Forecasting Final Assignment

Ratnak Saha (S3973869)

```
library(TSA)
library(urca)
library(readr)
library(magrittr)
library(tseries)
library(x12)
library(forecast)
library(seasonal)
library(dplyr)
library(lubridate)
library(tidyr)
library(car)
library(dlm)
library(dLagM)
library(dynlm)
library(lmtest)
library(Hmisc)
library(xts)
library(zoo)
library(corrplot)
library(PerformanceAnalytics)
```

Task 1

Description:

- Researchers evaluated disease-specific mortality in Paris, France, and its link with local climate conditions and pollution levels in a large study that spanned the years 2010 to 2020. This study aims to anticipate mortality rates four weeks ahead considering the mortality dataset for this investigation, which includes weekly data for all five key series.

Objective of Task1

- Understanding the nature of time series data through ACF and ADF test

Importing dataset

```
getwd()
```

```
## [1] "F:/2nd semester/Forecasting/Final_assignment"
```

```
setwd("F:/2nd semester/Forecasting/Final_assignment")
mortality<- read.csv("mortality.csv")
head(mortality)
```

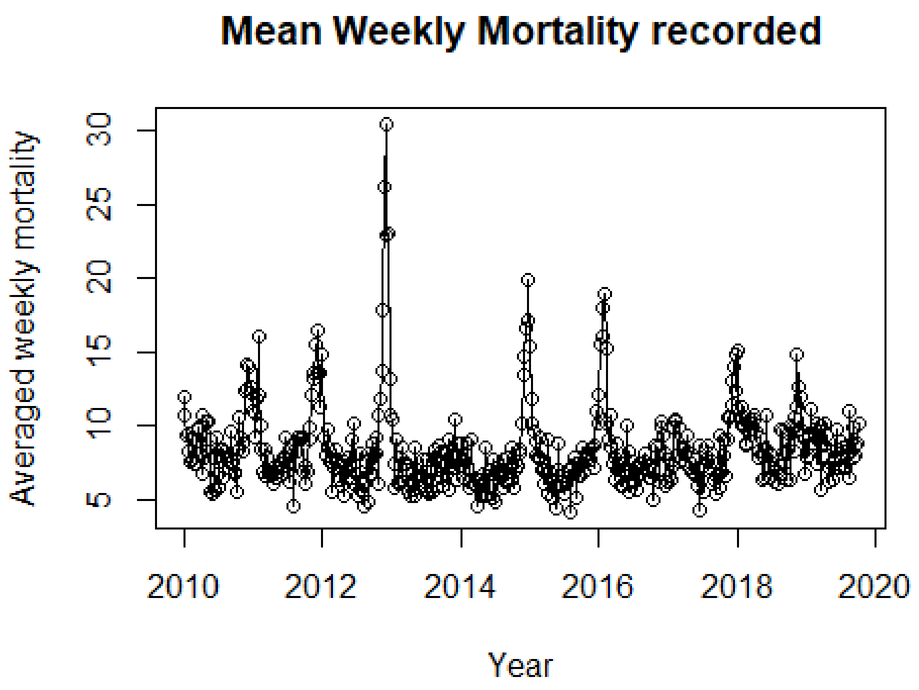
```
##   X mortality  temp chem1 chem2 particle.size
## 1 1     11.90 72.38  3.37 45.79         72.72
## 2 2     10.75 67.19  2.59 43.90         49.60
## 3 3      9.33 62.94  3.29 32.18         55.68
## 4 4      9.54 72.49  3.04 40.43         55.16
## 5 5      8.27 74.25  3.39 48.53         66.02
## 6 6      7.55 67.88  2.57 48.61         44.01
```

Converting variables into separate time series

```
mort_ts <- ts(mortality$mortality, start = c(2010,1), frequency = 52)
temp_ts <- ts(mortality$temp, start = c(2010,1), frequency = 52)
chem_1 <- ts(mortality$chem1, start = c(2010,1), frequency = 52)
chem_2 <- ts(mortality$chem2, start = c(2010,1), frequency = 52)
particle <- ts(mortality$particle.size, start = c(2010,1), frequency = 52)

class(mort_ts)

## [1] "ts"

class(temp_ts)

## [1] "ts"

class(chem_1)

## [1] "ts"

class(chem_2)

## [1] "ts"

class(particle)

## [1] "ts"
```

