuals do not seem to have normal distribution.

Intepretation: The MASE value for both ETS models fitted is exactly similar (0.7584965) but its not the lowest in comparison to other models. Hence, the model is not a good fit model for predicting FFD.

Forecasting using FFDets1 & FFDets2

```
plot(FFD, fcol = "black", main = "Figure 44: FFD series with four years ahead forecasts", yla
b = "FFD")
lines(fitted(FFDets1), col = "darkgreen")
lines(FFDets1$mean, col = "darkgreen", lwd = 2)
lines(fitted(FFDets2), col = "pink")
lines(FFDets2$mean, col = "pink", lwd = 2)
legend("topleft", lty = 1, col = c("black", "darkgreen","pink"), c("Data", "FFDets1","FFDets
2"))
```

Figure 44: FFD series with four years ahead forecasts




```
# Fitted values

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     287.806            310     332.194
## 2016     287.806            310     332.194
## 2017     287.806            310     332.194
## 2018     287.806            310     332.194
```

Intepretation: The MASE value for both ETS models fitted is exactly similar (0.7584965) but its not the lowest in comparison to other models. Moreover, the forecasts generated for ETS models is inaccurate as the model does not fit the data at all. Hence, the model is not a good fit model for predicting FFD.

Summary Table Task 2

Let's compare all the models fitted so far and compute the best model based on MASE score-

```
# Generating table

final_table <- data.frame(Model = c("model.finite2","model.radiationpoly","model.radiationkoy
ck","model.rainfallardl","dynamicmodel3","model.FFDses","model.FFDets1","model.FFDets2"), MAS
E= c(0.2319041,0.6355363,0.7540611,0.4143799,0.6549642,0.7446097,0.7584965,0.7584965))


# Comparing MASE score

arrange(final_table,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
```

As per the above summary, model.finite2 (with radiation as predictor) is the best model in terms of MASE for forecasting the next four years FFD values.

Conclusion

The task 2 analysis was successfully done on all the five series present in FFD.csv file. As the five series was yearly (frequency < 2),they could not be decomposed for deeper analysis. The five series were analyzed and different modelling methods(DLM,koyck,polyDLM,dynlm,ES,State space models) were used to fit uni variate models with only one climate indicator at a time. Using different accuracy measures like R2, MASE, AIC & BIC, best model from each method was chosen to generate forecasting. **Radiation & rainfall** were found to be the influential climate indicators in all the best methods that were used to predict the four year ahead values. The finite, poly, koyck and ARDL & dynlm methods generated accurate/reliable predictions while best models found using Exponential smoothing and ETS were found to be unfit for FFD series. The recommended model/best model out of all the models fitted was model.finite2 with radiation as the predictor in terms of MASE value, R2 value, F-test and.predictions generated for the next four years.

Task 3 Part(a)- Time series analysis & forecast of RBO series

Data Description

From 1983 to 2014, the data from Hudson & Keatley (2021) investigates the impact of long-term climatic changes in Victoria on the relative blooming order similarity of 81 plant species. The annual blooming order was computed using the Rank-based Order similarity metric, and changes in flowering order were evaluated by estimating the similarity between annual flowering order and the flowering order of 1983. (RBO). For the particular year under investigation, the earliest flowering species is rated 1 and the last flowering species is ranked 81.

Because 1983 is the reference year for flowering order FFD rankings, the data ‘RBO.csv’ comprises five time series: the RBO time series of the 81 plant species examined by Hudson & Keatley (2021) and the yearly averaged climatic variables recorded from 1984 to 2014. The time series are 31 in length. The other four variables are-temperature,rainfall, radiation & relhumidity.

Objective & Methodology

The main objective of task 3 is to analyze the RBO and give 3 years ahead forecast for RBO. In part (b) we will use dynamic models to transform the series to include the drought period of Australia and then again predict the three year ahead forecast. Thus, following steps will be performed-

1. Firstly, necessary data prepossessing steps will be performed on all the five series. The data file ‘RBO.csv’ will be checked for missing/unique/unknown values and will be dealt with accordingly. Each series from ‘RBO.csv’ will also be converted to time-series to demonstrate yearly values for all the five series to carry out a time series analysis.
2. The covariate file will also be loaded and will be used during forecasting.
3. The next step will be to check and understand the nature/characteristics of all the time series by plotting respective time series plots and plotting them together. All the five series will also be checked for stationarity through ACF ,PACF plots and Dicker-Fuller Unit tests.
4. The correlation between each series will be checked by plotting a correlation matrix for all of them.
5. Further, different distributed lag models will also be fitted on the series.
6. Different type of DLMs, dynlms will also be fitted depending upon the results generated from previous steps.
7. The best model/models based on R squared, AIC, BIC, MASE etc. will be selected for predicting the values of three years ahead RBO values.
8. For part(b), we will fit dynamic linear models(dynlm) to accommodate the drought period which affected RBO. This will be done by transforming the series to include that period’s affect and then the dynlm models will be fitted.
9. Finally, forecasting will be performed again on the RBO series for the next three years.

Data Prepossessing

Before executing the task 3, we need to perform some necessary data prepossessing steps to make the datasets ready for analysis.

1. Importing dataset into Rstudio:

```
# Importing dataset `RBO' & `covariate x values for task 3`

RBO_series <- read_csv("/cloud/project/RBO .csv",show_col_types = FALSE)
```

```
## New names:
## * `` -> ...7
## * `` -> ...8
## * `` -> ...9
## * `` -> ...10
## * `` -> ...11
## * ...
```

```
task3_covariate <- read_csv("/cloud/project/Covariate 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.
```

```
# Checking they has been loaded successfully

head(RBO_series)
```

```
## # A tibble: 6 × 14
##    Year   RBO Temperature Rainfall Radiation RelHumidity ...7  ...8  ...9  ...10
##   <dbl> <dbl>       <dbl>    <dbl>     <dbl>       <dbl> <lgl> <lgl> <lgl> <lgl>
## 1  1984 0.755        18.7     2.49      14.9        54.6 NA    NA    NA    NA
## 2  1985 0.741        19.3     2.48      14.7        55.0 NA    NA    NA    NA
## 3  1986 0.842        18.6     2.42      14.5        55.0 NA    NA    NA    NA
## 4  1987 0.748        19.1     2.32      14.7        53.9 NA    NA    NA    NA
## 5  1988 0.798        20.4     2.47      14.7        53.1 NA    NA    NA    NA
## 6  1989 0.794        19.6     2.74      14.8        55.4 NA    NA    NA    NA
## # … with 4 more variables: ...11 <lgl>, ...12 <lgl>, ...13 <lgl>, ...14 <lgl>
```

```
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        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
```

```
# Checking class of both 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"
```

1.A 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        54.6
## 2  1985 0.741        19.3     2.48      14.7        55.0
## 3  1986 0.842        18.6     2.42      14.5        55.0
## 4  1987 0.748        19.1     2.32      14.7        53.9
## 5  1988 0.798        20.4     2.47      14.7        53.1
## 6  1989 0.794        19.6     2.74      14.8        55.4
```

2. Checking for missing and special values:

Checking the datasets for missing values-

```
# Checking for missing values in both series

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

```
##        Year         RBO Temperature    Rainfall   Radiation RelHumidity
##           5           5           5           5           5           5
```

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

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

As per the above output there are missing values in both the files. We will remove the missing values in each of the files.

Now checking for special values-

```
#Checking for special values in both series

is.specialorNA <- function(x){
  if (is.numeric(x)) (is.infinite(x) | is.nan(x) | is.na(x))
}

sapply(RBO_series, function(x) sum( is.specialorNA(x) ))
```

```
##        Year         RBO Temperature    Rainfall   Radiation RelHumidity
##           5           5           5           5           5           5
```

```
sapply(task3_covariate, function(x) sum( is.specialorNA(x) ))
```

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

As per the above outputs, there are missing/special values of NA in both the data files. Let’s remove those-

3.A Removing NA values in task3_covariate

```
# Removing NA values

RBO_series <- na.omit(RBO_series)
task3_covariate <- na.omit(task3_covariate)


# Checking missing values have been removed

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
##           0           0           0           0           0
```

As per above output, missing/NA values have been removed from both the RBO_series and task3_covariate.

4. Converting each variable to time series and storing separately:

The five different series will be extracted from RBO_series. Then they will converted to time series and stored in separate datasets.The dataset RBO_series will also be converted 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 'tem'

tem <- 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 'rad'

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


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

hum <- 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"
```

```
class(tem)
```

```
## [1] "ts"
```

```
class(rain)
```

```
## [1] "ts"
```

```
class(hum)
```

```
## [1] "ts"
```

```
class(rad)
```

```
## [1] "ts"
```

```
class(RBO)
```

```
## [1] "ts"
```

The conversion was successfully done.The RBO_series data was converted to time series and each of the five variables in the RBO_series (except Year) were individually converted to five different series containing time-series data of yearly values of temperature, radiation, rainfall, humidity and RBO respectively.

Understanding Nature of Series

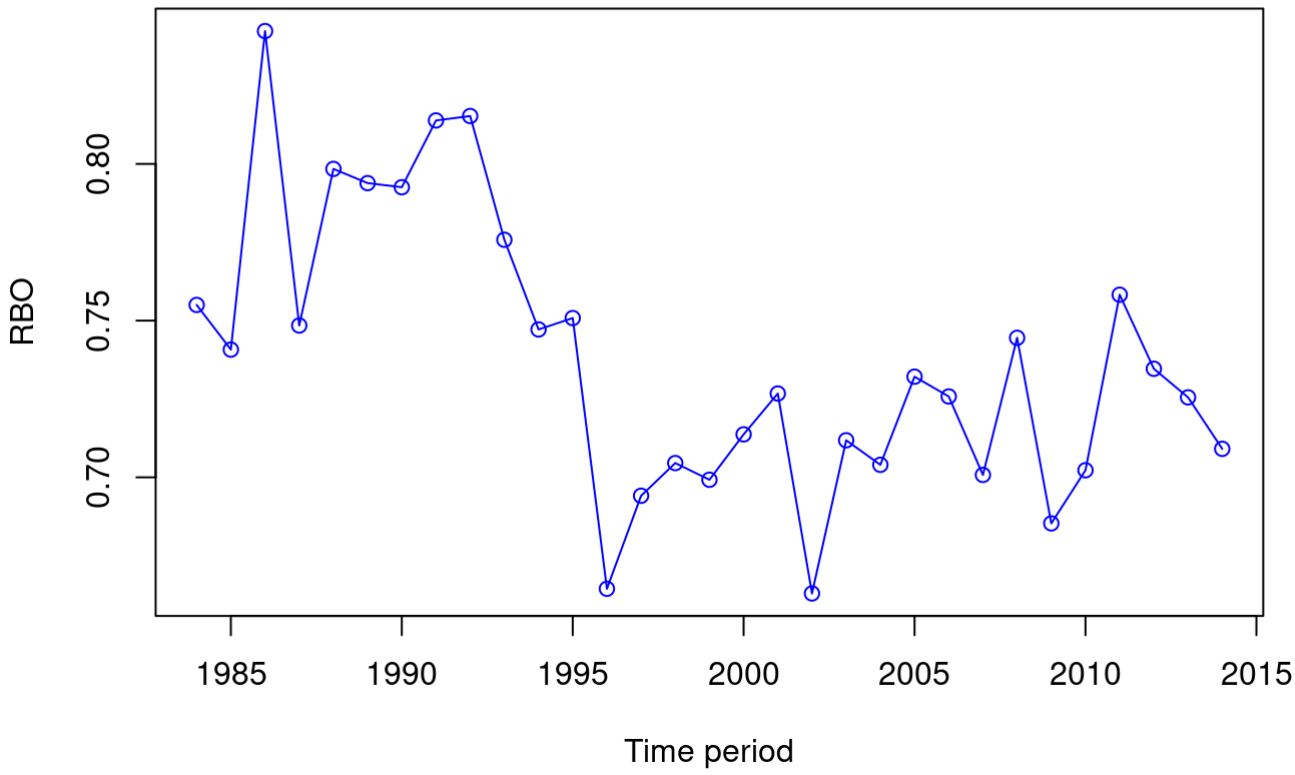
In order to understand the nature of each of the five series, we will plot a time-series plot for each of the series separately and together. We will also calculate correlation of the dataset `climate2` .

Time-series plot of RBO:

A time-series plot of the yearly values of RBO was plotted using plot function.

```
plot(RBO, ylab='RBO', xlab='Time period', type='o', col='blue', main = 'Figure 45: Time-serie
s plot of change in yearly  values of RBO')
```

Figure 45: Time-series plot of change in yearly values of RBO



Following observations were seen from the time series plot of the RBO series-

1.) Trend : There seems to be presence of a downward(negative) trend in the series. However, it is not very obvious.

2.) Seasonality : No particular seasonality can be seen in the series.

3.) Changing variance : There is presence of change in variance in the series.

4.) Intervention : There are two intervention points around 1996 and before 2005.

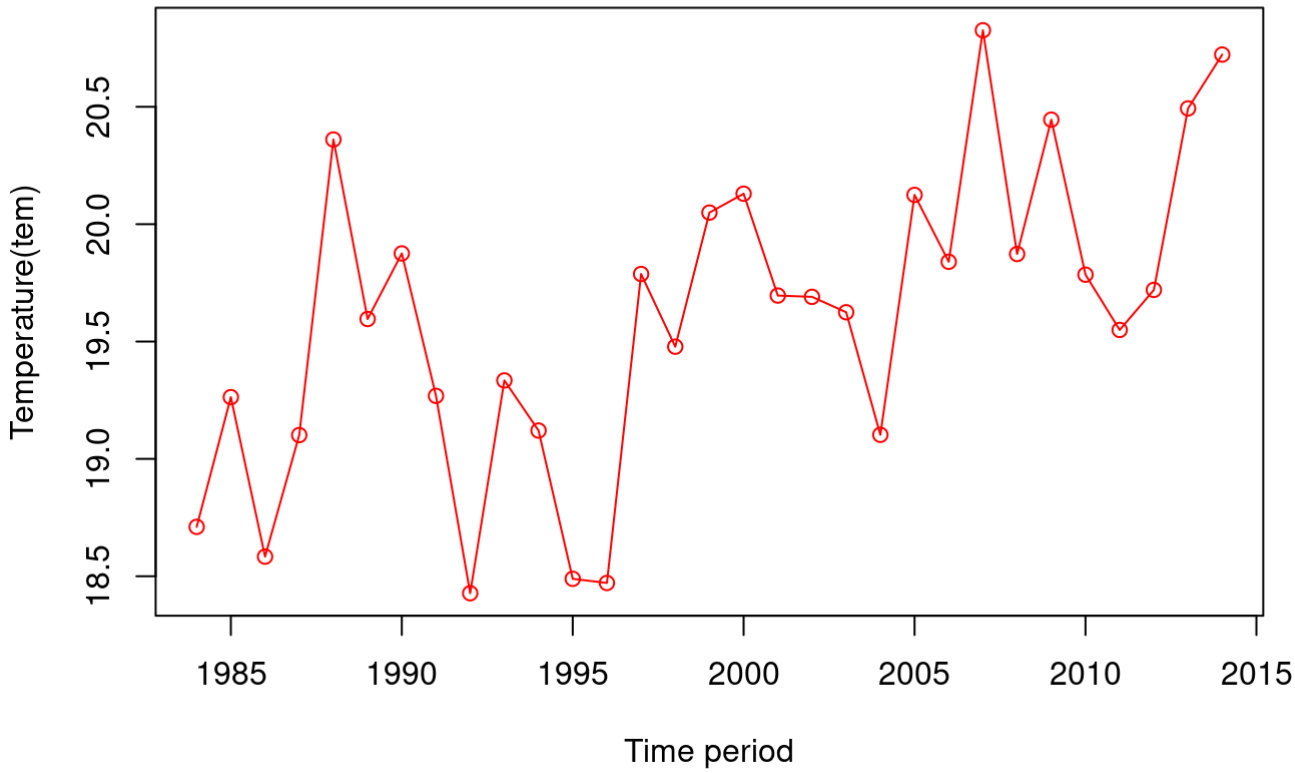
5.) Behavior : Several successive points and fluctuations can be observed along the mean level in the series therefore, a high order AR and MA behavior is seen.

Time-series plot of Temperature(tem):

A time-series plot of the yearly values of temperature(tem) was plotted using plot function.

```
plot(tem, ylab='Temperature(tem)', xlab='Time period', type='o', col='red', main = 'Figure 4
6: Time-series plot of change in yearly values of temperature')
```

Figure 46: Time-series plot of change in yearly values of temperature



Following observations were seen from the time series plot of the tem series-

1.) Trend : There seems to be presence of an upward(positive) trend in the series from 1984 to 2014.

2.) Seasonality : No particular seasonality can be seen in the series.

3.) Changing variance : There is presence of change in variance in the tem series.

4.) Intervention : There is no particular intervention point in the tem series.

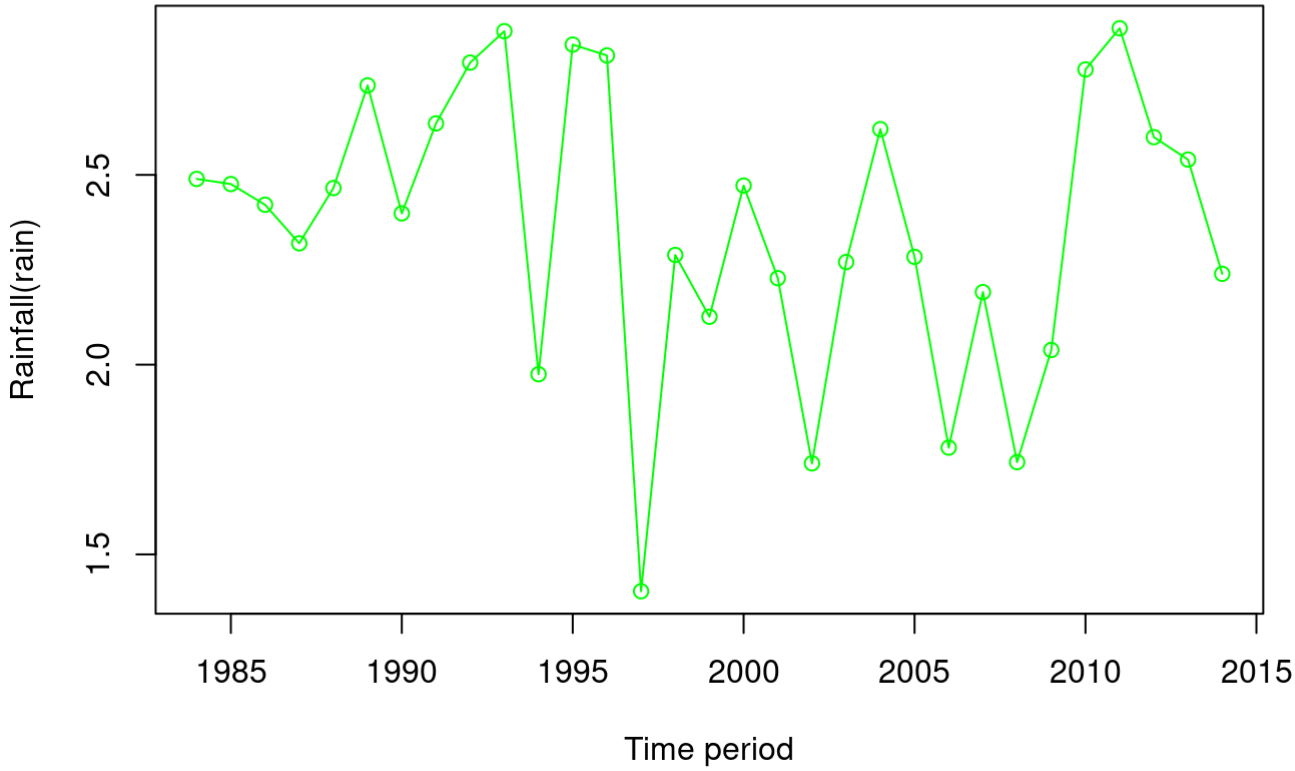
5.) Behavior : Several successive points and fluctuations can be observed along the mean level in the series therefore, a high order AR and MA behavior is seen.

Time-series plot of Rainfall(rain):

A time-series plot of the yearly values of rainfall(rain) was plotted using plot function.

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

Figure 47: Time-series plot of change in yearly values of rainfall



Following observations were seen from the time series plot of the rain series-

1.) Trend : There seems to be no presence of a particular trend in the series.

2.) Seasonality : No particular seasonality can be seen in the series.

3.) Changing variance : There is presence of change in variance in the rain series.

4.) Intervention : There is one major negative intervention point in the rain series between 1995-2000.

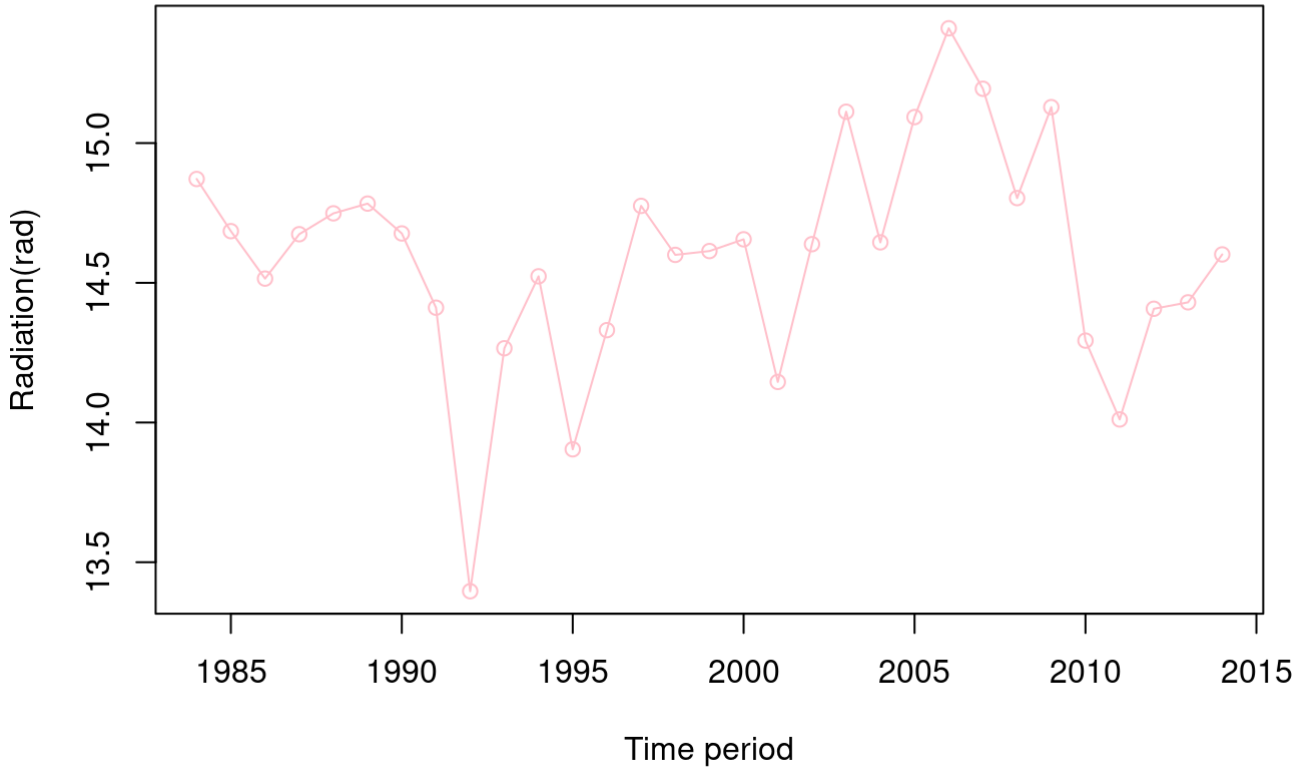
5.) Behavior : Several successive points and fluctuations can be observed along the mean level in the series therefore,a high order AR and MA behavior is seen.

Time-series plot of Radiation(rad):

A time-series plot of the yearly values of radiation(rad) was plotted using plot function.