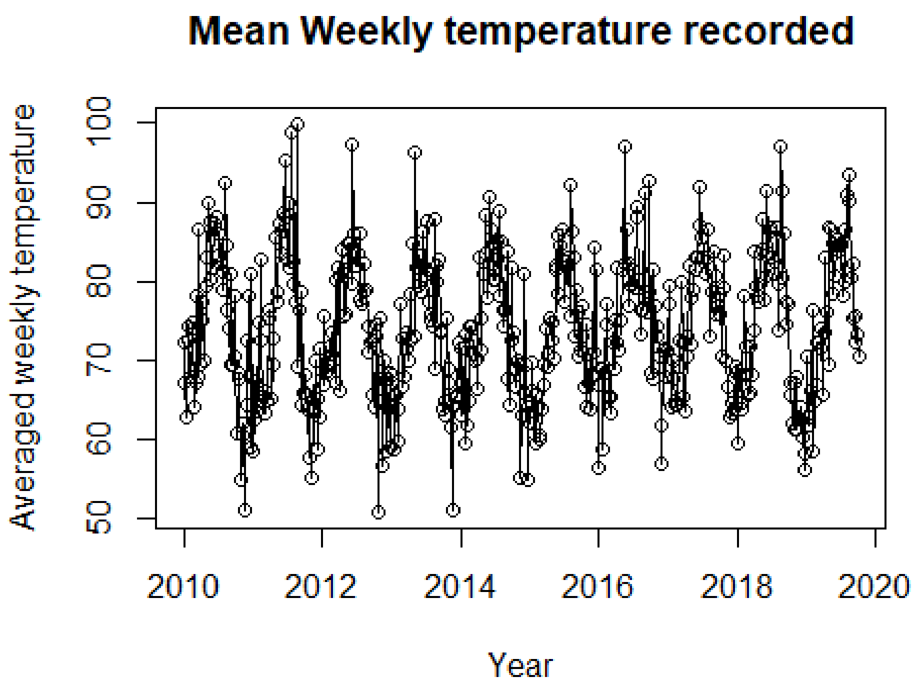
Determining the nature of series

```
par(mfrow=c(1,1))
plot(mort_ts, ylab="Averaged weekly mortality", xlab = "Year", main =
"Mean Weekly Mortality recorded", type ="o")
```



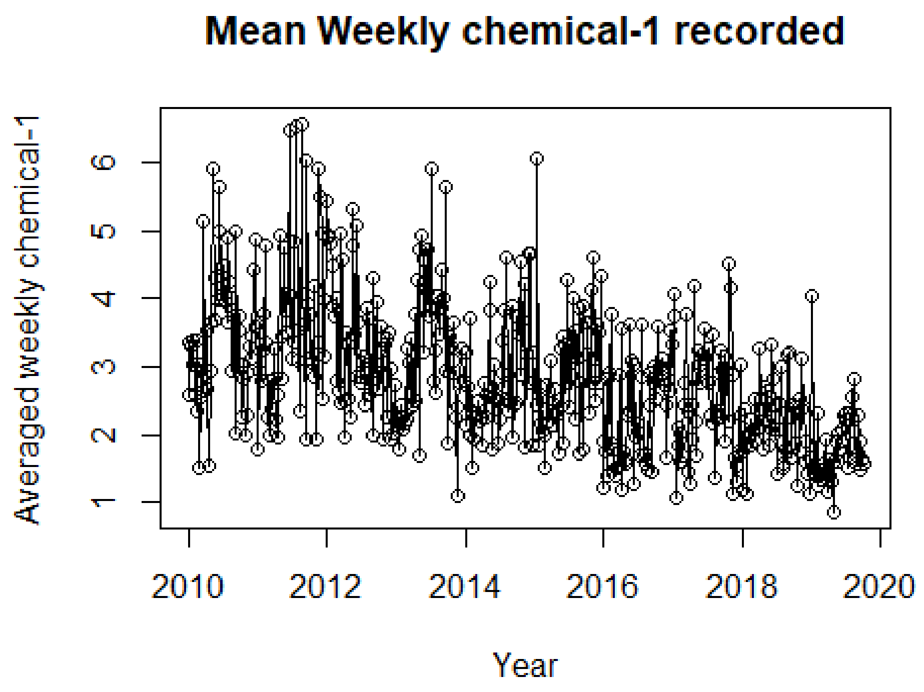
* The time series data shows a clear seasonal pattern with no evident trend or variance change. However, there is a notable intervention around 2013, suggesting a significant external event or policy change influenced the data. Investigating this intervention is crucial for understanding its impact on the data and its implications.

```
par(mfrow=c(1,1))
plot(temp_ts, ylab="Averaged weekly temperature", xlab = "Year", main =
"Mean Weekly temperature recorded", type ="o")
```



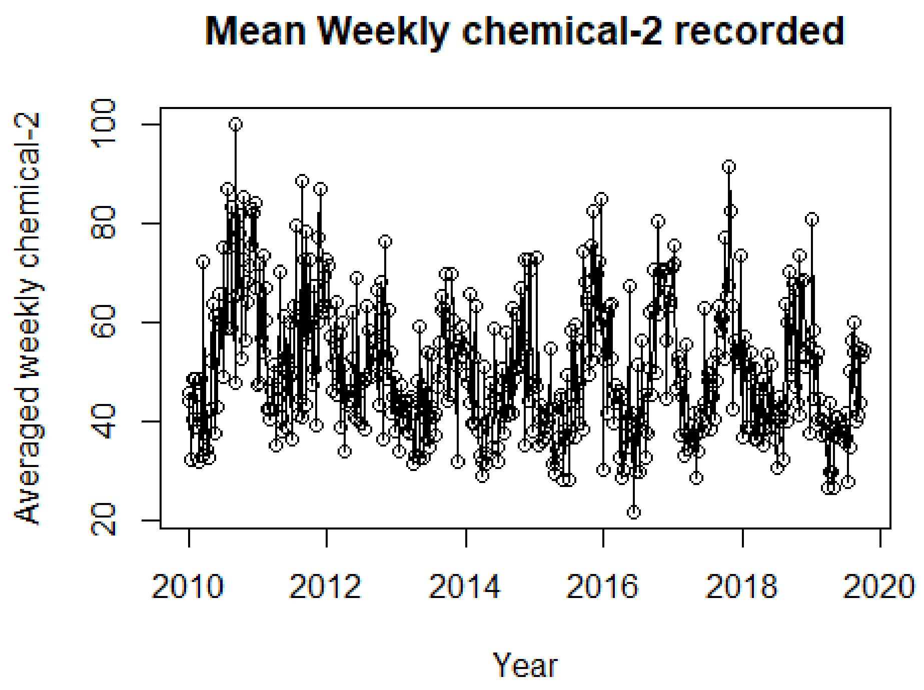
* The time series data show a seasonal pattern with variations, but there is no indication of a trend or changes in variance. Notably, it can be observed that the increased values in the first weeks of 2011, 2013, and late 2013, as well as the first weeks of 2017, and these spikes do not appear to be connected with any apparent interventions or external influences.

```
par(mfrow=c(1,1))
plot(chem_1, ylab="Averaged weekly chemical-1", xlab = "Year", main =
"Mean Weekly chemical-1 recorded", type ="o")
```



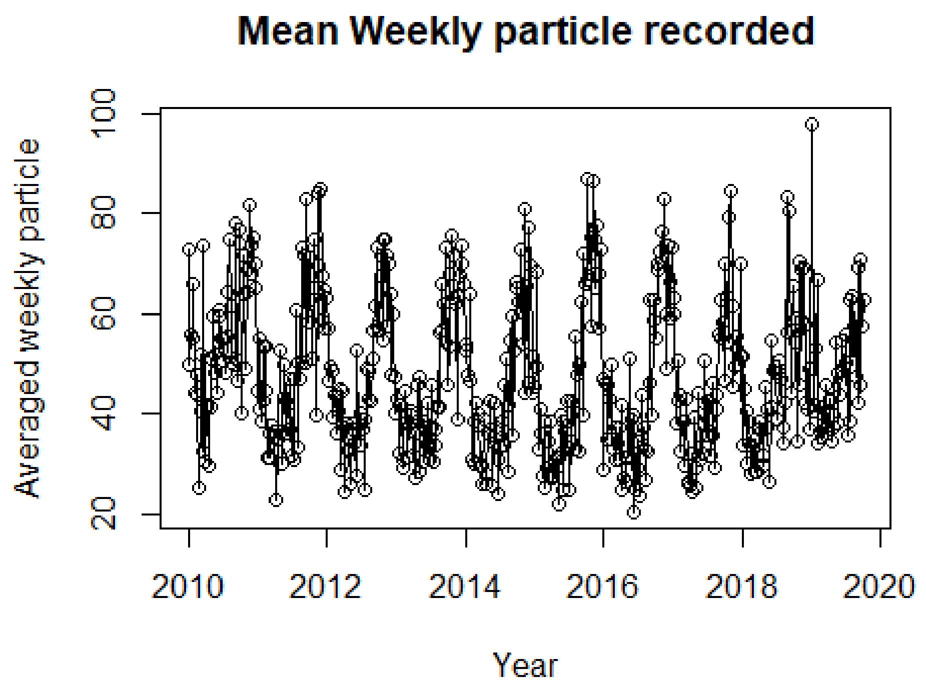
* The time series data displays a pronounced seasonal pattern, accompanied by a declining trend and reduced variance over time. Despite these changes, there is no apparent intervention or significant external influence on the data. The presence of seasonality indicates regular, recurring patterns, making it a notable feature in the dataset. Notably,it can be observed that the increased values in the later weeks of 2011 and 2013.

```r
par(mfrow=c(1,1))
plot(chem_2, ylab="Averaged weekly chemical-2", xlab = "Year", main =
"Mean Weekly chemical-2 recorded", type ="o")
```