```
plot(rad, ylab='Radiation(rad)', xlab='Time period', type='o', col='pink', main = 'Figure 48:
Time-series plot of change in yearly values of radiation')
```

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



Following observations were seen from the time series plot of the rad series-

1.) Trend : There seems to be no presence of a particular trend in the series.

2.) Seasonality : No particular seasonality can be seen in the series.

3.) Changing variance : There is presence of change in variance in the rad series.

4.) Intervention : There is one major negative intervention point in the rad series between 1990-1995.

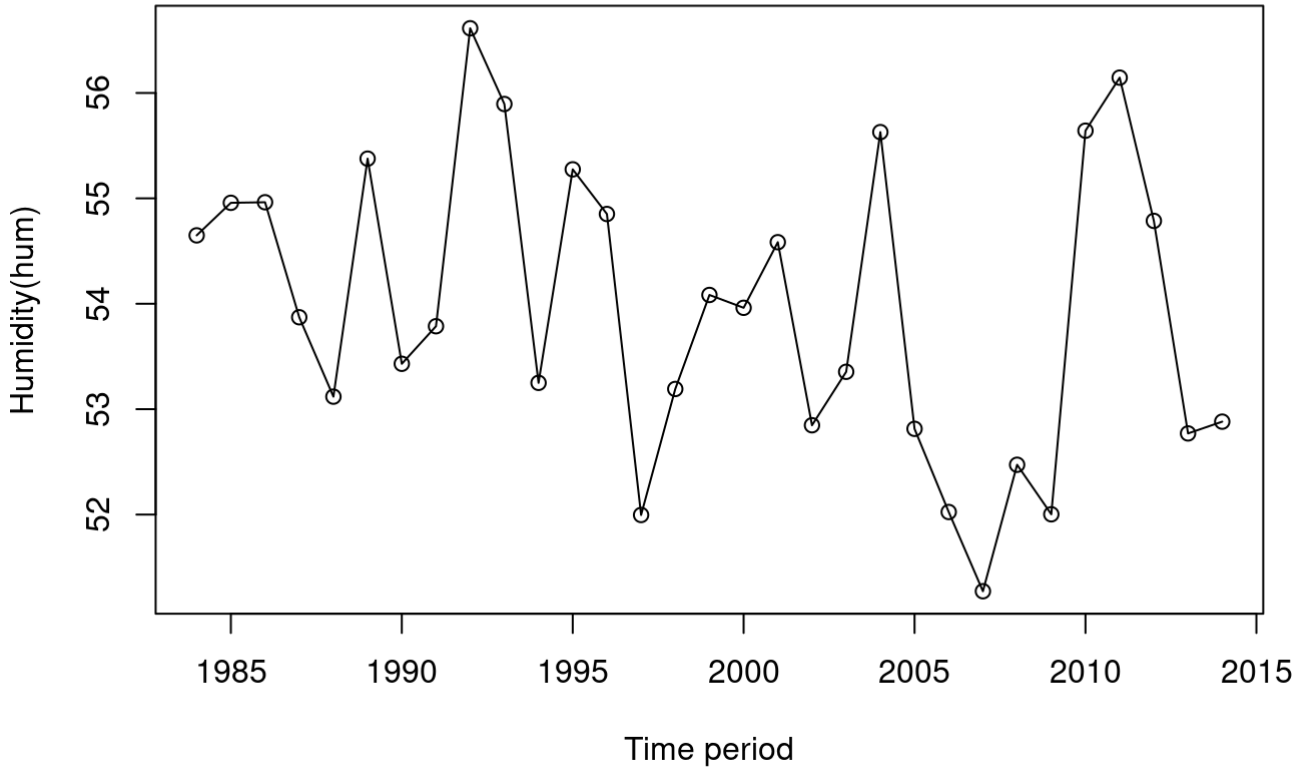
5.) Behavior : Several successive points and fluctuations can be observed along the mean level in the series therefore, a high order AR and MA behavior is seen.

Time-series plot of Humidity(hum):

A time-series plot of the yearly values of humidity(hum) was plotted using plot function.

```
plot(hum, ylab='Humidity(hum)', xlab='Time period', type='o', col='black', main = 'Figure 49: Time-series plot of change in yearly values of humidity')
```

Figure 49: Time-series plot of change in yearly values of humidity



Following observations were seen from the time series plot of the hum series-

1.) Trend : There seems to be no presence of a particular trend in the hum series.

2.) Seasonality : No particular seasonality can be seen in the series.

3.) Changing variance : There is presence of change in variance in the hum series.

4.) Intervention : There is no particular intervention point in the series, however, the lowest humidity was observed around 2005-2007 and the highest humidity just after 1990.

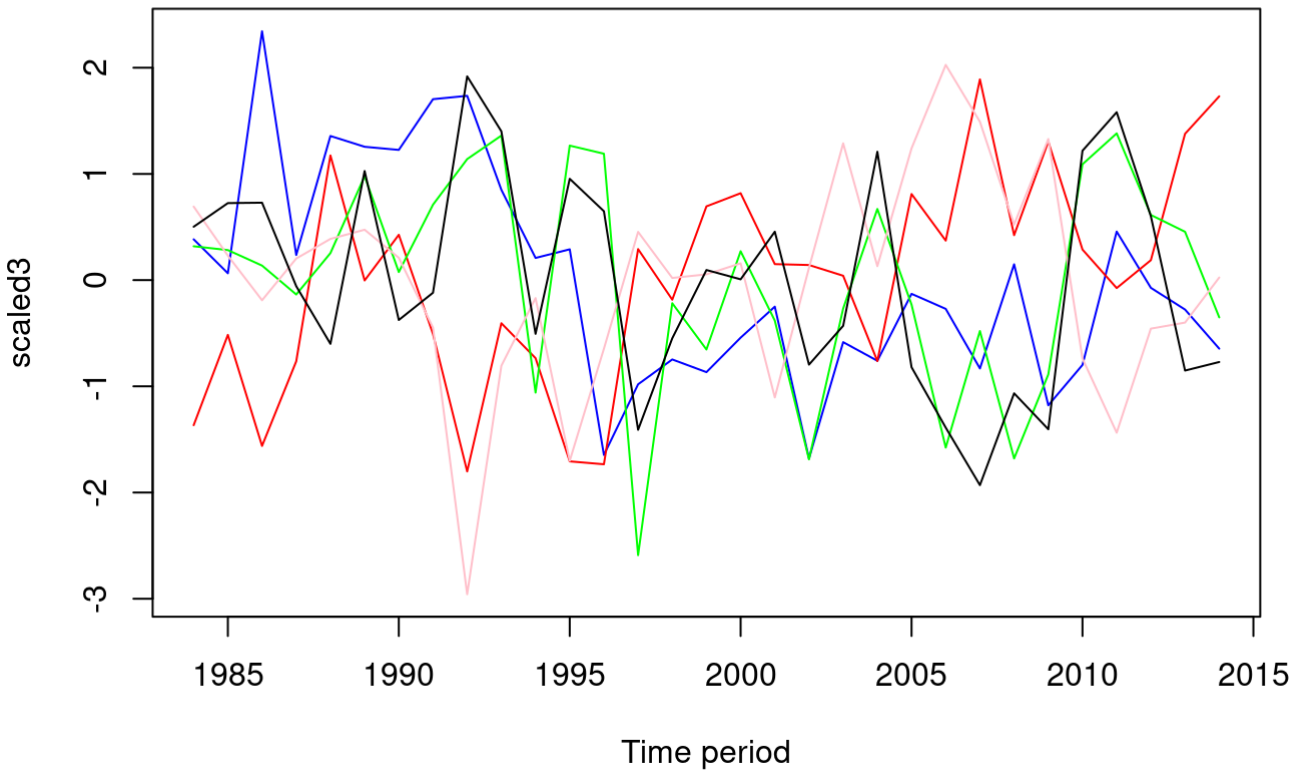
5.) Behavior : Several successive points and fluctuations can be observed along the mean level in the series therefore, a high order AR and MA behavior is seen.

Time-series plot of data set climate2:

After analyzing each of the series independently, we will plot them together to have a visual comparison in the nature of the each of the series. In order to plot them together in the same plot, scaling will be done for the `climate2` .

```
# Scaling 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","pink","black"), main = "Figure
 50: Time Series plot of  scaled climate2",xlab="Time period")
```

Figure 50: Time Series plot of scaled climate2



All the five series are likely to be correlated and are not seasonal in nature but some of them have a particular trend. In the next step, correlation between the five series will be calculated to mathematically see the above visual results-

Calculating correlation for `climate2`

The correlation is calculated for each variable in climate2 using a user-defined function flattenCorrMatrix where correlation between each variable along with its significance is calculated (at 5% level of significance) and displayed-

```
# Calculating correlation

res2<-rcorr(as.matrix(climate2))
flattenCorrMatrix(res2$r, res2$P)
```

```
##            row      column        cor            p
## 1          RBO Temperature -0.3452053 5.717467e-02
## 2          RBO    Rainfall  0.3932282 2.863685e-02
## 3  Temperature    Rainfall -0.3915072 2.940424e-02
## 4          RBO   Radiation -0.3173602 8.191765e-02
## 5  Temperature   Radiation  0.5193576 2.753070e-03
## 6     Rainfall   Radiation -0.5813161 6.046040e-04
## 7          RBO RelHumidity  0.3885540 3.076005e-02
## 8  Temperature RelHumidity -0.6621980 4.955000e-05
## 9     Rainfall RelHumidity  0.7911079 1.174543e-07
## 10   Radiation RelHumidity -0.7354087 2.443007e-06
```

From the above results we can say that-

1.Rainfall & relhumidity are highly positively correlated (0.791) with each other while radiation and relhumidity are highly negatively correlated.(-0.735).

2.Temperature & relhumidty (-0.662) and rainfall & radiation (-0.581) have moderate negative correlation with each other.

3.The major observation here is that all the correlation between respective variables is highly significant as p-value for all of them is less than 5% level of significance which verfies all the correlation computed are true and not spurious.

4.The dependent variable RBO is not majorly correlated with any of the four climate indicators. However, it has low level of significant corrleation with all the four indicators.

5.RBO is somewhat negatively correlated with temperature & radiation and it is somewhat postively correlated with rainfall & humidity.

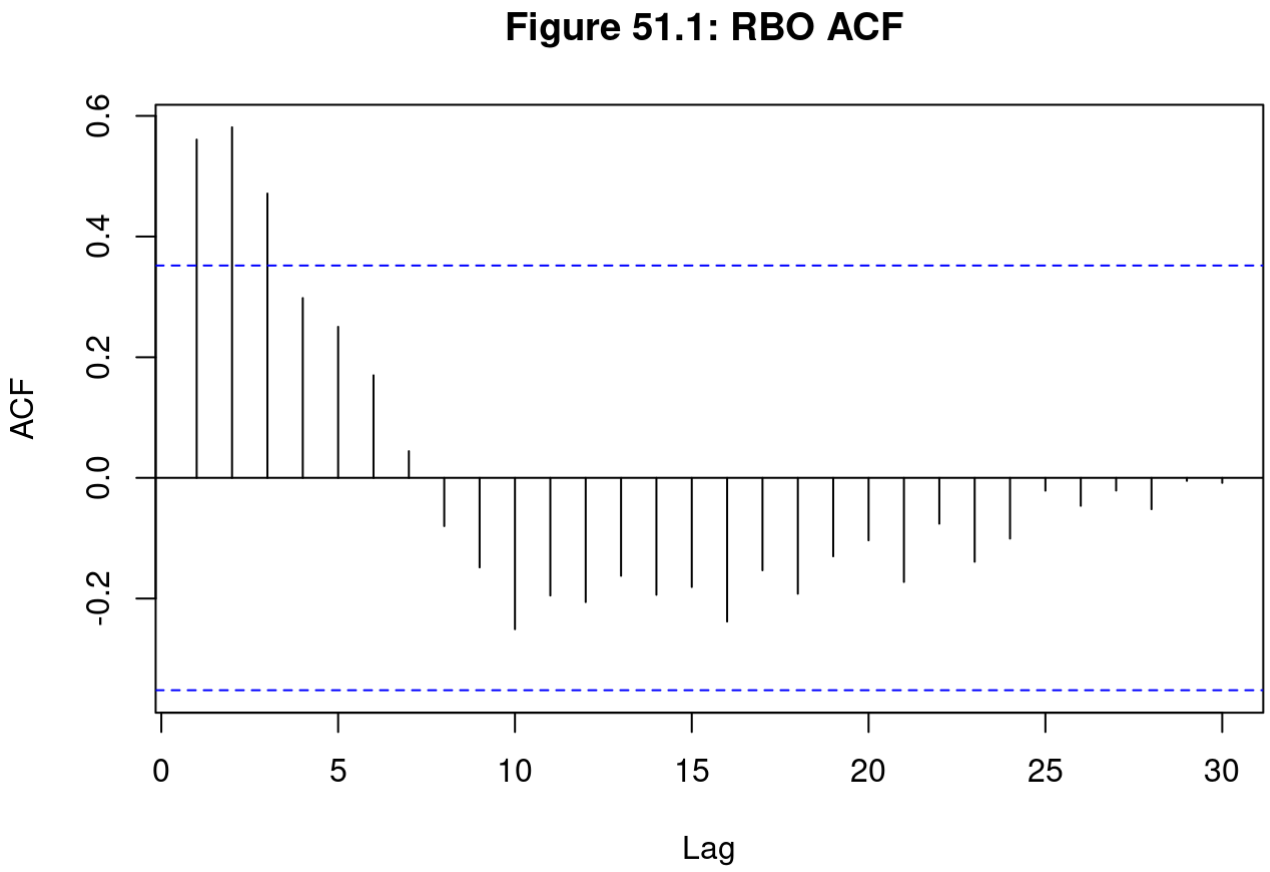
Checking non-stationarity

Let's now check whether there is existence of non-stationarity in the five series.The first check can be done through visualization by plotting ACF & PACF for each of the series. This can be further explored and verified by performing ADF tests for each series.

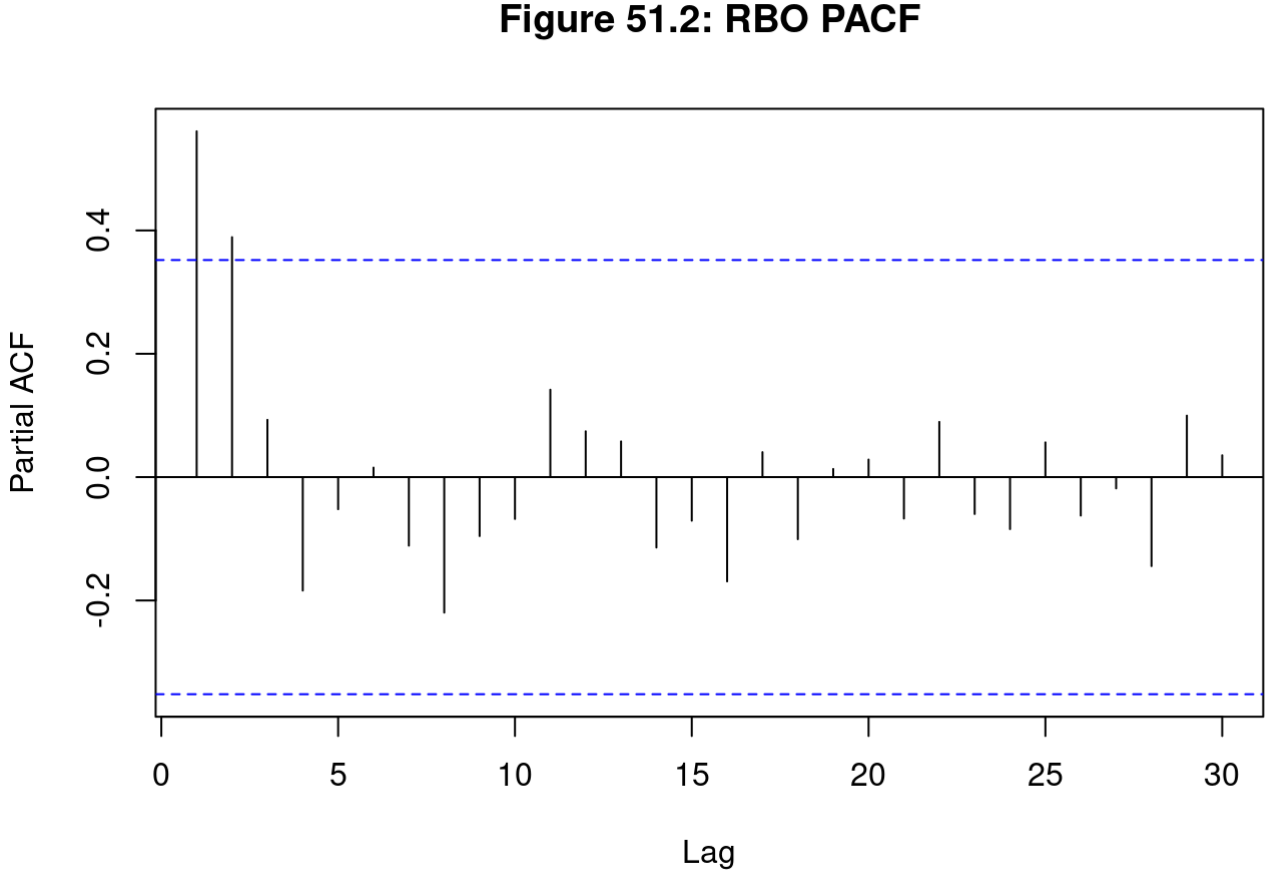
Plotting ACF and PACF:

Plotting ACF and PACF for RBO:

```
# Plotting ACF and PACF

acf(RBO, lag.max = 48, main = "Figure 51.1: RBO ACF ")
```



```
pacf(RBO, lag.max = 48, main = "Figure 51.2: RBO PACF")
```

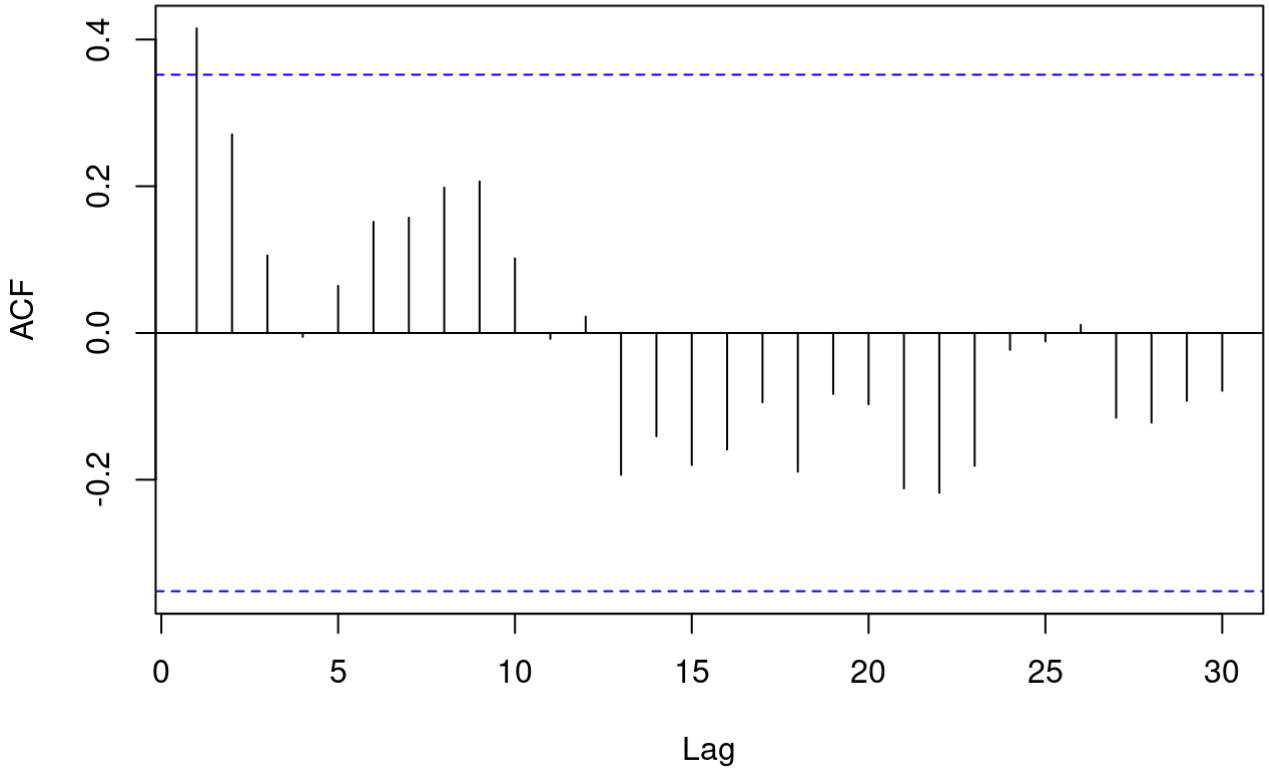


In the ACF plot, there is a slowly decaying pattern of positive & negative lags and we can also see a trend. In the PACF plot, there are two significant lags which suggests that the RBO series is non-stationary.

Plotting ACF and PACF for temperature(tem):

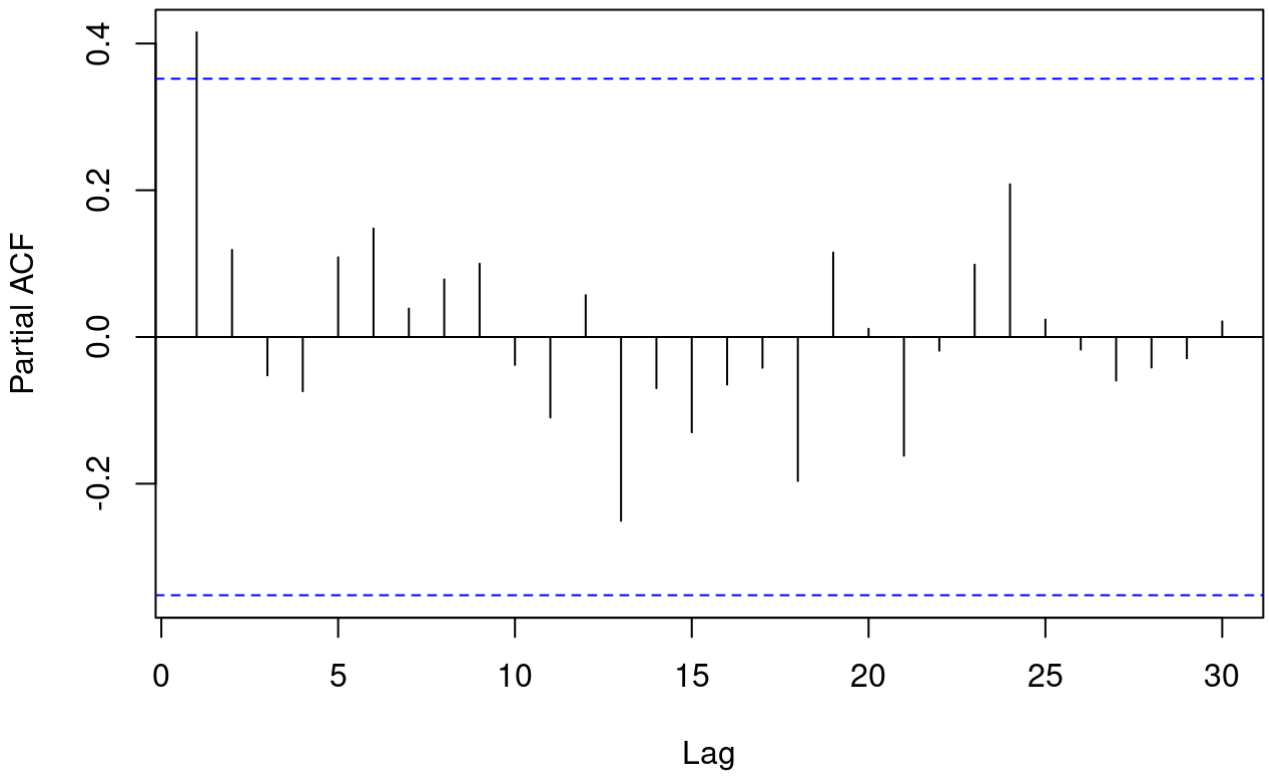
```
# Plotting ACF and PACF

acf(tem,main = "Figure 52.1: Tem ACF ",lag.max = 48)
```

Figure 52.1: Tem ACF




```
pacf(tem, main = "Figure 52.2: Tem PACF",lag.max = 48)
```

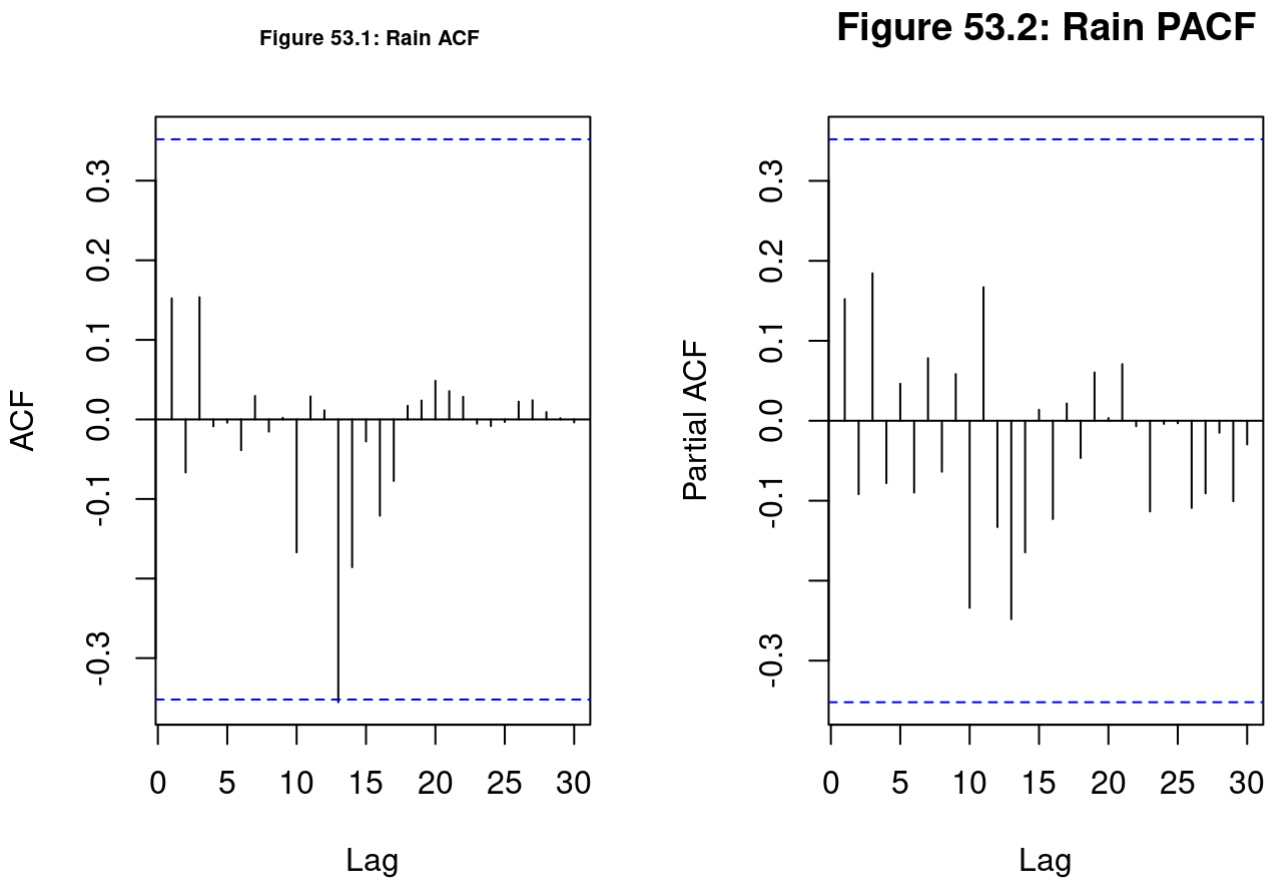
Figure 52.2: Tem PACF



In the ACF plot, there is a slowly decaying pattern of positive & negative lags. In the PACF plot, the first lag is significant which suggests that the tem series is non-stationary.

Plotting ACF and PACF for rainfall(rain):

```
# Plotting ACF and PACF

par(mfrow=c(1,2))
acf(rain, main = "Figure 53.1: Rain ACF ",lag.max = 48,cex.main=0.65)
pacf(rain, main = "Figure 53.2: Rain PACF",lag.max = 48,cex.main=0.05)
```



In the ACF plot, there is presence of no significant lag. In the PACF plot, no lag is significant which suggests that the rain series is stationary.

Plotting ACF and PACF for humidity(hum):

```
# Plotting ACF and PACF

par(mfrow=c(1,2))
acf(hum, main = "Figure 54.1: Hum ACF ",lag.max = 48,cex.main=0.65)
pacf(hum, main = "Figure 54.2: Hum PACF",lag.max = 48,cex.main=0.05)
```

Figure 54.1: Hum ACF

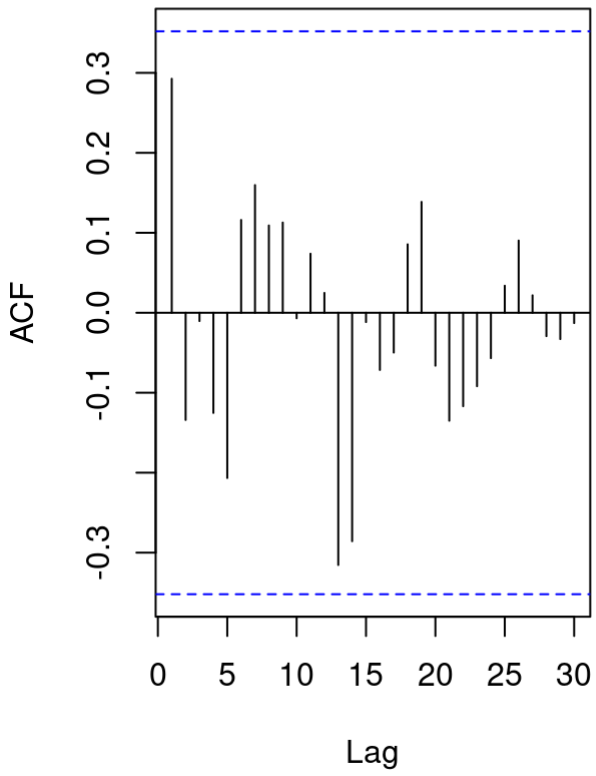
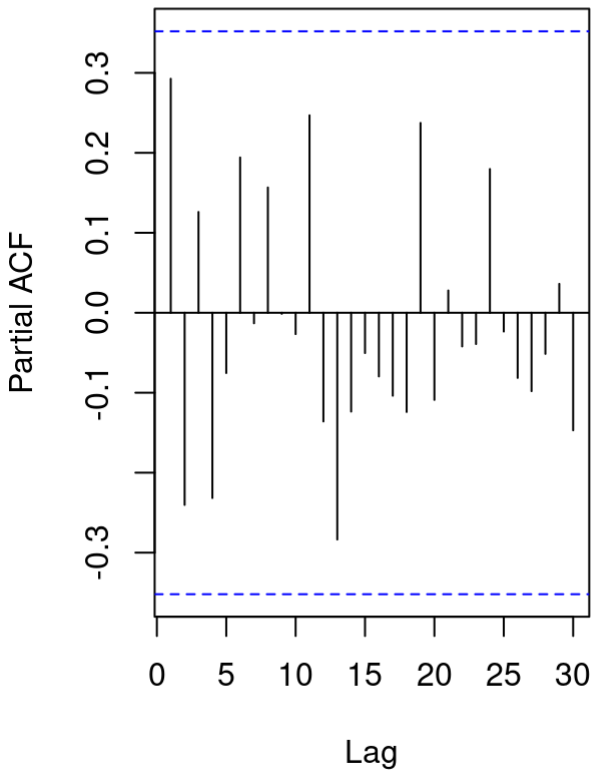


Figure 54.2: Hum PACF



In the ACF plot, there is presence of no significant lags and no particular pattern is seen. In the PACF plot, no lag is significant which suggests that the hum series is stationary.

Plotting ACF and PACF for radiation(rad):

```
# Plotting ACF and PACF

par(mfrow=c(1,2))
acf(rad, main = "Figure 55.1: Radiation ACF ",lag.max = 48,cex.main=0.65)
pacf(rad, main = "Figure 55.2: Radiation PACF",lag.max = 48,cex.main=0.05)
```

Figure 55.1: Radiation ACF

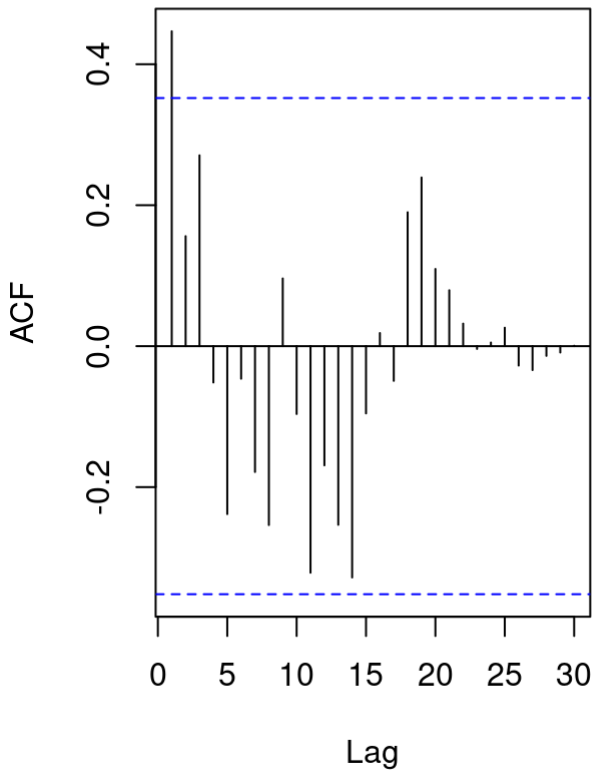
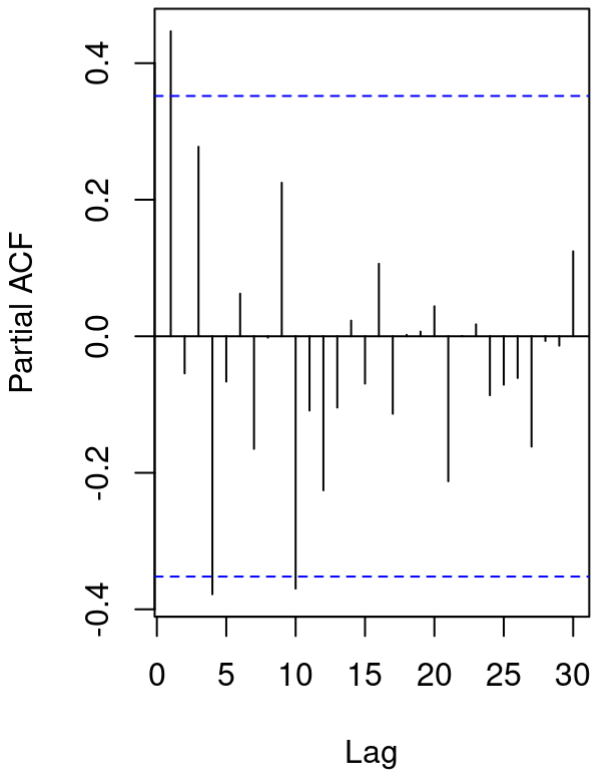


Figure 55.2: Radiation PACF



The ACF plot shows slowly decaying pattern and in the PACF plot, first lag is significant. So, the rad series is non-stationary.

Intepretation of ACF and PACF for all five series:

From the ACF and PACF plots it is seen that series RBO, tem and rad are non-stationary while rain & hum are stationary. However, ADF tests will be conducted next to verify this.

Dicker-Fuller Unit-Root test (ADF):

RBO series:

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
```

The ADF test of 2 lag order for RBO indicates series is non-stationary as p-value is greater than 5% level of significance.

Temperature(tem) series:

Performing ADF test on Temperature(tem) series-

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

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

The ADF test of 1 lag order supports the ACF & PACF plot indicating series is non-stationary at 5% significance level.

Rainfall(rain) series:

Performing ADF test on rainfall(rain) series-

```
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
```

The ADF test of 0 lag order for rain indicates series is stationary as p-value is less than 5% level of significance supporting the ACF & PACF plots plotted for it.

Humidity(hum) series:

Performing ADF test on humidity(hum) series-

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

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

The ADF test of 1 lag order for hum indicates series is stationary as p-value is less than 5% level of significance supporting the ACF & PACF plots plotted for it.

Radiation series:

Performing ADF test on radiation series-

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

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

The ADF test of 4 lag order supports the ACF & PACF plot indicating rad series is non-stationary at 5% significance level.

Interpretation: The ADF test computed for all the series completely matches the visual interpretation of ACF & PACF plots as there are no contradicting results.

Transformation for non-stationary series

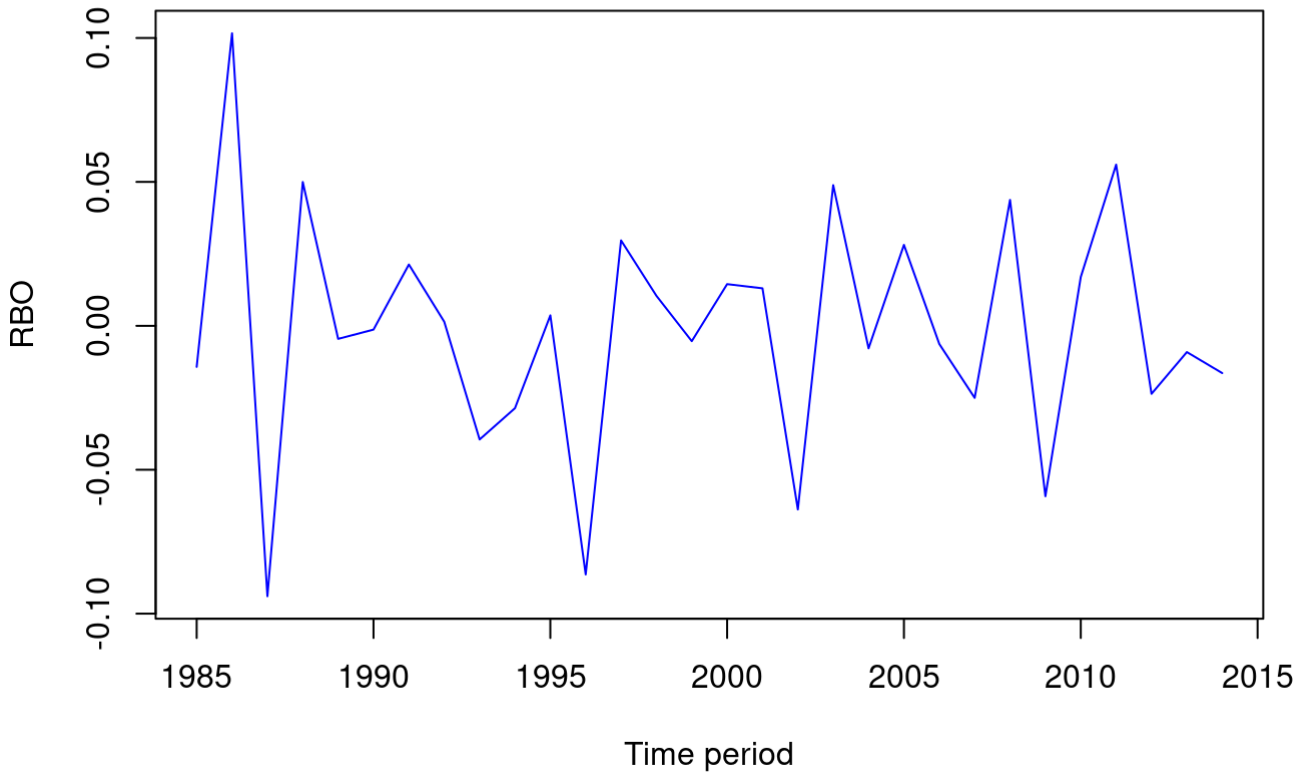
As verified from previous results, FFD, temperature & radiation are non-stationary series. Hence, they need to be converted to stationary in order to proceed with the analysis. Thus, first, second, third etc order of differencing will be applied on each of these series until they become stationary.