* The time series data exhibits a consistent seasonal pattern without any discernible long-term trend or variance change. In this case, there is no clear evidence of a significant intervention or external factor influencing the data. The seasonality suggests regular, cyclical patterns that repeat over time, but there is no unusual behavior observed in the dataset. Notably, it can be observed that the increased values in the later weeks of 2010, 2011, 2015 and 2017.

```
par(mfrow=c(1,1))
plot(particle, ylab="Averaged weekly particle", xlab = "Year", main =
"Mean Weekly particle recorded", type ="o")
```

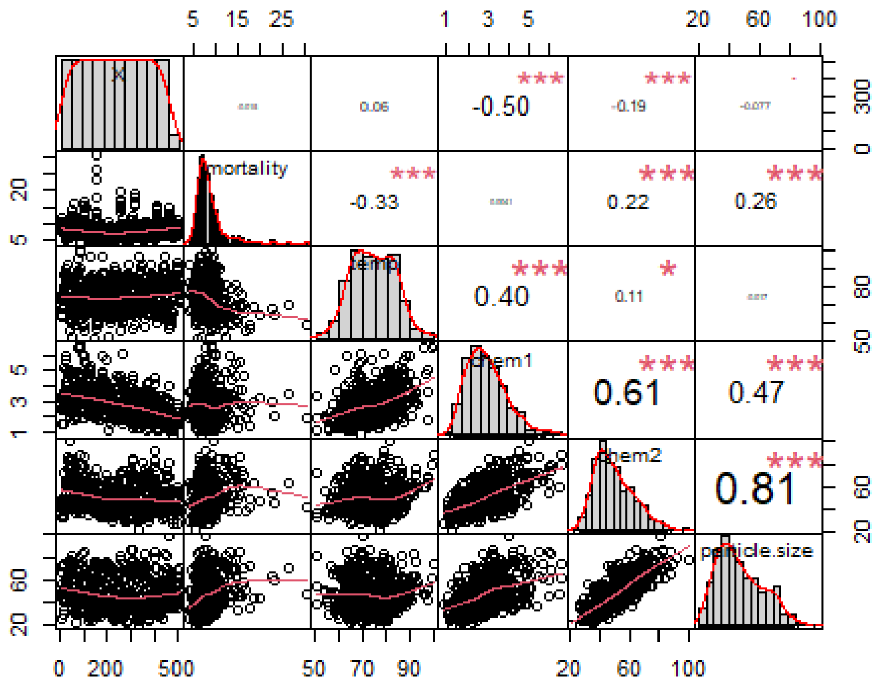


* The time series data displays a consistent seasonal pattern over time, with no noticeable extended trend or significant variation change. There is no clear indication of any noteworthy intervention or external factor impacting the data. Notably, it can be observed that the increased values in the later weeks of 2010, 2011 and continues till 2019.

Time Series Plot

```
mortality_new <- ts(mortality, start = c(2010,1), frequency = 52)
chart.Correlation(mortality_new)
```

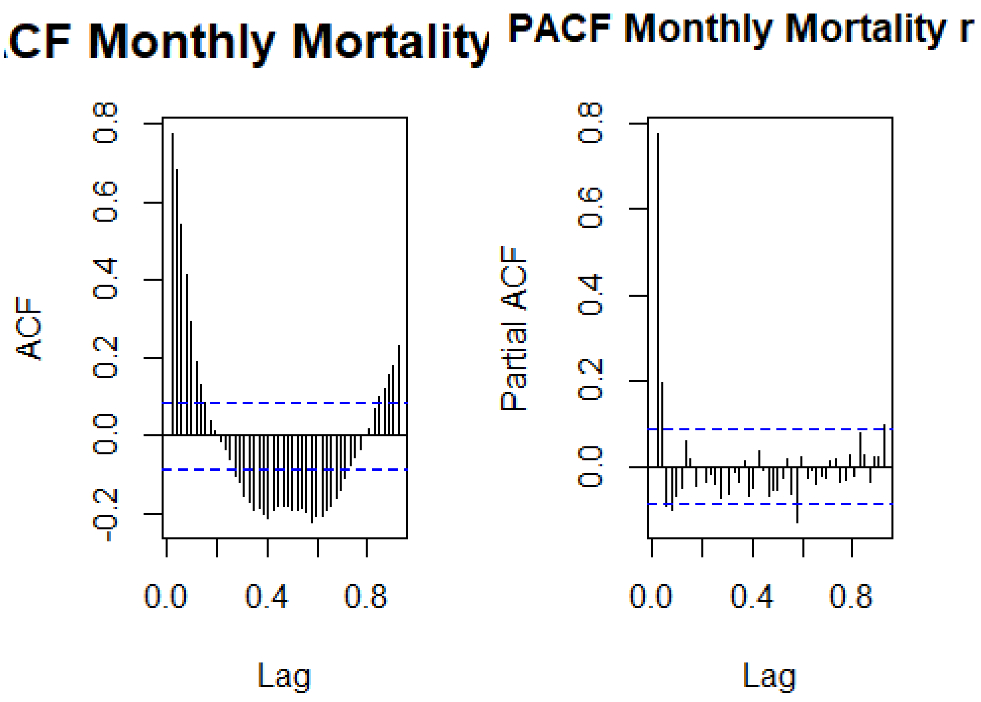
```
## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter

## Warning in par(usr): argument 1 does not name a graphical parameter
```



* Chemical 2 and Particle exhibits strong correlation with each other and other variables exhibit low to moderate correlation.

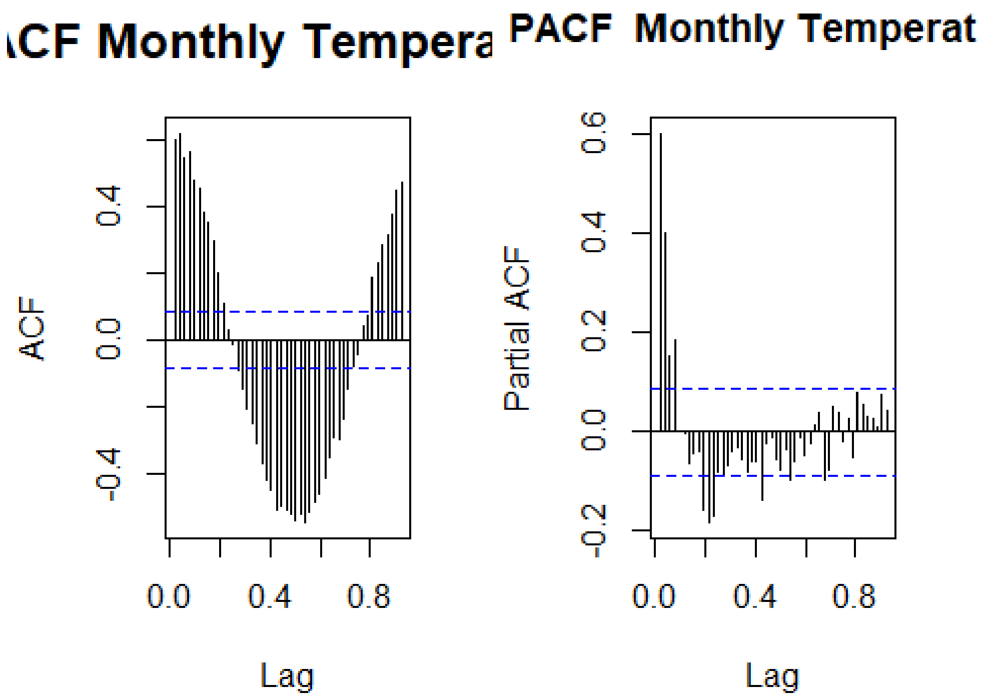
Analysing Non-Stationarity in dataset

```
par(mfrow=c(1,2))
acf(mort_ts, lag.max = 48, main = "ACF Monthly Mortality rate", cex.main = 1.5)
pacf(mort_ts, lag.max = 48, main = "PACF Monthly Mortality rate", cex.main = 1.5)
```