Transformation of RBO series:

Applying first order differencing on FFD series and performing ADF test to check stationarity status-

```
RBOdiff = diff(RBO)
plot(RBOdiff,ylab='RBO',xlab='Time period', col="blue", main = "Figure 56: Time series plot o
f first differenced RBO")
```

Figure 56: Time series plot of first differenced RBO



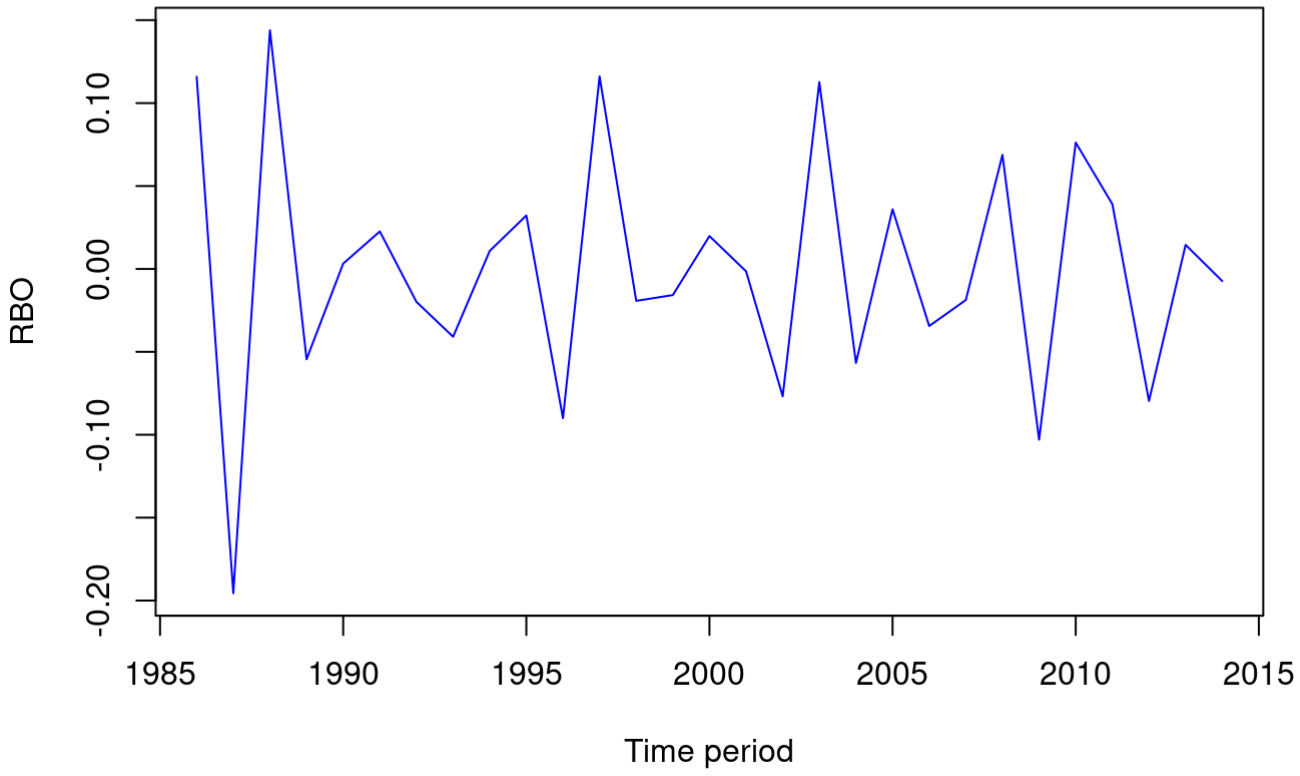
```
adf.test(RBO)
```

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

The series has not become stationary (as p-value > 0.05), however it has changed visually from the original series. Let's apply second order of differencing to make it stationary-

```
RBOdiff2 = diff(RBOdiff)
plot(RBOdiff2,ylab='RBO',xlab='Time period', col="blue", main = "Figure 57: Time series plot
 of second differenced RBO")
```

Figure 57: Time series plot of second differenced RBO

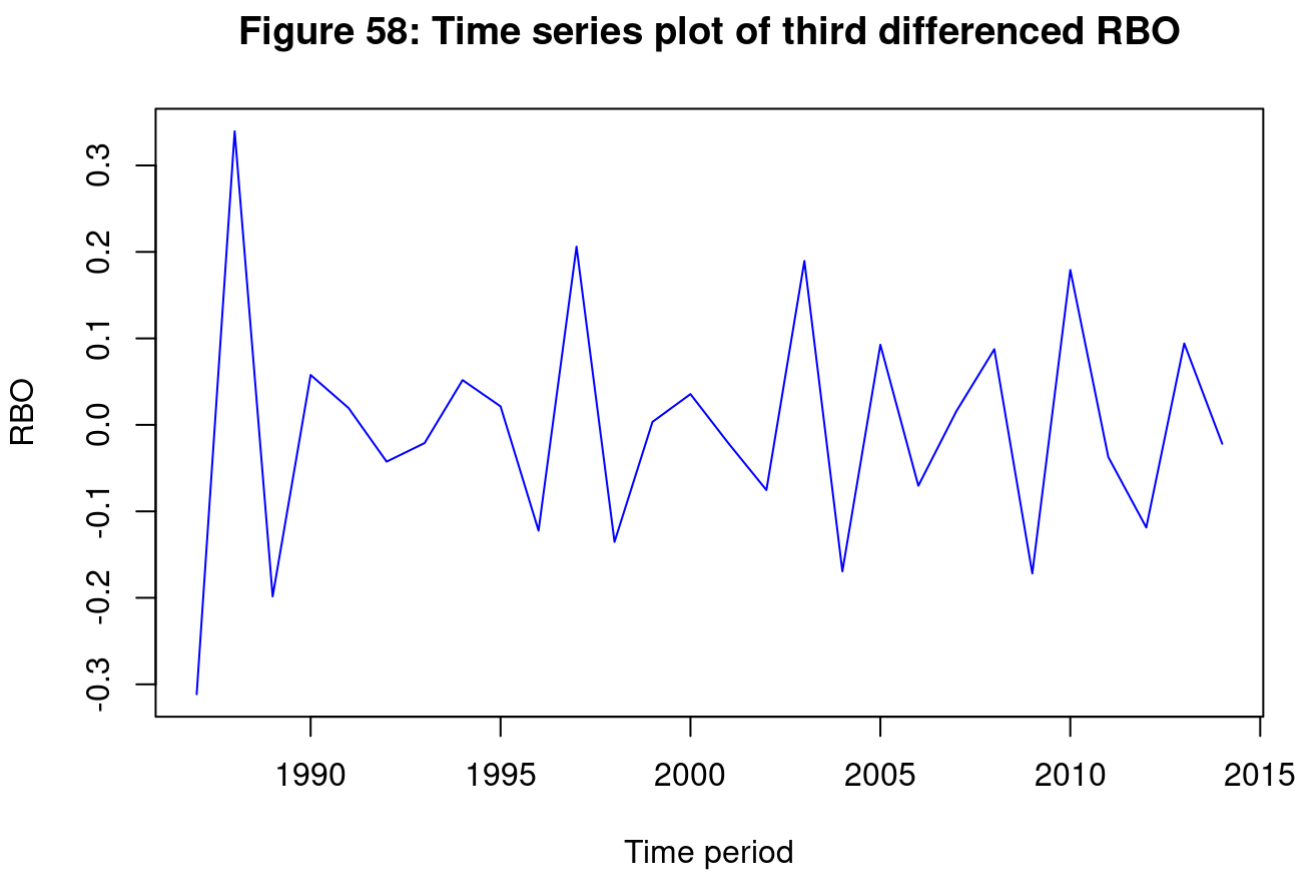



```
adf.test(RBOdiff2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  RBOdiff2
## Dickey-Fuller = -3.1875, Lag order = 3, p-value = 0.1174
## alternative hypothesis: stationary
```

The series has not become stationary (as p-value > 0.05), however it has changed visually from the original series. Let's apply third order of differencing to make it stationary-

```
RBOdiff3 = diff(RBOdiff2)
plot(RBOdiff3,ylab='RBO',xlab='Time period', col="blue", main = "Figure 58: Time series plot
 of third differenced RBO")
```




```
adf.test(RBOdiff3)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  RBOdiff3
## Dickey-Fuller = -4.0249, Lag order = 3, p-value = 0.02192
## alternative hypothesis: stationary
```

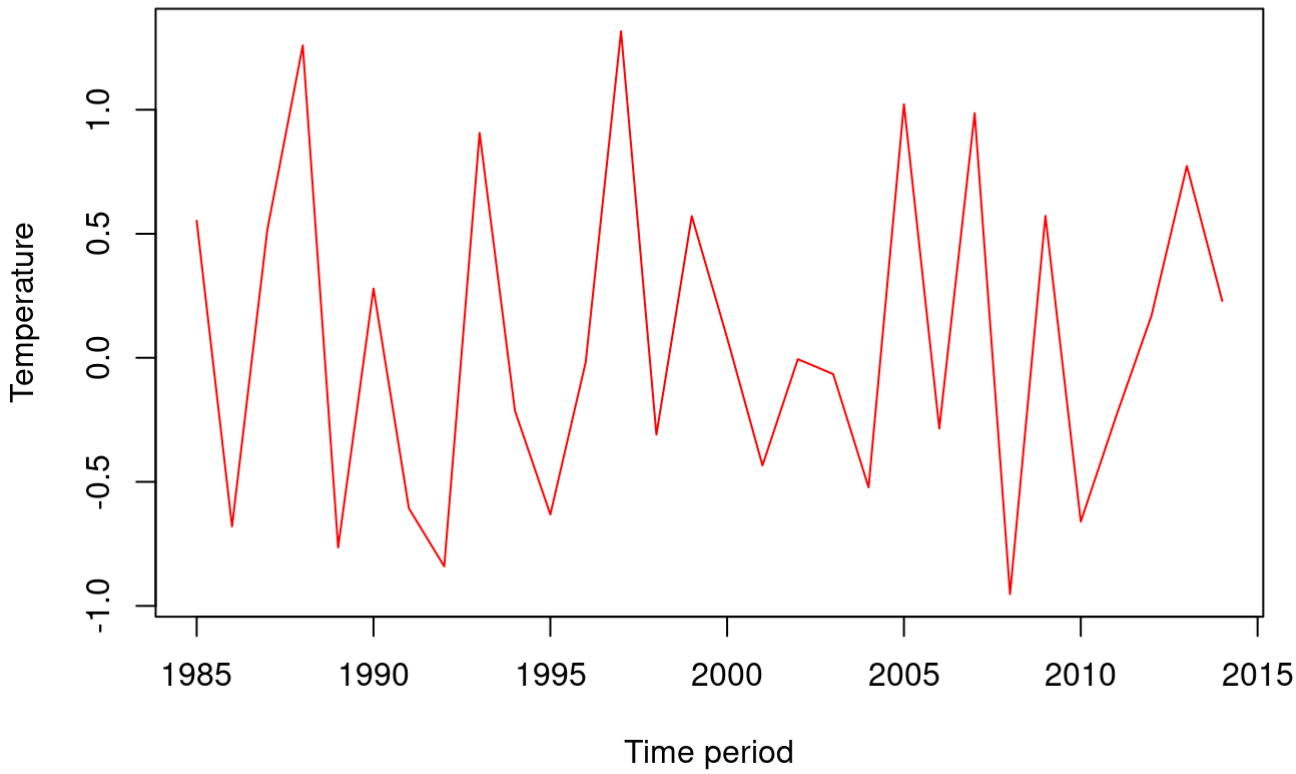
The series has finally become completely stationary as p-value is less than 5% significance level after third order differencing.

Transformation of temperature(tem) series:

Applying first order differencing on tem series and performing ADF test to check stationarity status-

```
temdiff = diff(tem)
plot(temdiff,ylab='Temperature',xlab='Time period', col="red", main = "Figure 59: Time series
plot of first differenced tem series")
```

Figure 59: Time series plot of first differenced tem series



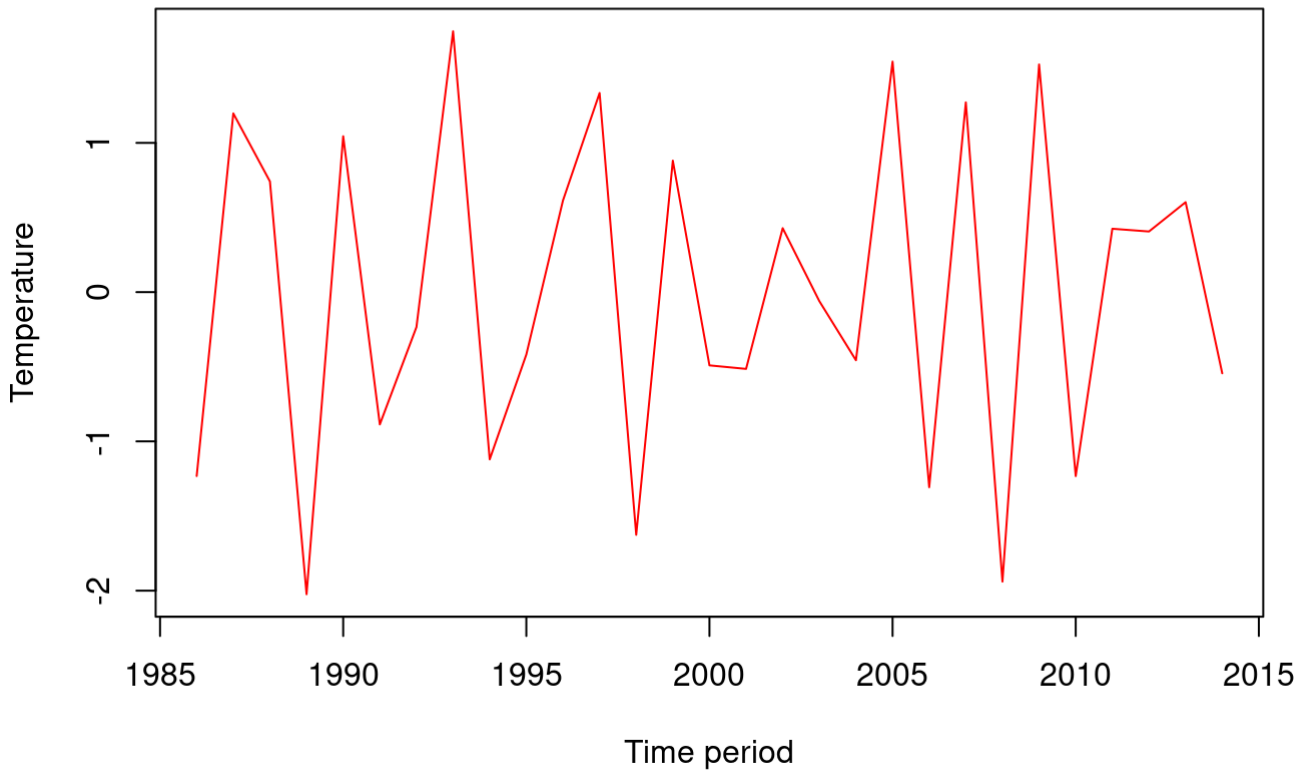

```
adf.test(temdiff)
```

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

The series has not become stationary (as p-value > 0.05), however it has changed visually from the original series. Let's apply second order of differencing to make it stationary-

```
temdiff2 = diff(temdiff)
plot(temdiff2,ylab='Temperature',xlab='Time period', col="red", main = "Figure 60: Time serie
s plot of second differenced tem series")
```

Figure 60: Time series plot of second differenced tem series



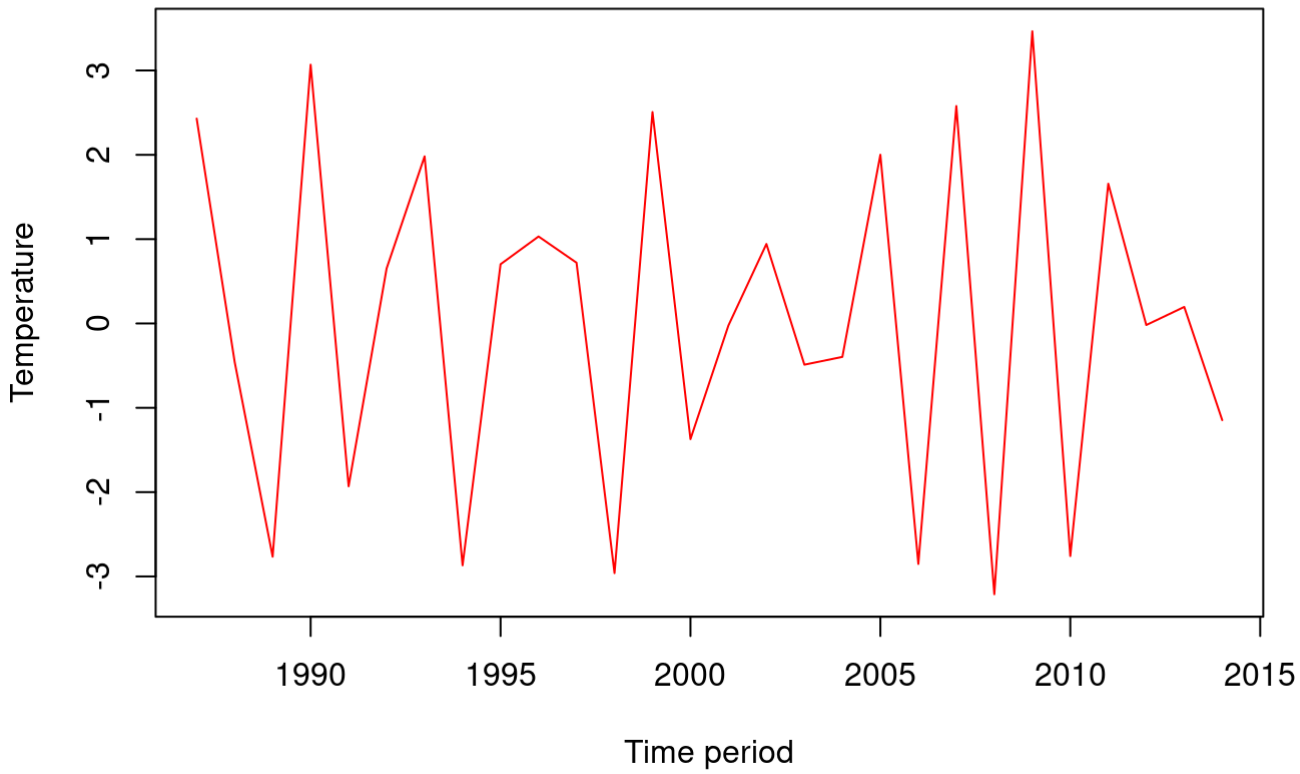

```
adf.test(temdiff2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  temdiff2
## Dickey-Fuller = -3.2642, Lag order = 3, p-value = 0.09558
## alternative hypothesis: stationary
```

The series has still not become stationary (as p-value > 0.05), however it has changed visually from the original series. Let's apply third order of differencing to make it stationary-

```
temdiff3 = diff(temdiff2)
plot(temdiff3,ylab='Temperature',xlab='Time period', col="red", main = "Figure 61: Time serie
s plot of third differenced tem series")
```

Figure 61: Time series plot of third differenced tem series




```
adf.test(temdiff3)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  temdiff3
## Dickey-Fuller = -4.0851, Lag order = 3, p-value = 0.01979
## alternative hypothesis: stationary
```

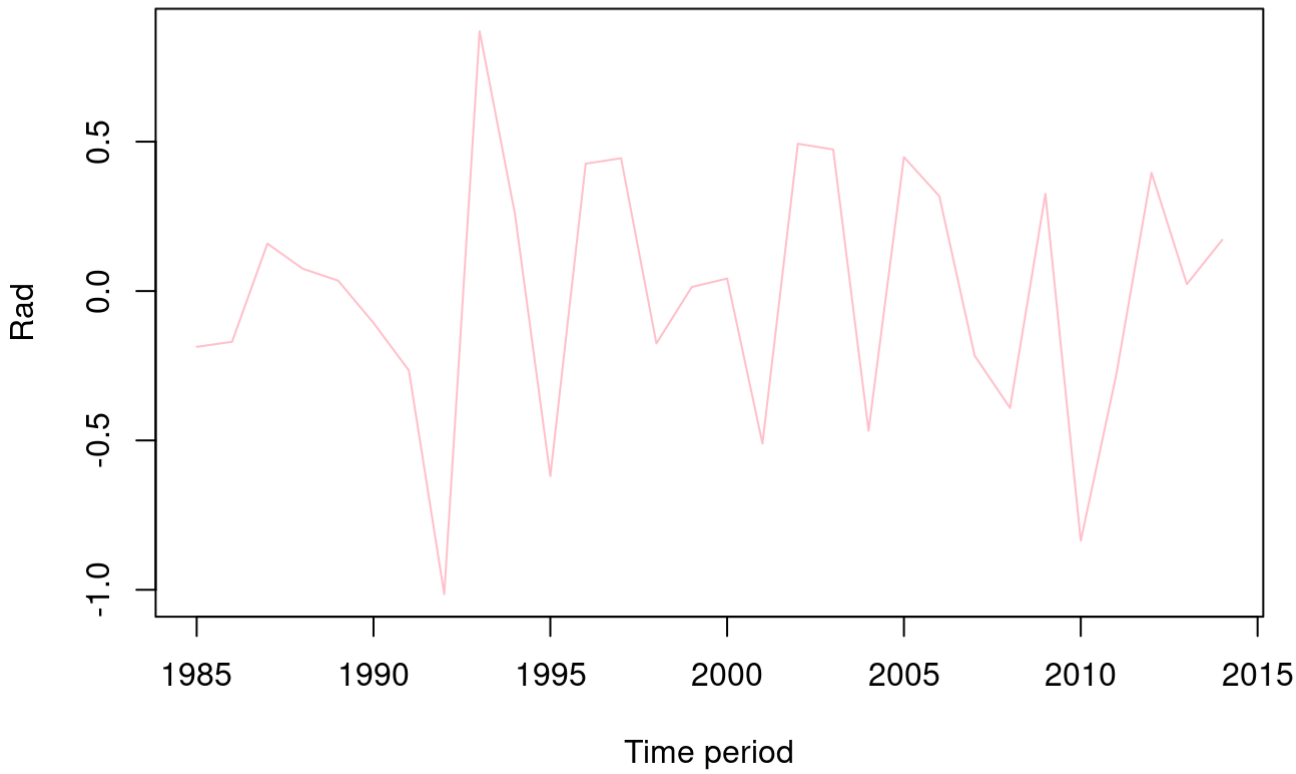
The tem series has finally become completely stationary as p-value is less than 5% significance level after third order differencing.