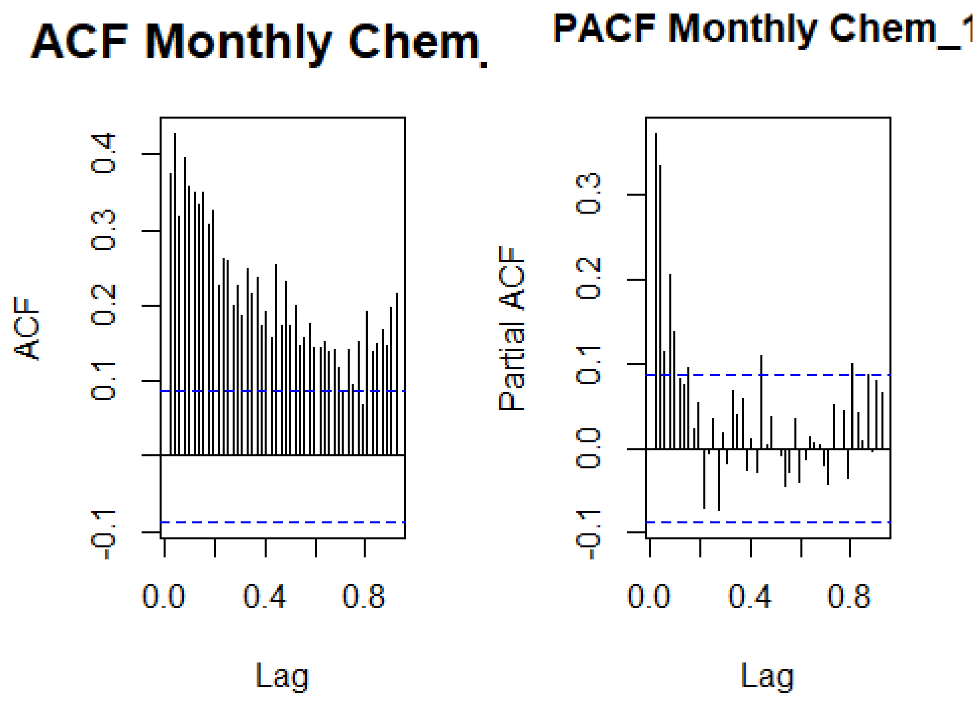
* The ACF plot shows signs of declining trend in the time series data. There is a possibility of a seasonal pattern, and many of the lag values are well above the 95% confidence interval.On the other hand, when looking at the PACF plot, the first lag appears highly significant, suggesting that the series is not stationary.

```
par(mfrow=c(1,2))
acf(temp_ts, lag.max = 48, main = "ACF Monthly Temperature", cex.main = 1.5)
pacf(temp_ts, lag.max = 48, main = "PACF  Monthly Temperature", cex.main = 1.5)
```



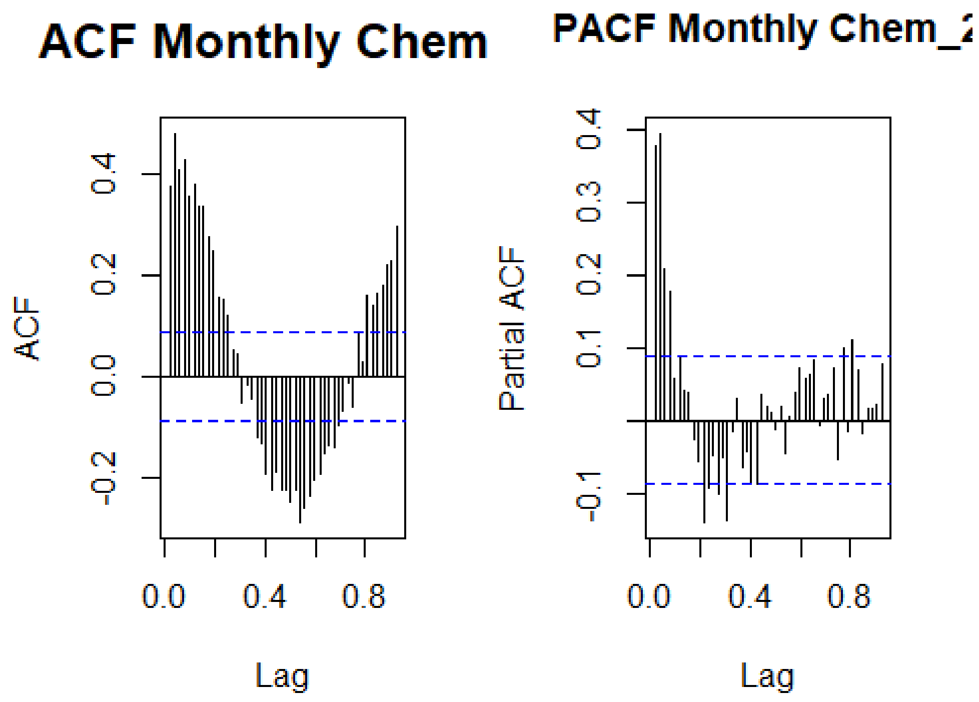
* The ACF plot exhibits sign of declining trend in the time series dataset. There is very little possibility of a seasonal pattern, and a considerable number of lag values exceed the 95% confidence interval.Conversely, when examining the PACF plot, it becomes evident that the first two lags hold significant importance, implying that the series lacks stationarity .

```
par(mfrow=c(1,2))
acf(chem_1, lag.max = 48, main = "ACF Monthly Chem_1", cex.main = 1.5)
pacf(chem_1, lag.max = 48, main = "PACF Monthly Chem_1", cex.main = 1.5)
```