Transformation of radiation(rad) series:

Applying first order differencing on rad series and performing ADF test to check stationarity-

```
raddiff = diff(rad)
plot(raddiff,ylab='Rad',xlab='Time period', col="pink", main = "Figure 62: Time series plot o
f first differenced rad series")
```

Figure 62: Time series plot of first differenced rad series



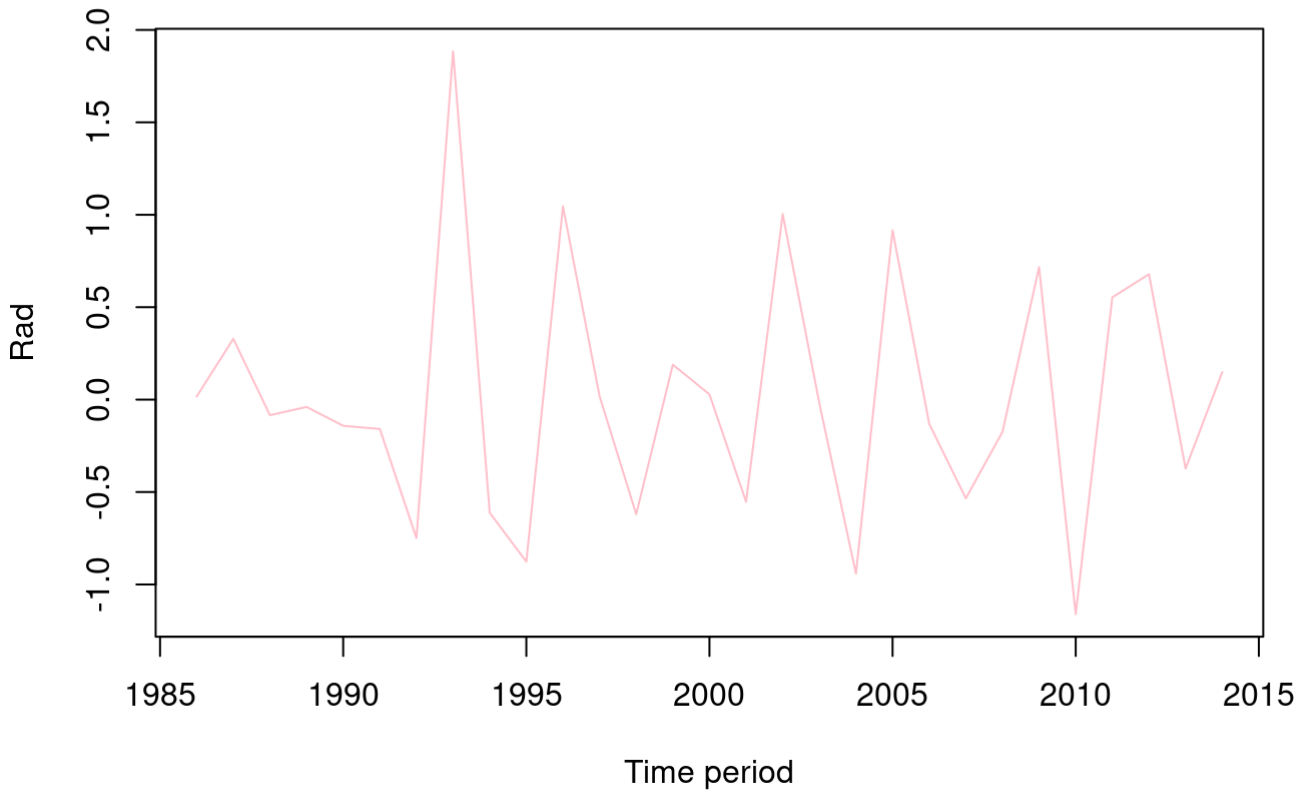

```
adf.test(raddiff)
```

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

The rad series has not become stationary (as p-value > 0.05), however it has changed gradually from the original series. Let's apply second order of differencing to make it stationary-

```
raddiff2 = diff(raddiff)
plot(raddiff2,ylab='Rad',xlab='Time period', col="pink", main = "Figure 63: Time series plot
 of second differenced rad series")
```

Figure 63: Time series plot of second differenced rad series



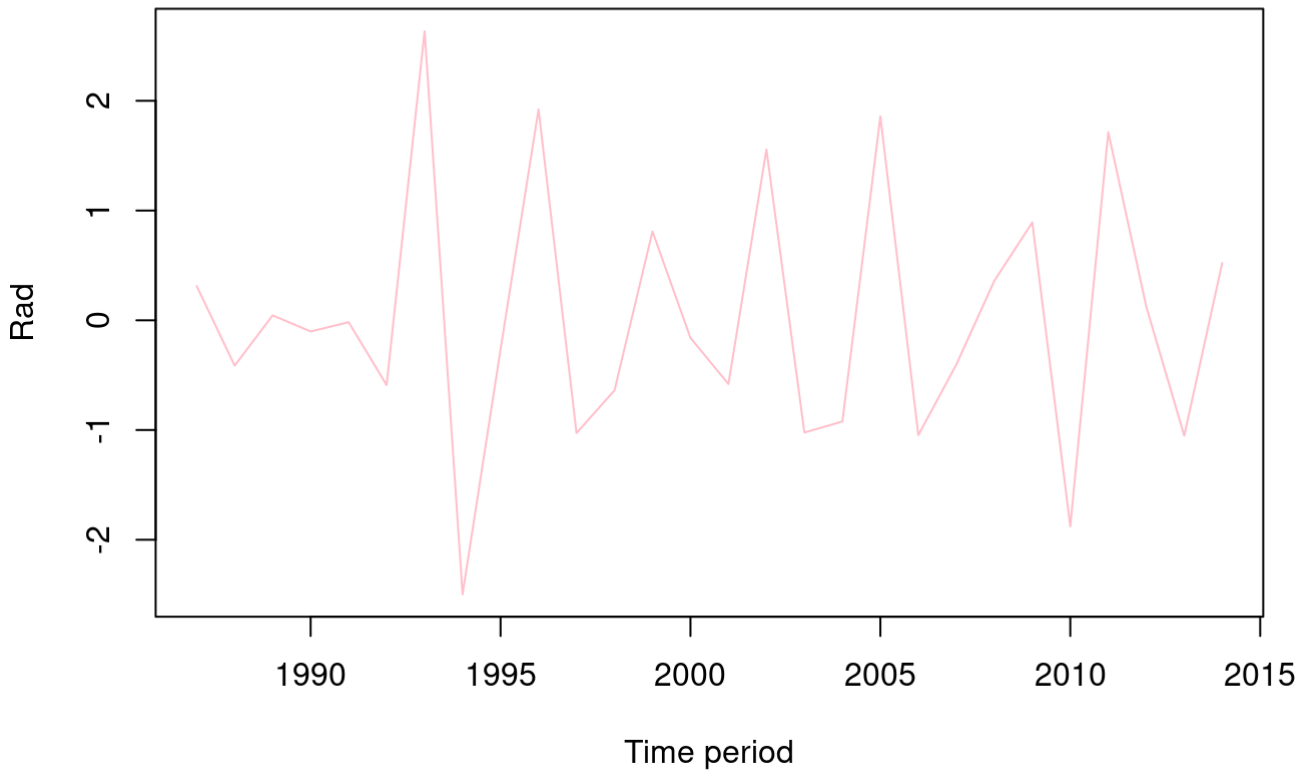

```
adf.test(raddiff2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  raddiff2
## Dickey-Fuller = -3.0559, Lag order = 3, p-value = 0.1678
## alternative hypothesis: stationary
```

The rad series has still not become stationary (as p-value > 0.05), however it has changed gradually from the original series. Let's apply third order of differencing to make it stationary-

```
raddiff3 = diff(raddiff2)
plot(raddiff3,ylab='Rad',xlab='Time period', col="pink", main = "Figure 64: Time series plot
 of third differenced rad series")
```

Figure 64: Time series plot of third differenced rad series




```
adf.test(raddiff3)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  raddiff3
## Dickey-Fuller = -4.1613, Lag order = 3, p-value = 0.01709
## alternative hypothesis: stationary
```

The tem series has finally become completely stationary as p-value is less than 5% significance level after third order differencing.

Intepretation: All the three series(RBO,tem and rad) have transformed from non-stationary to stationary after applying third order differencing.

Decomposition

As we can't decompose an yearly time series of frequency less than 2, we are unable to utilize the known X12/STL decomposition. We will proceed to fitting different models for the purpose of predicting RBO.

Modelling using Distributed Lag Model(DLM)

As all the five series have become stationary, let's now fit different DLM models with RBO as dependent variable and other climate indicators as predictors one by one (univariate models).

Finite DLM

For finding the best model using finite DLM, we will consider all the four series (except RBO) as predictor one by one and fit finite DLM based on appropriate lag length.

Computing lag length for the model with tem as predictor and RBO as dependent variable by using finiteDLMauto function that will perform a comparison based on AIC, BIC and MASE values for lags ranging from 1 to 10. We will choose model having lowest of AIC,BIC and MASE values-

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

colnames(climate2) <- c("RBO","tem","rain","rad","hum")

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

finiteDLMauto( x=as.vector(tem), y= as.vector(RBO), 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
## 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
```

As per the above output, lag-length should be 1 based on the AIC, BIC and MASE values. Using the finiteDLMauto function, all the other predictors were replaced to check the lag length. For all the predictor variables in the dataset, it was found that optimum lag length is 1. **Thus, for all the finite DLMs, lag length will be taken as 1.**

Tem vs RBO:

Fitting finite DLM model with tem as predictor-

```
model.tem = dlm(x=as.vector(tem), y=as.vector(RBO), q=1)
```

Rain vs RBO:

Fitting finite DLM model with rain as predictor-

```
model.rain = dlm(x=as.vector(rain), y=as.vector(RBO), q=1)
```

Humidity vs FFD:

Fitting finite DLM model with humidity as predictor-

```
model.hum = dlm(x=as.vector(hum), y=as.vector(RBO), q=1)
```

Rad vs RBO:

Fitting finite DLM model with rad as predictor-

```
model.rad = dlm(x=as.vector(rad), y=as.vector(RBO), q=1)
```

Tem vs RBO (without intercept):

Fitting finite DLM model with tem as predictor without the intercept-

```
model.nointercepttem = dlm(formula = RBO ~ tem - 1, data=data.frame(climate2), q=1)
```

Rain vs RBO (without intercept):

Fitting finite DLM model with rain as predictor without the intercept-

```
model.nointerceptrain = dlm(formula = RBO ~ rain - 1, data=data.frame(climate2), q=1)
```

Hum vs RBO (without intercept):

Fitting finite DLM model with hum as predictor without the intercept-

```
model.nointercepthum = dlm(formula = RBO ~ hum - 1, data=data.frame(climate2), q=1)
```

Rad vs RBO (without intercept):

Fitting finite DLM model with rad as predictor without the intercept-

```
model.nointerceptrad = dlm(formula = RBO ~ rad - 1, data=data.frame(climate2), q=1)
```

Sorting based on AIC,BIC and MASE:

```
# Sorting based on AIC

sort.score(AIC(model.tem$model,model.rain$model,model.hum$model,model.rad$model,model.nointer
cepttem$model,model.nointerceptrain$model,model.nointercepthum$model,model.nointerceptrad$mod
el), score = "aic")
```

```
##                              df        AIC
## model.rain$model              4 -100.89798
## model.nointercepthum$model    3 -100.84952
## model.hum$model               4  -98.90083
## model.rad$model               4  -97.71113
## model.tem$model               4  -97.52442
## model.nointerceptrad$model    3  -89.33439
## model.nointercepttem$model    3  -86.53783
## model.nointerceptrain$model   3  -62.07890
```

```
# Sorting based on BIC

sort.score(BIC(model.tem$model,model.rain$model,model.hum$model,model.rad$model,model.nointer
cepttem$model,model.nointerceptrain$model,model.nointercepthum$model,model.nointerceptrad$mod
el), score = "bic")
```

```
##                              df       BIC
## model.nointercepthum$model    3 -96.64593
## model.rain$model              4 -95.29319
## model.hum$model               4 -93.29604
## model.rad$model               4 -92.10634
## model.tem$model               4 -91.91963
## model.nointerceptrad$model    3 -85.13079
## model.nointercepttem$model    3 -82.33424
## model.nointerceptrain$model   3 -57.87530
```

```
# Sorting based on MASE

sort.score(MASE(model.tem$model,model.rain$model,model.hum$model,model.rad$model,model.nointe
rcepttem$model,model.nointerceptrain$model,model.nointercepthum$model,model.nointerceptrad$mo
del), score = "mase")
```

```
##                               n      MASE
## model.rain$model             30 0.9417954
## model.hum$model              30 0.9858746
## model.nointercepthum$model   30 0.9918515
## model.tem$model              30 1.0056219
## model.rad$model              30 1.0630293
## model.nointerceptrad$model   30 1.3296554
## model.nointercepttem$model   30 1.4121567
## model.nointerceptrain$model  30 1.9110310
```

Based on AIC and MASE, the model.rain is the best model and based on BIC, it is the second best model. While model.nointercept hum is the best model based on BIC. Therefore, analyzing both models further-

Analysing model.rain:

Using summary and checkresiduals function, analysing the model.rain finite DLM-

```
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 above summary, we can interpret the following things about model.rain-

1. The model model.rain is a poor fitted model. It contains insignificant lags except the intercept.
2. The value of adjusted R2 is 21.56%. However, since there is only one independent variable, we will look at the R2 value which is 15.75%. This means that model.rain is able to explain only 15.75% variation in the dependent variable (RBO).
3. The residuals have maximum value of 0.099699 and minimum value of -0.105903 with residual standard error (RSE) as 0.04153.
4. The AIC & BIC values of the model are -100.898 & -95.29319 respectively.
5. As per the F-test of the overall significance of model, the model.rain is a significant model at 5% level of significance despite having very poor R2 value

Calculating VIF for model.rain finite DLM:

Calculating VIF for the model.rain DLM to check for presence of multicollinearity in the model-

```
vif(model.rain$model)
```

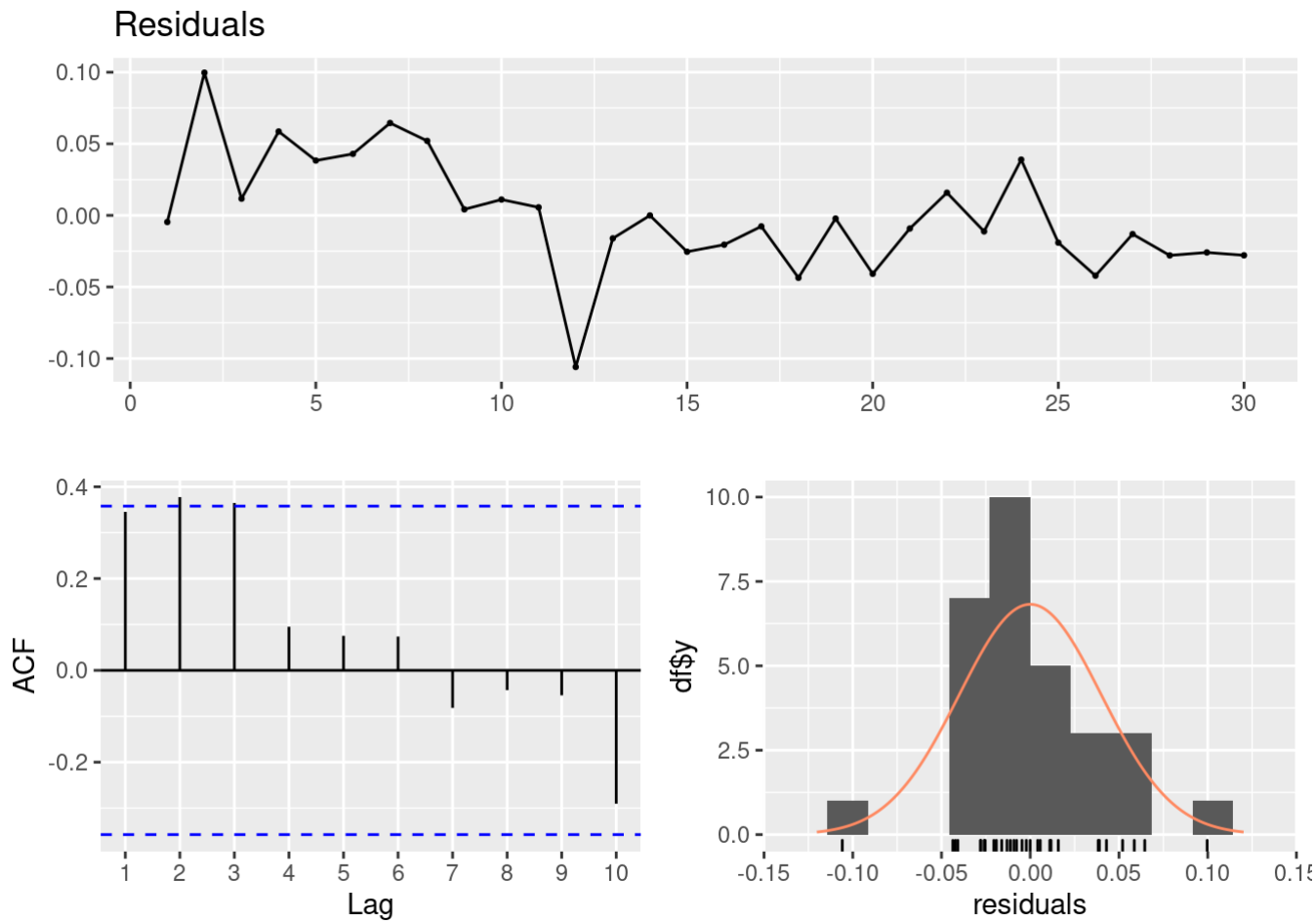
```
##      x.t      x.1
## 1.023917 1.023917
```

As seen above, the VIF <10 for all the lags in the model. Hence, there is no issue of multicollinearity in the model.rain model.

Residual Analysis for model.rain DLM:

The residual analysis will be performed on the model.rain using checkresiduals function. In addition to generating residual plots, the Breusch-Godfrey test will be performed. In this test, H0 indicates there is no serial correlation in the residuals while Ha indicates there is serial correlation in the residuals.

```
#Residual analysis of model.rain model


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 Breusch-Godfrey test conducted, the p-value is greater than 5% level of significance which indicates that no serial correlation exists among the residuals. The results obtained from the Breusch-Godfrey test are supported by the residuals plots generated as it is seen that residuals seem to follow a decreasing trend and are not distributed randomly. From the ACF plot, it is clearly seen that no lags are significant supporting the BG test. The histogram shows that residuals are not following normal distribution.

Analyzing model.nointercepthum:

Using summary and checkresiduals function, analysing the model.nointercepthum finite DLM-

```
summary(model.nointercepthum)
```

```
##
## Call:
## lm(formula = as.formula(model.formula), data = design)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.086693 -0.022890 -0.005036  0.011807  0.091105
##
## Coefficients:
##       Estimate Std. Error t value Pr(>|t|)
## hum.t 0.010456   0.004723   2.214   0.0351 *
## hum.1 0.003213   0.004717   0.681   0.5014
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0422 on 28 degrees of freedom
## Multiple R-squared:  0.997,  Adjusted R-squared:  0.9967
## F-statistic:  4581 on 2 and 28 DF,  p-value: < 2.2e-16
##
## AIC and BIC values for the model:
##         AIC       BIC
## 1 -100.8495 -96.64593
```

From the above summary, we can interpret the following things about model.nointercepthum-

1. The model model.nointercepthum is a perfectly fitted model. It contains one significant and one insignificant lag.
2. The value of R2 is very good (99.7%)..This means that model.nointercepthum is able to explain only (99.7%) variation in the dependent variable (RBO).
3. The residuals have maximum value of 0.091105 and minimum value of -0.086693 with residual standard error (RSE) as 0.0422.
4. The AIC & BIC values of the model are -100.8495 -96.64593 respectively.
5. As per the F-test of the overall significance of model, the model.nointercepthum is a significant model at 5% level of significance.

Calculating VIF for model.nointercepthum finite DLM:

Calculating VIF for the model.nointercepthum DLM to check for presence of multicollinearity in the model-

```
vif(model.nointercepthum$model)
```

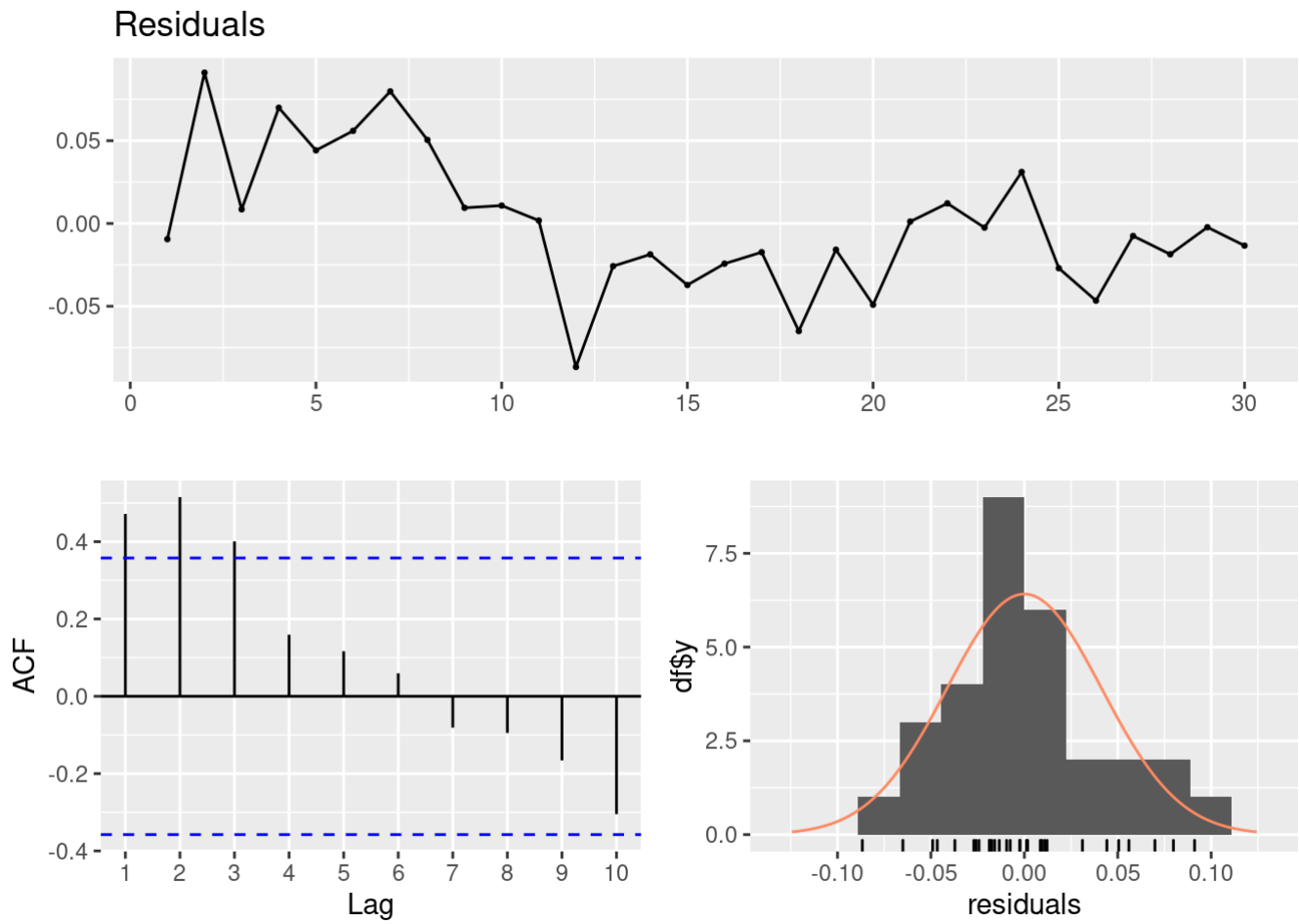
```
##    hum.t    hum.1
## 1093.203 1093.203
```

As seen above, the VIF > 10 for all the lags in the model. Hence, there is issue of multicollinearity in the model.nointercepthum model.

Residual Analysis for model.nointercepthum DLM:

The residual analysis will be performed on the model.nointercepthum using checkresiduals function.

```
#Residual analysis of model.nointercepthum model


checkresiduals(model.nointercepthum$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 6
##
## data:  Residuals
## LM test = 12.005, df = 6, p-value = 0.06186
```

From the Breusch-Godfrey test conducted, the p-value is greater than 5% level of significance which indicates that no serial correlation exists among the residuals. The results obtained from the Breusch-Godfrey test are supported by the residuals plots generated as it is seen that residuals seem to follow a decreasing trend and are not distributed randomly. From the ACF plot, it is clearly seen that two lags are significant which contradict the BG test. The histogram shows that residuals are not following normal distribution.

Intepretation: Both the models model.rain & model.nointercepthum are significant models as per the F-test and have good accuracy scores in terms of AIC, BIC and MASE. However, model.nointercepthum has a very good R2 value in comparison to model. But this value can be influenced by the multicollinearity discovered through VIF test. Moreover, it is not a good idea to fit a model without an intercept in general regression analysis. Hence, based on MASE and F-test, model.rain will be considered further for prediction.

Polynomial DLM

The finite DLM model was a poor fit , we will try to fit a polynomial model.Before, we fit the polynomial model, using finiteDLMauto, we will find the best lag length based on fitted models AIC,BIC and MASE values-

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

finiteDLMauto(x = as.vector(tem), 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
```

As per the above output, lag-length should be 1 but we cannot fit a poynomial model with lag-length 1. Thus, we will select the second best lag-length which is 2. Based on the AIC, BIC and MASE values. Using the finiteDLMauto function, all the other predictors were replaced to check the lag length. For all the predictor variables in the dataset, it was found that optimum lag length is 1. **Thus, for all the polynomial DLMs, lag length will be taken as 2 & order as 2.**

Fitting Polynomial DLM:

As models without intercept are not considered a good fit for prediction, we will only fit models with intercept using other DLM methods. Let's fit the possible combination of possible models using polynomial DLM-

```
# Fitting all combination of polynomial models

model.tempoly = polyDlm(x = as.vector(tem), y = as.vector(RBO), q = 2, k = 2)

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

model.humpoly = polyDlm(x = as.vector(hum), y = as.vector(RBO), q = 2, k = 2)

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

In the above code, we fitted 4 different polynomial models. Let's generate the accuracy score for each of them-

Comparison based on AIC,BIC and MASE:

```
# Sorting based on AIC, BIC and MASE

sort.score(AIC(model.tempoly$model,model.rainpoly$model,model.humpoly$model,model.radpoly$mod
el), score = "aic")
```

```
##                       df       AIC
## model.rainpoly$model  5 -96.70956
## model.humpoly$model   5 -94.63327
## model.tempoly$model   5 -91.57332
## model.radpoly$model   5 -91.50708
```

```
sort.score(BIC(model.tempoly$model,model.rainpoly$model,model.humpoly$model,model.radpoly$mod
el), score = "bic")
```

```
##                       df       BIC
## model.rainpoly$model  5 -89.87308
## model.humpoly$model   5 -87.79679
## model.tempoly$model   5 -84.73684
## model.radpoly$model   5 -84.67061
```

```
sort.score(MASE(model.tempoly$model,model.rainpoly$model,model.humpoly$model,model.radpoly$mo
del), score = "mase")
```

```
##                        n      MASE
## model.rainpoly$model  29 0.9993747
## model.humpoly$model   29 1.1026478
## model.tempoly$model   29 1.1474219
## model.radpoly$model   29 1.1747093
```

Based on AIC, BIC and MASE, model.rainpoly is the best polynomial DLM. Let's analyze this model further-

Analyzing model.rainpoly:

```
summary(model.rainpoly$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 above summary, we can generate following insights about the model.rainpoly-

1.The model model.rainpoly is a poorly fitted model and almost all lags are insignificant except the intercept and z.t0.

2. The value of R2 is 0.2784.This means that model.rainpoly is able to explain only (27.84%) variation in the dependent variable (RBO).
3. The residuals have maximum value of 0.096239 and minimum value of -0.092950 with residual standard error (RSE) as 0.0414.

4. As per the F-test of the overall significance of model, the model.rainpoly is a moderately significant model at 5% level of significance despite having poor R2 value.

Calculating VIF for model.rainpoly DLM:

Calculating VIF for the model.rainpoly DLM to check for presence of multicollinearity in the model-

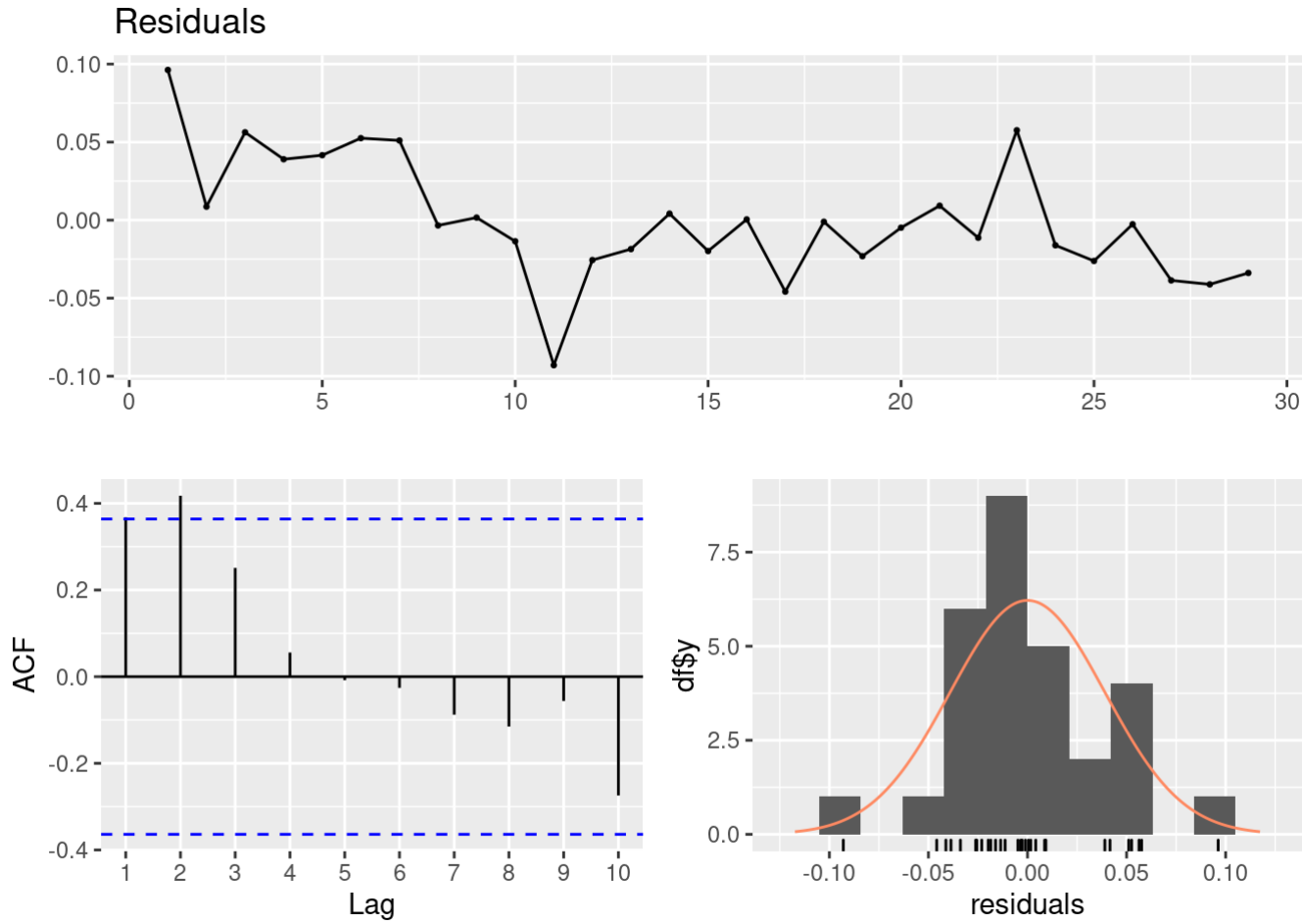
```
vif(model.rainpoly$model)
```

```
##      z.t0      z.t1      z.t2
##  3.588818 46.764244 35.865056
```

As seen above, there is presence of multicollinearity in the model.rainpoly because VIF>10.

Residual Analysis for model.rainpoly:

```
checkresiduals(model.rainpoly$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 Breusch-Godfrey test conducted, the p-value is greater than 5% level of significance which indicates that no autocorrelation exists among the residuals. The results obtained from the Breusch-Godfrey test are supported by the residuals plots generated as it is seen that residuals seem to follow a decreasing trend and are not distributed randomly. From the ACF plot, it is clearly seen that one lag is moderately significant which slightly supports the BG test. The histogram shows that residuals are not normal distributed.

Intepretation: The model.rainpoly is a significant model based on F-test and is the best in terms of AIC, BIC and MASE. However, it suffers from multicollinearity and has a poor R2 value.

Koyck DLM

Fittng all possible Koyck models:

The polynomial DLMs generated a poor fit model in terms of R2. Let's try to fit different koyck model that do not depend upon any lag length-

```
model.temkoyck = koyckDlm(x = as.vector(tem), y = as.vector(RBO))
model.rainkoyck = koyckDlm(x = as.vector(rain), y = as.vector(RBO))
model.humkoyck = koyckDlm(x = as.vector(hum), y = as.vector(RBO))
model.radkoyck = koyckDlm(x = as.vector(rad), y = as.vector(RBO))
```

Comparison based on AIC,BIC and MASE:

```
# Sorting based on  MASE


MASE_koyck2 <- MASE(model.temkoyck,model.rainkoyck,model.humkoyck,model.radkoyck)


# Sorting in ascending order

arrange(MASE_koyck2,MASE)
```

```
##                   n       MASE
## model.humkoyck   30  0.8400135
## model.temkoyck   30  0.9535116
## model.radkoyck   30  1.0314227
## model.rainkoyck 30 19.1057647
```

Based on MASE value, model.humkoyck is the best model. Let's analyze this model further-

Analyzing model.humkoyck:

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

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.081436 -0.017779 -0.005166  0.019919  0.101789
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.215367   1.204579   0.179   0.8594
## Y.1         0.548465   0.290591   1.887   0.0699 .
## X.t         0.002164   0.025588   0.085   0.9332
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03795 on 27 degrees of freedom
## Multiple R-Squared: 0.3452,  Adjusted R-squared: 0.2967
## Wald test:  6.62 on 2 and 27 DF,  p-value: 0.004576
##
## Diagnostic tests:
##                  df1 df2  statistic   p-value
## Weak instruments   1  27 1.19587955 0.2838077
## Wu-Hausman         1  26 0.04541097 0.8329122
##
##                              alpha        beta       phi
## Geometric coefficients:  0.4769667 0.002164239 0.5484653
```

From the above summary, we can generate following insights about the model.humkoyck-

1. The model model.humkoyck is a poorly fitted model and all of the lags are insignificant.
2. The value of R2 is 0.3452. This means that model.humkoyck is able to explain only 34.52% variation in the dependent variable (RBO).
3. The residuals have maximum value of 0.101789 and minimum value of -0.081436 with residual standard error (RSE) as 0.03795.
4. As per the F-test of the overall significance of model, the model.humkoyck is a pretty significant model at 5% level of significance.
5. The model in the first stage of least-squares estimation is insignificant at the 5% level, according to the Weak instruments test.
6. We may infer that there is an insignificant connection between the explanatory variable and the error term at the 5% level based on the Wu-Hausman test.
7. The geometric coefficients alpha, beta and phi are 0.4769667, 0.002164239 and 0.5484653 respectively.
8. This model is a better fit in comparison to other previous models in terms of R2.

Calculating VIF for model.humkoyck DLM:

Calculating VIF for the model.humkoyck DLM to check for presence of multicollinearity in the model-

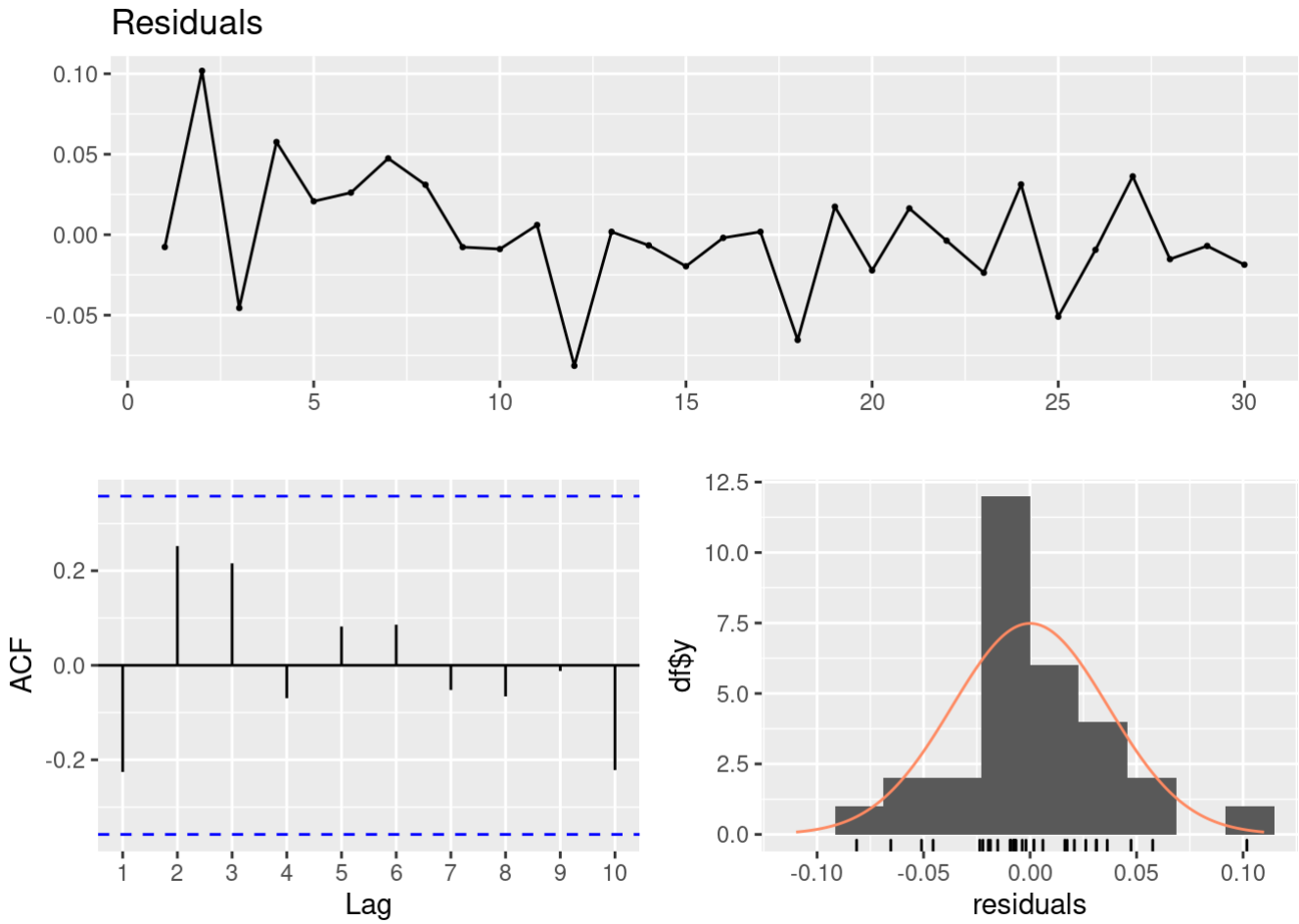
```
vif(model.humkoyck$model)
```

```
##      Y.1      X.t
## 3.449396 3.449396
```

As seen above, the VIF < 10 for all the lags in the model. Hence, there is no issue of multicollinearity in the model.humkoyck model.

Residual Analysis for model.humkoyck DLM:

```
# Residual analysis of model

checkresiduals(model.humkoyck$model)
```



As per the above residual plots generated, we can see that residuals seem to be following a negative trend. In ACF plot, no particular lag is significant which shows there is no presence of autocorrelation in the residuals. The residuals do not have normal distribution as per the histogram plot.

Intepretation: As per the diagnostic checking (residual analysis and other tests), Koyck DLM seems to be a better fit in comparison to the previous two models as the model is significant,have better R2 value and do not face any issues of multicollinearity.

Autoregressive DLM

Computing autoregressive DLMs for a range of lag lengths and AR process orders using the loop, and fitting the model with the lowest information criterion-

```
for (i in 1:5){
  for(j in 1:5){
    model.autoreg = ardlDlm(x= as.vector(tem),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
```

As per the above output, ARDL(5,5) is the best based on the AIC, BIC and MASE values. Using the finiteDLMauto function, all the other predictors were replaced to check the optimum p & q values. For all the predictor variables, it was found that optimum output is given by ARDL(5,5) only. **Thus, for all the autoregressive DLMs, p & q values will be taken as (5,5).**

Fitting ARDL(5,5) for all combination of predictors:

Fitting all possible combinations of ARDL models-

```
model.temardl = ardlDlm(x=as.vector(tem), y=as.vector(RBO), p = 5, q = 5)
model.rainardl = ardlDlm(x=as.vector(rain), y=as.vector(RBO), p = 5, q = 5)
model.humardl = ardlDlm(x=as.vector(hum), y=as.vector(RBO), p = 5, q = 5)
model.radardl = ardlDlm(x=as.vector(rad), y=as.vector(RBO), p = 5, q = 5)
```

Comparison based on AIC,BIC and MASE:

```
# Sorting based on AIC, BIC and MASE

sort.score(AIC(model.temardl$model,model.rainardl$model,model.humardl$model,model.radardl$mod
el), score = "aic")
```

```
##                       df       AIC
## model.temardl$model   13 -93.93583
## model.radardl$model   13 -93.64810
## model.humardl$model   13 -92.45536
## model.rainardl$model 13 -90.68282
```

```
sort.score(BIC(model.temardl$model,model.rainardl$model,model.humardl$model,model.radardl$mod
el), score = "bic")
```

```
##                       df       BIC
## model.temardl$model   13 -77.58057
## model.radardl$model   13 -77.29285
## model.humardl$model   13 -76.10010
## model.rainardl$model 13 -74.32757
```

```
Maseardl2 <- MASE(model.temardl,model.rainardl,model.humardl,model.radardl)

arrange(Maseardl2,MASE)
```

```
##                  n      MASE
## model.radardl   26 0.7145698
## model.temardl   26 0.7335979
## model.humardl   26 0.7370787
## model.rainardl 26 0.7469974
```

Based on AIC & BIC, model.temardl is the best model while model.radardl is the best model in terms of MASE. Let's analyze both these models further-

Analyzing ARDL(5,5) with tem as predictor:

```
summary(model.temardl)
```

```
##
## 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
## -0.047576 -0.006868  0.002825  0.018248  0.042903
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.549009   0.825391  -0.665   0.5168
## X.t         -0.014566   0.017365  -0.839   0.4157
## X.1          0.035046   0.019316   1.814   0.0911 .
## X.2          0.006488   0.019000   0.341   0.7378
## X.3         -0.012848   0.017678  -0.727   0.4794
## X.4          0.023252   0.015741   1.477   0.1618
## X.5         -0.006891   0.013378  -0.515   0.6145
## Y.1          0.494224   0.245359   2.014   0.0636 .
## Y.2          0.317096   0.293643   1.080   0.2985
## Y.3         -0.243288   0.317824  -0.765   0.4567
## Y.4          0.025812   0.325837   0.079   0.9380
## Y.5          0.332310   0.301469   1.102   0.2889
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03285 on 14 degrees of freedom
## Multiple R-squared:  0.6505, Adjusted R-squared:  0.3758
## F-statistic: 2.369 on 11 and 14 DF,  p-value: 0.06562
```

From the above summary, we can generate following insights about the model.temardl-

1.The model model.temardl is a decently fitted model. However, all lags are insignificant in the model including the intercept.

2. The value of R2 is 0.6505. This means that model.temardl is able to explain (65%) variation in the dependent variable (RBO).
3. The residuals have maximum value of 0.042903 and minimum value of -0.047576 with residual standard error (RSE) as 0.03285.
4. As per the F-test of the overall significance of model, the model.temardl is an insignificant model at 5% level of significance despite having a moderate R2 value.