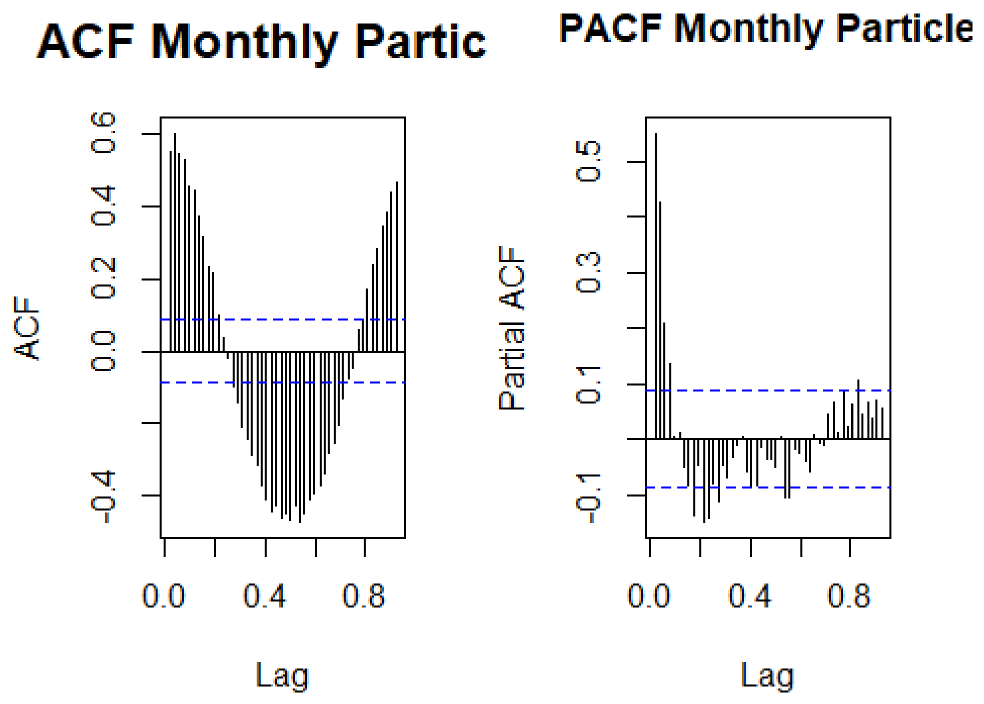
* The ACF plot shows a gradual decline, indicating a noticeable decreasing trend in the time series data. There is a possibility of little seasonal pattern in this data. On the other hand, when you look at the PACF plot, it’s clear that the first few lags are highly significant, suggesting that the series is not stationary.

```
par(mfrow=c(1,2))
acf(chem_2, lag.max = 48, main = " ACF Monthly Chem_2", cex.main = 1.5)
pacf(chem_2, lag.max = 48, main = "PACF Monthly Chem_2", cex.main = 1.5)
```



* The ACF plot shows signs of decreasing trend in the time series data. There seems to be no possibility of seasonal pattern in this data. On the other hand, when looking at the PACF plot, it's clear that the first few lags are highly significant, suggesting that the series is not stationary.

```
par(mfrow=c(1,2))
acf(particle, lag.max = 48, main = "ACF Monthly Particle", cex.main = 1.5)
pacf(particle, lag.max = 48, main = "PACF Monthly Particle", cex.main = 1.5)
```



* The ACF plot's gradual decline signifies a pronounced decreasing trend within the time series data. There is a possibility of a seasonal pattern in this dataset. Conversely, an inspection of the PACF plot reveals that the initial lags hold substantial significance, implying the absence of stationarity.

```
k <- ar(mort_ts)$order
adf.test(mort_ts, k = k)

## Warning in adf.test(mort_ts, k = k): p-value smaller than printed p-value

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

* In this situation, with a lag order of 5 and a p-value of 0.01, we reject the null hypothesis. This essentially means that all the time series are stationary. In other words,it has no time dependent structure and does have constant variance over time. This demonstrates the time-dependent patterns and its variance is uniform over time.