VIF for model.temardl:

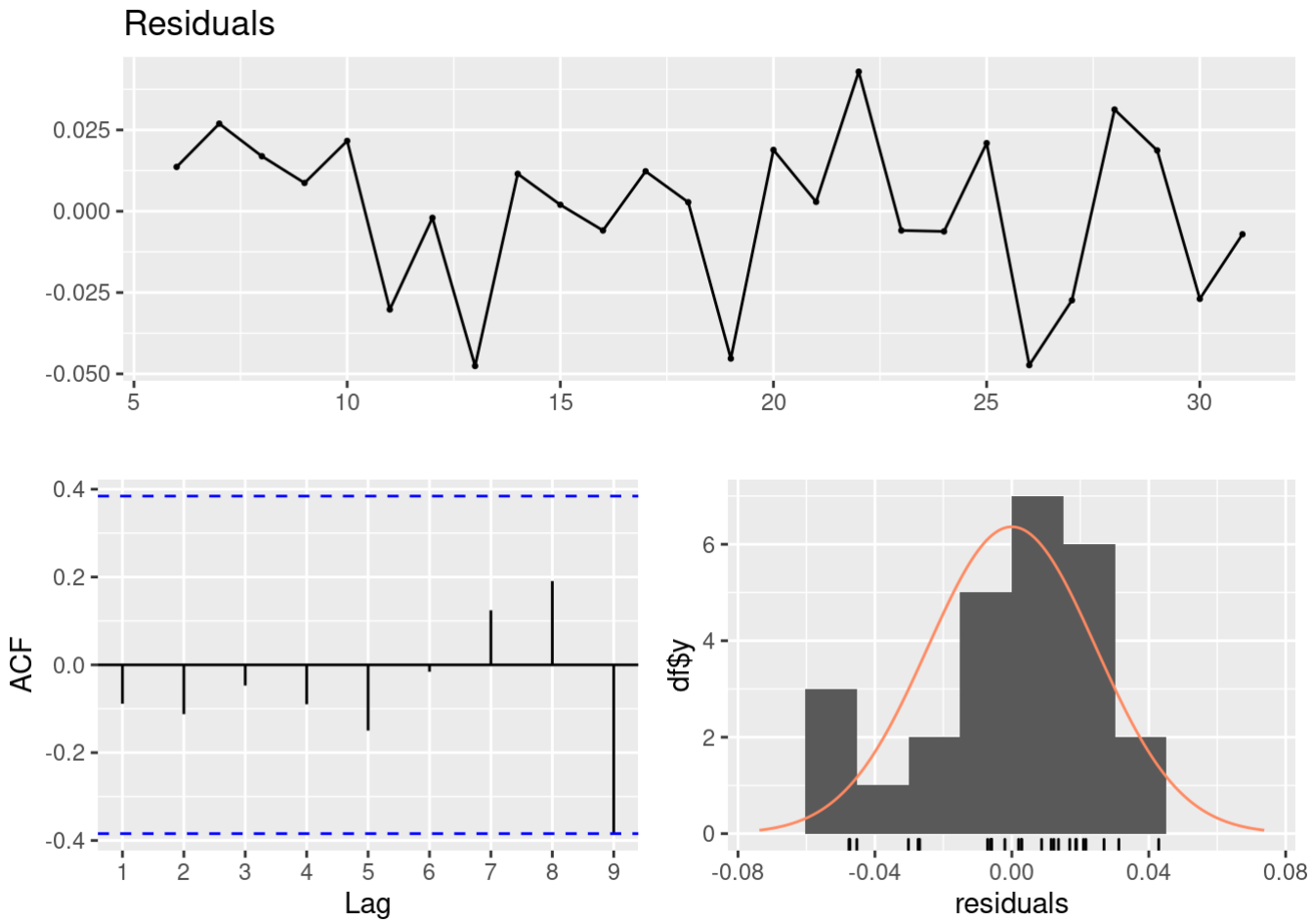
```
vif(model.temardl$model)
```

```
##       X.t L(X.t, 1) L(X.t, 2) L(X.t, 3) L(X.t, 4) L(X.t, 5) L(y.t, 1) L(y.t, 2)
##  2.727514  3.155850  2.900395  2.795929  2.236437  1.716485  2.627310  3.773171
## L(y.t, 3) L(y.t, 4) L(y.t, 5)
##  5.464592  5.705572  4.790602
```

As VIF is less than 10 for all elements in the model, there is no presence of multicollinearity in the model.

Residual Analysis (Diagnostic checking) of model.temardl:

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



From the above residual plots generated, we can see that residuals seem to be spread randomly without any particular pattern. In ACF plot, no particular lag is significant which shows there is no presence of autocorrelation in the residuals. The residuals do not have normal distribution as per the histogram plot.

Analyzing ARDL(5,5) with rad as predictor:

```
summary(model.radardl)
```

```
##
## 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 above summary, we can generate following insights about the model.radardl-

1.The model model.radardl is a decently fitted model but all its lags are insignificant including intercept.

2. The value of R2 is 0.6466. This means that model.radardl is able to explain (64.66%) variation in the dependent variable (RBO).
3. The residuals have maximum value of 0.039893 and minimum value of -0.058181 with residual standard error (RSE) as 0.03303.
4. As per the F-test of the overall significance of model, the model.radardl is an insignificant model at 5% level of significance despite having a moderate R2 value.

VIF for model.radardl:

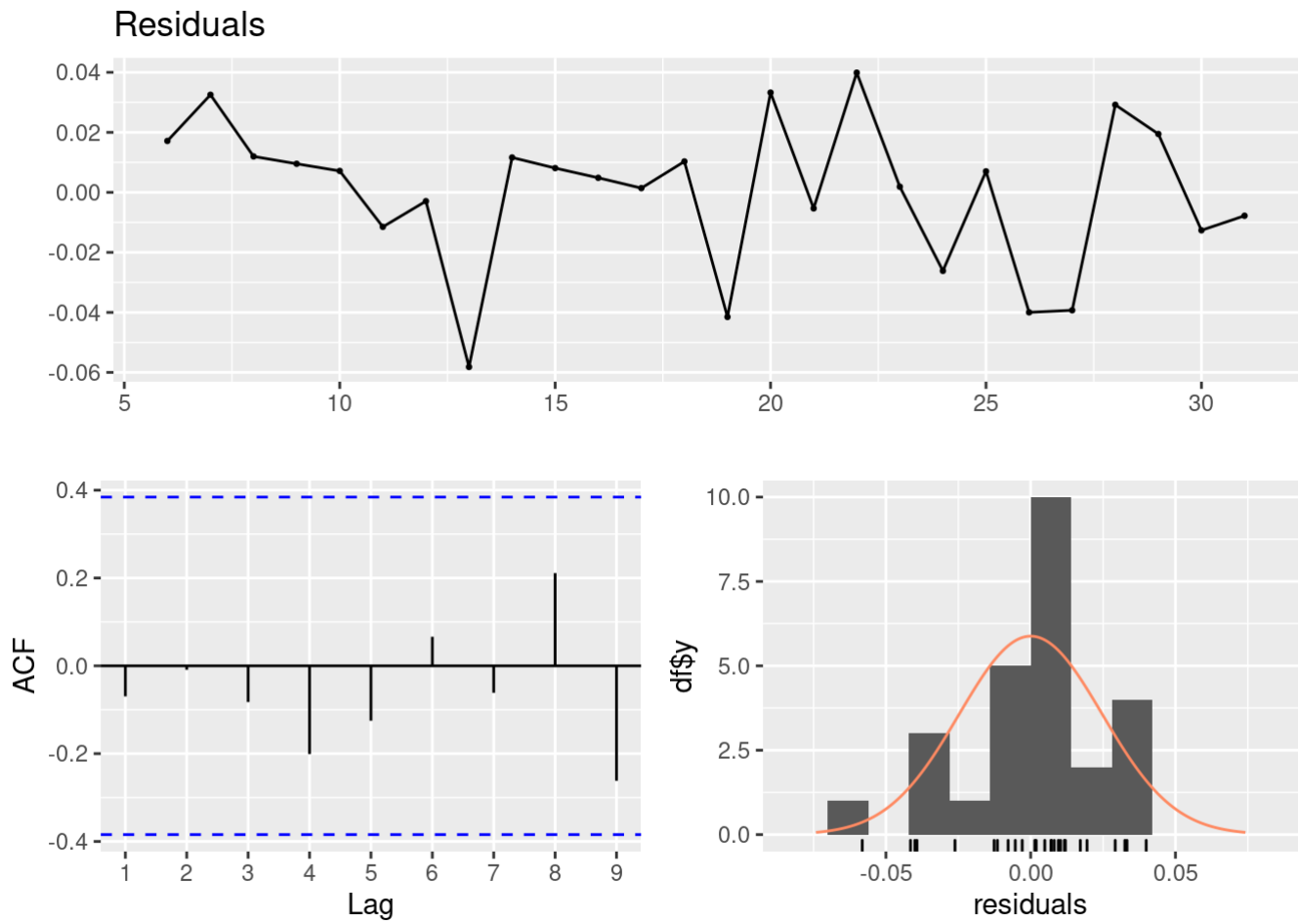
```
vif(model.radardl$model)
```

```
##       X.t L(X.t, 1) L(X.t, 2) L(X.t, 3) L(X.t, 4) L(X.t, 5) L(y.t, 1) L(y.t, 2)
##  1.884005  2.448098  2.454577  2.374717  2.484701  1.966329  2.901087  3.066142
## L(y.t, 3) L(y.t, 4) L(y.t, 5)
##  3.162682  2.626466  3.015791
```

As VIF is < 10 for all elements in the model, there is no presence of multicollinearity in the model.

Residual Analysis (Diagnostic checking) of model.radardl:

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



From the above residual plots generated, we can see that residuals seem to be spread randomly without following any particular pattern. In ACF plot, no particular lag is significant which shows there is no presence of autocorrelation in the residuals. The residuals do not have normal distribution as per the histogram plotted.

Interpretation: As per the diagnostic checking (residual analysis and other tests), both the autoregressive DLMs analyzed seem to be a poor fit in comparison to the previous Koyck model fitted. This is so because the models are insignificant even though they does not face any issue of multicollinearity and explain a decent variation in the dependent variable RBO.

Modelling using Dynamic Linear Models

The Dynamic Linear models are often considered suitable for forecasting. As there is no seasonality in the RBO series, we will only fit models with univariate independent variables (with & without intercept). Let's fit a few dynlm models-

```
# Fitting univariate models with RBO

dynamic1 <- dynlm(RBO ~ L(RBO , k = 1 )+ trend(RBO))
dynamic2 <- dynlm(RBO~ L(RBO , k = 2 )+ trend(RBO))
dynamic3 <-dynlm(RBO~ L(RBO , k = 1 ))
dynamic4 <-dynlm(RBO~ L(RBO , k = 1 )+ L(RBO , k = 2 ))
dynamic5 <-dynlm(RBO~ L(RBO , k = 1 )+ L(RBO , k = 2 )+L(RBO , k = 3 )+trend(RBO))
dynamic6 <-dynlm(RBO~ L(RBO , k = 1 )+ L(RBO , k = 2 )+L(RBO , k = 3 ) + L(RBO)+trend(RBO))

# Fitting models with tem as independent variable

temdynlm <-dynlm(RBO ~ tem + L(RBO, k=1))
temdynlm1 <-dynlm(RBO ~ tem + L(RBO, k=2))
temdynlm2 <-dynlm(RBO ~ tem)
temdynlm3 <-dynlm(RBO ~ tem - 1)
temdynlm4 <-dynlm(RBO ~ tem + L(tem, k=1))
temdynlm5 <-dynlm(RBO ~ tem + L(tem, k=2))
temdynlm6 <-dynlm(RBO ~ tem + L(tem, k=3))
temdynlm7 <-dynlm(RBO ~ tem + 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 hum as independent variable

humdynlm <- dynlm(RBO ~ hum + L(RBO, k=1))
humdynlm1 <- dynlm(RBO ~ hum + L(RBO, k=2))
humdynlm2 <- dynlm(RBO ~ hum)
humdynlm3 <- dynlm(RBO ~ hum - 1)
humdynlm4 <- dynlm(RBO ~ hum + L(hum, k=1))
humdynlm5 <- dynlm(RBO ~ hum + L(hum, k=2))
humdynlm6 <- dynlm(RBO ~ hum + L(RBO, k=3))
humdynlm7 <- dynlm(RBO ~ hum + L(hum, k=3))

# Fitting models with rad as independent variable

raddynlm <- dynlm(RBO ~ rad + L(RBO, k=1))
raddynlm1 <- dynlm(RBO ~ rad + L(RBO, k=2))
raddynlm2 <- dynlm(RBO ~ rad)
raddynlm3 <- dynlm(RBO ~ rad - 1)
raddynlm4 <- dynlm(RBO ~ rad + L(rad, k=1))
raddynlm5 <- dynlm(RBO ~ rad + L(rad, k=2))
raddynlm6 <- dynlm(RBO ~ rad + L(rad, k=3))
raddynlm7 <- dynlm(RBO ~ rad + L(RBO, k=3))
```

Comparing models based on MASE:

```
dynlm_MASE3 <- MASE(lm(temdynlm), lm(temdynlm1), lm(temdynlm2), lm(temdynlm3), lm(temdynlm4),
lm(temdynlm5),lm(temdynlm6),lm(temdynlm7),lm(raindynlm),lm(raindynlm1),lm(raindynlm2),lm(rain
dynlm3),
lm(raindynlm4),lm(raindynlm5),lm(raindynlm6),lm(raindynlm7),lm(humdynlm),lm(humdynlm1),lm(hum
dynlm2),
lm(humdynlm3),lm(humdynlm4),lm(humdynlm5),lm(humdynlm6),lm(humdynlm7),lm(raddynlm),lm(raddynl
m1),lm(raddynlm2),lm(raddynlm3),lm(raddynlm4),lm(raddynlm5),lm(raddynlm6),lm(raddynlm7),lm(dy
namic1),lm(dynamic2),lm(dynamic3),lm(dynamic4),lm(dynamic5))

arrange(dynlm_MASE3,MASE)
```

```
##                 n      MASE
## lm(dynamic1)   30 0.8252265
## lm(temdynlm)   30 0.8260073
## lm(humdynlm)   30 0.8261181
## lm(raddynlm)   30 0.8434230
## lm(raindynlm)  30 0.8528680
## lm(dynamic5)   28 0.8555020
## lm(dynamic3)   30 0.8559138
## lm(dynamic2)   29 0.8900758
## lm(humdynlm1)  29 0.8918682
## lm(dynamic4)   29 0.9008622
## lm(temdynlm1)  29 0.9129771
## lm(raindynlm1) 29 0.9225460
## lm(raindynlm4) 30 0.9417954
## lm(raddynlm1)  29 0.9641928
## lm(raindynlm6) 28 0.9688508
## lm(humdynlm4)  30 0.9858746
## lm(temdynlm2)  31 0.9918512
## lm(humdynlm6)  28 0.9980484
## lm(temdynlm4)  30 1.0056219
## lm(humdynlm2)  31 1.0134082
## lm(humdynlm3)  31 1.0139900
## lm(raddynlm7)  28 1.0175256
## lm(raindynlm2) 31 1.0280987
## lm(temdynlm7)  28 1.0546593
## lm(raddynlm4)  30 1.0630293
## lm(raddynlm2)  31 1.0799582
## lm(humdynlm5)  29 1.1087830
## lm(raindynlm5) 29 1.1090943
## lm(raddynlm6)  28 1.1295639
## lm(temdynlm5)  29 1.1311104
## lm(humdynlm7)  28 1.1393207
## lm(raindynlm7) 28 1.1481806
## lm(temdynlm6)  28 1.1660166
## lm(raddynlm5)  29 1.1667122
## lm(raddynlm3)  31 1.4041012
## lm(temdynlm3)  31 1.5252468
## lm(raindynlm3) 31 2.6180543
```

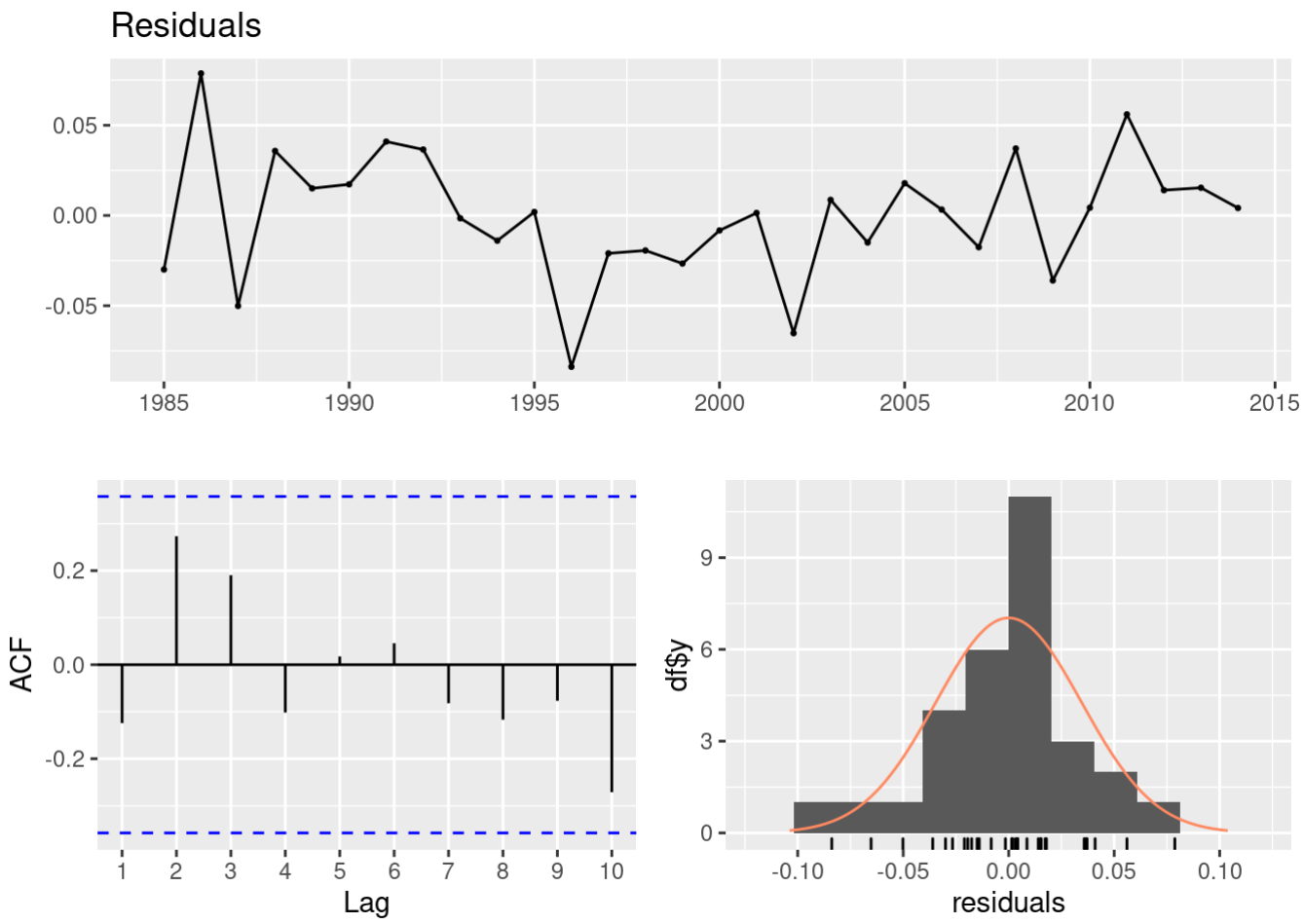
From the above results, let's analyze the dynamic1 model as it has the lowest MASE value out of all the models fitted-

Analyzing dynamic1 using summary & residual analysis:

```
# Generating summary of model

summary(dynamic1)
```

```
##
## 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
```

```
checkresiduals(dynamic1)
```



```
##
##  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 above summary of dynamic1 model, we can generate following insights-

1. All the terms in the model are significant at 5% level of significance.
2. The value of adjusted R2 is poor (37.17%). This means that dynamic model fitted is able to explain only 37.17% variation in the dependent variable (RBO).
3. The residuals have maximum value of 0.078745 and minimum value of -0.083854 with residual standard error (RSE) as 0.415 which is not better in comparison to previous models.
4. As per the F-test of the overall significance of model, the dynamic1 is a significant model at 5% level of significance.
5. In comparison to other DLM models, the dynamic model generated decent summary statistics.

Residual Analysis:The BG test indicate no serial correlation is there in the residuals.The residuals plot indicate that residuals are randomly distributed. In the ACF plot, there are no significant lags which support BG test. As a result, no autocorrelation found in the residuals. The histogram shows residuals do not seem to follow normal distribution.

VIF for dynamic1:

```
vif(dynamic1)
```

```
## L(RBO, k = 1)    trend(RBO)
##      1.450819      1.450819
```

As VIF is less than 10 for all elements in the model, there is no presence of multicollinearity in the model

Interpretation: The model dynamic1 obtained using Dynamic linear method is a significant model in terms of F-test and it has best MASE accuracy score than other dynlm models fitted. However, like previous models, it has poor R2 value.

Summary Table Task 3

Let's compare all the models fitted so far and compute the best model based on MASE score-

```
# Generating table

final_table2 <- data.frame(Model = c("model.rain","model.nointercepthum","model.rainpoly","mo
del.humkoyck","model.temardl","model.radardl","dynamic1"), MASE= c(0.9417954,0.9918515,0.9993
747,0.8400135,0.7335979,0.7145698,0.8252265))


# Comparing MASE score

arrange(final_table2,MASE)
```

```
##                  Model      MASE
## 1        model.radardl 0.7145698
## 2        model.temardl 0.7335979
## 3             dynamic1 0.8252265
## 4       model.humkoyck 0.8400135
## 5           model.rain 0.9417954
## 6 model.nointercepthum 0.9918515
## 7       model.rainpoly 0.9993747
```

Best Model: Based on the MASE score, the best models generated out of all the DLM and dynamic linear methods are sorted in terms of MASE. We can see that the best MASE score is given by both ARDL method models (model.radardl & model.temardl). Let's consider both these models for the purpose of forecasting RBO for the next 3 years.

Forecasting RBO

For generating the forecasts for the next 3 years, we will use the values of radiation & temperature in their respective models for the next three years from the `task3_covariate` data.

Three years ahead values for both radiation and temperature:

```
# The radiation values after 2014 are-


  task2_covariate[,4]
```

```
## # A tibble: 4 × 1
##   Radiation
##       <dbl>
## 1      14.6
## 2      14.6
## 3      14.8
## 4      14.8
```

```
# The temperature values after 2014 are-

task2_covariate[,2]
```

```
## # A tibble: 4 × 1
##   Temperature
##         <dbl>
## 1        20.7
## 2        20.5
## 3        20.5
## 4        20.6
```

Computing Prediction intervals & point forecasts for model.radardl:

```
# Using the first three values of radiation from task3_covariate to compute the three year ah
ead RBO

# Generating point forecasts

forecastRBOpoint <- dLagM::forecast(model.radardl,x = c(14.60,14.56,14.79) ,h = 3)

forecastRBOpoint <- round(forecastRBOpoint$forecasts,2)

# Generating prediction intervals

forecastRBO <- forecast(model = model.radardl , x = c(14.60,14.56,14.79) ,
                h = 3 , interval = TRUE)

round(forecastRBO$forecasts,2)
```

```
##   95% LB Forecast 95% UB
## 1   0.63     0.70   0.76
## 2   0.66     0.72   0.78
## 3   0.67     0.73   0.79
```

The above shows 95% prediction intervals and point forecasts for the three years ahead period from 2015-2017 using model.radardl. The prediction/confidence intervals are not very wide and point forecasts seem to be reliable.

Computing Prediction intervals & point forecasts for model.temardl:

```
# Using the first three values of temperature from task3_covariate to compute the three year
 ahead RBO

# Generating point forecasts

forecastRBOpoint2 <- dLagM::forecast(model.temardl,x = c(20.74,20.49,20.52) ,h = 3)

forecastRBOpoint2 <- round(forecastRBOpoint2$forecasts,2)

# Generating prediction intervals

forecastRBO2 <- forecast(model = model.temardl , x = c(20.74,20.49,20.52) ,
                h = 3 , interval = TRUE)

round(forecastRBO2$forecasts,2)
```

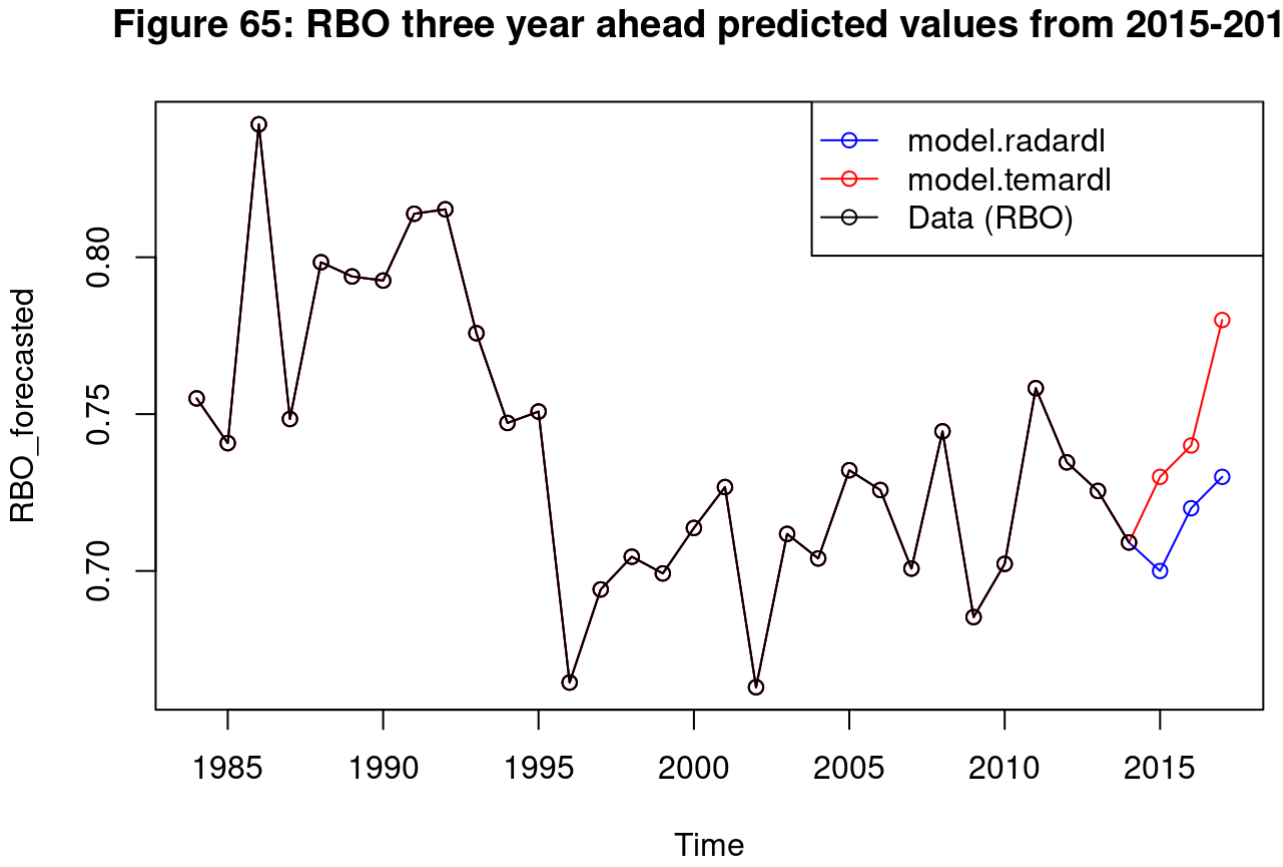
```
##   95% LB Forecast 95% UB
## 1   0.66     0.73   0.79
## 2   0.68     0.74   0.82
## 3   0.71     0.78   0.86
```

The above shows 95% prediction intervals and point forecasts for the three years ahead period from 2015-2017 using model.temardl. The prediction/confidence intervals are not very wide and point forecasts seem to be reliable.

Plotting the three years ahead forecasts using both models:

Let's plot both the models together along with the original RBO series-

```
plot(ts(c(as.vector(RBO),forecastRBOpoint),start=1984),type="o",col="blue", ylab="RBO_forecas
ted",
main="Figure 65: RBO three year ahead predicted values from 2015-2017")
lines(ts(c(as.vector(RBO),forecastRBOpoint2),start=1984),type="o",col="red")
lines(ts(as.vector(RBO),start=1984),col="black",type="o")
legend("topright", lty = 1,pch=1,text.width=11, col = c("blue","red","black"), c("model.radar
dl","model.temardl","Data (RBO)"))
```



From Figure 60 we can say that the predictions generated by both the fitted models are a little different from each other. As per model.radardl, the RBO value will decrease in 2015 and then will follow an increasing pattern in the next two years. As per model.temardl, the RBO values will only follow an increasing trend in the next three years.

Additionally, both the models fit the previous values of RBO accurately. Moreover, the predictions generated by both of them are reliable because their 95% prediction interval is also not wide. In conclusion, we can say that RBO values are going to increase in the next three years as per both the ARDL models fitted. This means, higher similarity of the order of the first flowering occurrence is predicted in the next three years from 2015-2017.

Task 3 Part(b)- Model Fitting for Drought affected RBO

Let's now forecast the drought affected RBO series-

Modelling the drough effect

Fitting intervention dynlm models to predict RBO while including the drought effect on RBO which started from 1996 to 2009 (13 years).

```
# Including the pulse affect when drought happened (1996)

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 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~tem + L(Y.t , k = 1 ) + L(Y.t , k = 2 )+ S.t+ + S.t.1 + trend(Y.t))
model11 = dynlm(Y.t~rad + L(Y.t , k = 1 ) + L(Y.t , k = 2 )+ S.t+ + S.t.1 + trend(Y.t))
model12 = dynlm(Y.t~hum + L(Y.t , k = 1 ) + L(Y.t , k = 2 )+ S.t+ + S.t.1 + trend(Y.t))
```

Let’s now compare all these models in terms of MASE-

Comparing 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(model12) 29 0.6639501
## lm(model11) 29 0.6734863
## lm(model8)  28 0.6748310
## lm(model10) 29 0.6762744
## lm(model1)  30 0.6768533
```

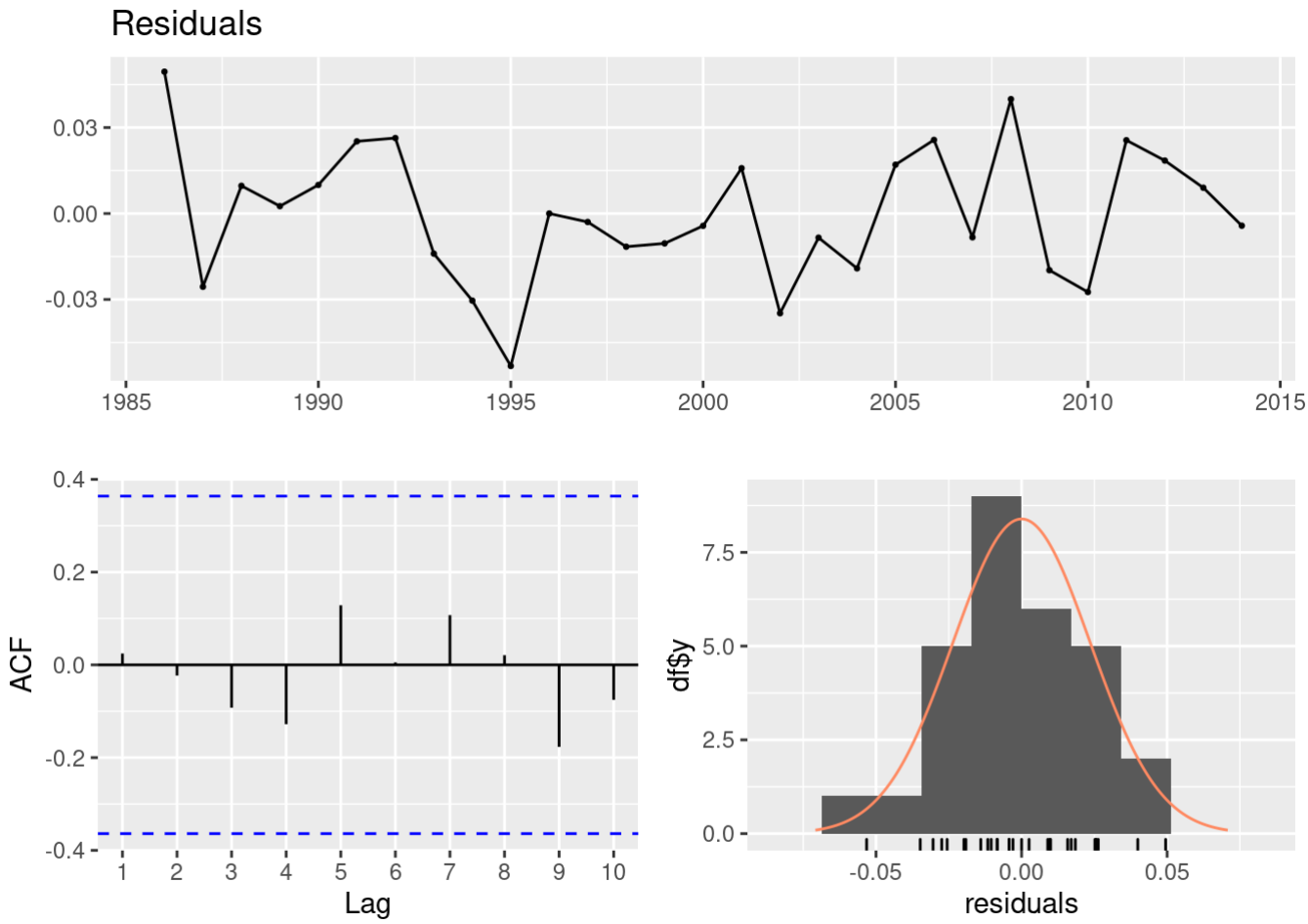
From the above results, let’s analyze the model9(with rain as predictor) as it has the lowest MASE value out of all the models fitted.

Analyzing model9 using summary & residual analysis:

```
# Generating summary of model

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
```

```
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 above summary of model9, we can generate following insights-

1. Most of the terms in the model are insignificant at 5% level of significance except the intercept and S.t.
2. The value of adjusted R2 is decent (66.55%). This means that dynamic model fitted is able to explain 66.55% variation in the dependent variable (RBO).
3. The residuals have maximum value of 0.049497 and minimum value of -0.053222 with residual standard error (RSE) as 0.02663.
4. As per the F-test of the overall significance of model, the model9 is a highly significant model at 5% level of significance.

Residual Analysis:As per the BG test, no serial correlation exist among the residuals.The residuals seem to be randomly distributed. In the ACF plot, there is no significant lag. As a result, no obvious autocorrelation found in the residuals. The histogram shows residuals do not seem to follow normal distribution.

Interpretation: The intervention model model9 (with rain as predictor) obtained using Dynamic linear method is a significant model in terms of F-test and have a decent R2 & MASE value. This model will be considered further for 3 year prediction of RBO with drought effect in place-

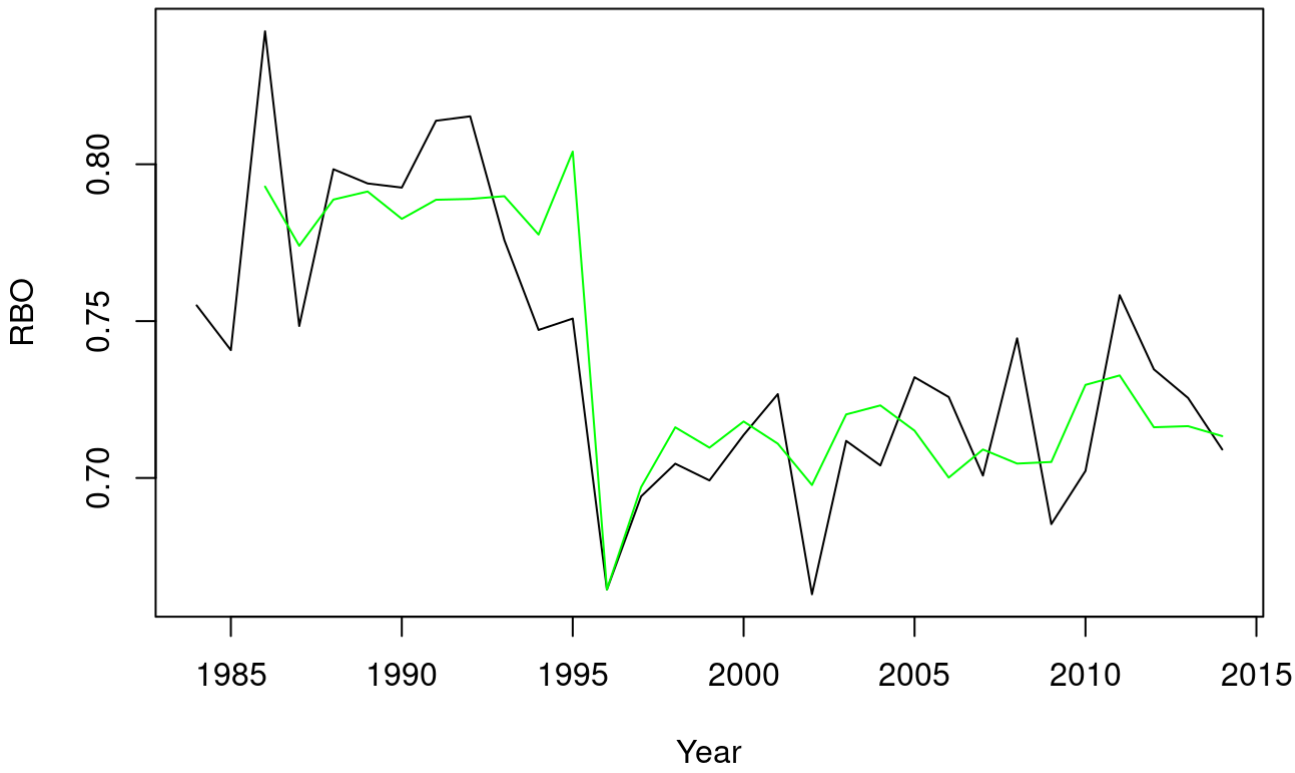
Forecasting

Let’s now use our dynlm model model9 to predict three year ahead RBO values.

Time series plot of the fitted model with the original series:

```
par(mfrow=c(1,1))
plot(Y.t,ylab='RBO',xlab='Year',main = "Figure 66: Time series plot of the  yearly RBO with f
itted model9")
lines(model9$fitted.values,col="green")
```

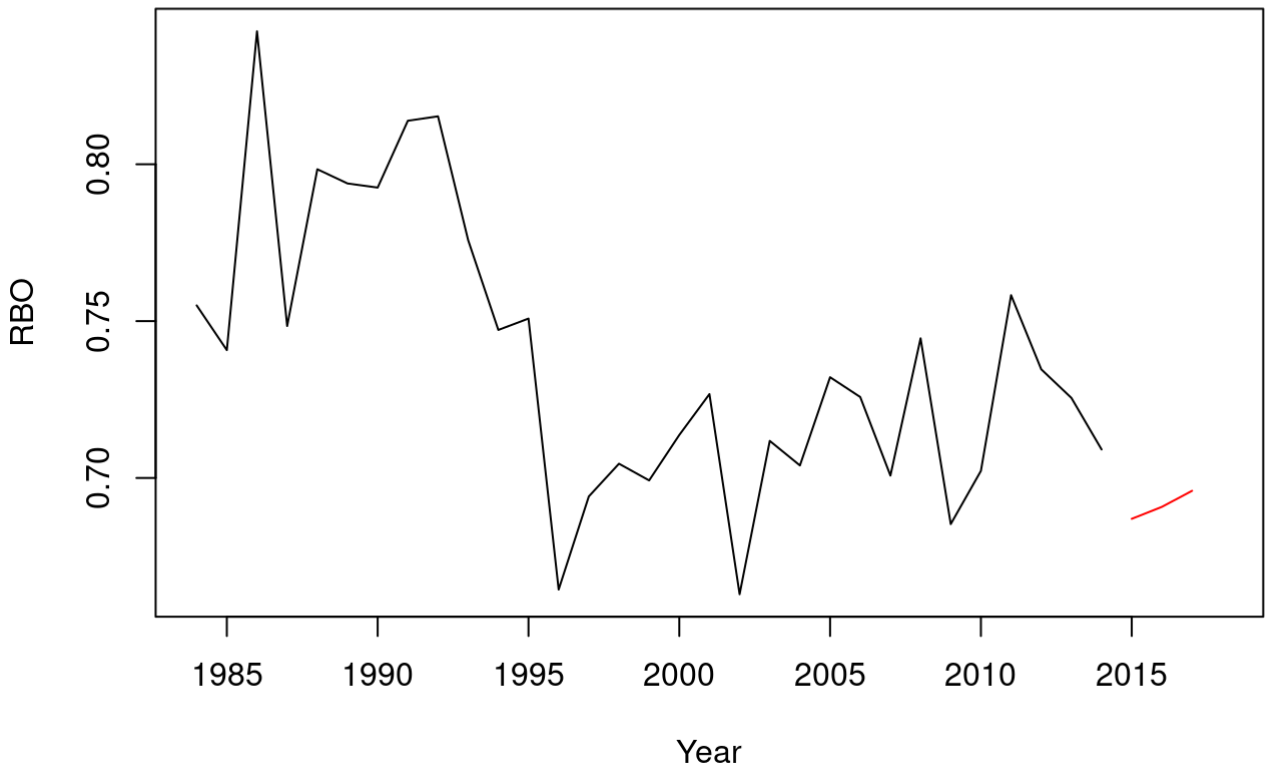
Figure 66: Time series plot of the yearly RBO with fitted model9



As we can see above, the model is not able to exactly fit all the previous values of RBO. However, it is the best computed in terms of MASE, R2 and F-test. Let's compute the forecast for the next 3 years using dynamic model-

```
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 67: Time series plot of RBO with 3 years ahead forecast")
lines(ts(rbo.frc[(n+1):(n+q)],start=c(2015)),col="red")
```

Figure 67: Time series plot of RBO with 3 years ahead forecast



The above plot shows four year ahead forecast using model9. As per the forecast, RBO values will be lower than previous average RBO values in the next three years as per the dynamic linear model due to drought effect.

```
# Forecast for next three years

round(rbo.frc[(n+1):(n+q)],2)
```

```
## [1] 0.69 0.69 0.70
```

The above output shows predicted values for the next three years. The predicted values of RBO will lie between 0.69-0.70. The values are lower than average RBO values than previous years. This can be the result of the drought effect which lasted for 13 years (between 1996-2009). But as seen that the model did not fit the RBO series well, these predictions cannot be considered accurate.

Conclusion

Task 3 analysis was successfully performed on the RBO.csv data. For part (a) Uni-variate analysis was performed on all the five series (RBO, temperature, rainfall, radiation, relhumidity). The five series could not be decomposed as their frequency was less than 2. Different uni variate models were fitted for the purpose of predicting the RBO values for the next 3 years (2015-2017) using different modelling methods(DLM,koyck,polyDLM,ARDL & dynlm). The best model computed in terms of R2, F-test, AIC, BIC and MASE value were found to be two ARDL models with temperature & radiation as respective predictors. Both the models were used for forecasting the three year ahead forecast for RBO. Both the models perfectly fitted the previous RBO values. However, the model with temperature predicted that RBO value will only follow an

increasing pattern while model with radiation predicted that RBO will decrease in first year and then will increase in the next two years (2016-2017). From both the models, it can be concluded that RBO values will have a gradual increase in the next three years (2015-2017).

For part(b), the univariate dynlm models were fitted after incorporating the drought effect that lasted for 13 years (from 1996-2009). The best model was computed in terms of MASE, R2 & F-test and it was found to be model9 with rain as predictor. The model9 was then fitted with the original series and we found that it was deviating at an extreme degree from the original series. Thus, the model9 was found inaccurate for forecasting the next three years values for RBO as it did not fit the previous values of RBO accurately. However, as it was the best model, thus, it was further used to forecast the next three year RBO values. The values generated were lower than average RBO values which could be due to the drought effect that lasted for thirteen years between 1996-2009. Since, the model was not fitting the RBO series accurately, the forecast generated were not completely reliable.

References

1.) Irene Hudson's presentation slides and lab solutions for MATH1307: Forecasting.

2.) Dr. Haydar Demirhan notes for MATH1318: Time Series Analysis.

3.) Correlation matrix : A quick start guide to analyze, format and visualize a correlation matrix using R software - EasyGuides - Wiki -STHDA. (2021). Retrieved 18 October 2021, from http://www.sthda.com/english/wiki/correlation-matrix-a-quick-start-guide-to-analyze-format-and-visualize-a-correlation-matrix-using-r-software (http://www.sthda.com/english/wiki/correlation-matrix-a-quick-start-guide-to-analyze-format-and-visualize-a-correlation-matrix-using-r-software) .

Appendix

1.) The packages used were TSA, fUnitRoots,tseries, readr, dplyr, magrittr,tidyr,stats,forecast,urca,car,x12,xts,stats,dLagM,zoo and hmisc.

2.) Abbreviations used are given below:

a. **AR** - Auto Regressive model
b. **MA** - Moving average model
c. **AIC**- Akaike information criterion
d. **BIC**- Bayesian information criterion
e. **MASE**- Mean Absolute Scaled Error
f. **R2**- Coefficient of Determination
g. **MAPE**- Mean Absolute Percentage Error
h. **RMSE**- Root Mean Square Deviation