```
k <- ar(temp_ts)$order
adf.test(temp_ts, k = k)

## Warning in adf.test(temp_ts, k = k): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
```

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

 * In this situation, with a lag order of 22 and a p-value of 0.01, we reject the null hypothesis for all the time series data. This essentially means that all the time series are stationary. In other words,it has no time dependent structure and does have constant variance over time. This demonstrates the time-dependent patterns and its variance is uniform over time.

```
k <- ar(chem_1)$order
adf.test(chem_1, k = k)

## Warning in adf.test(chem_1, k = k): p-value smaller than printed p-value

## 
##  Augmented Dickey-Fuller Test
## 
## data:  chem_1
## Dickey-Fuller = -5.3362, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

 * In this situation, with a lag order of 8 and a p-value of 0.01, we reject the null hypothesis for all the time series data. This essentially means that all the time series are stationary. In other words,it has no time dependent structure and does have constant variance over time. This demonstrates the time-dependent patterns and its variance is uniform over time.

```
k <- ar(chem_2)$order
adf.test(chem_2, k = k)

## Warning in adf.test(chem_2, k = k): p-value smaller than printed p-value

## 
##  Augmented Dickey-Fuller Test
## 
## data:  chem_2
## Dickey-Fuller = -6.5194, Lag order = 16, p-value = 0.01
## alternative hypothesis: stationary
```

- In this situation, with a lag order of 16 and a p-value of 0.01, we reject the null hypothesis for all the time series data. This essentially means that all the time series are stationary. In other words,it has no time dependent structure and does have constant variance over time. This demonstrates the time-dependent patterns and its variance is uniform over time.

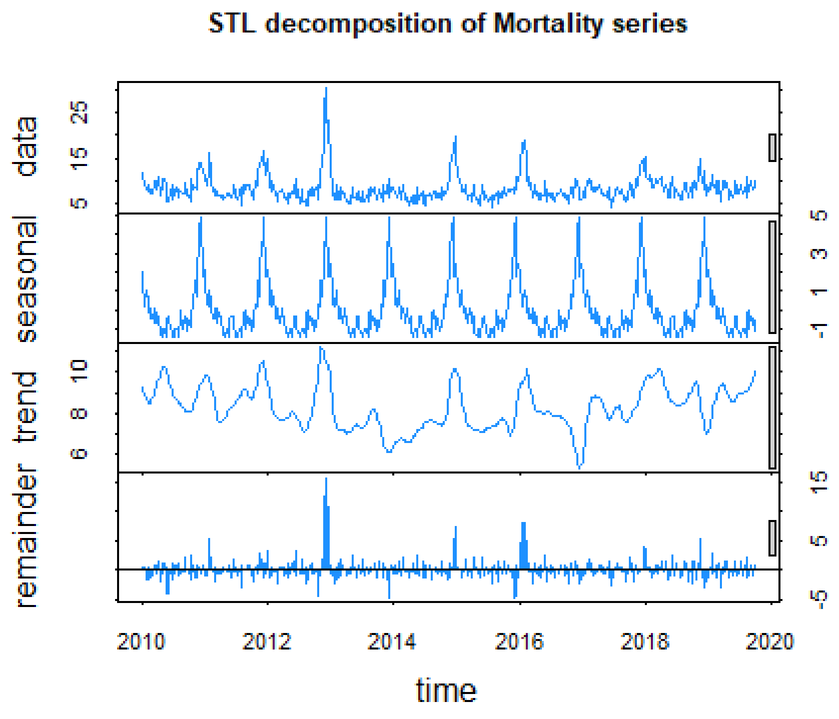
```
k <- ar(particle)$order
adf.test(particle, k = k)

## Warning in adf.test(particle, k = k): p-value smaller than printed p-value

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

 * In this situation, with a lag order of 14 and a p-value of 0.01, we reject the null hypothesis for all the time series data. This essentially means that all the time series are stationary. In other words,it has no time dependent structure and does have constant variance over time. This demonstrates the time-dependent patterns and its variance is uniform over time.

STL decomposition

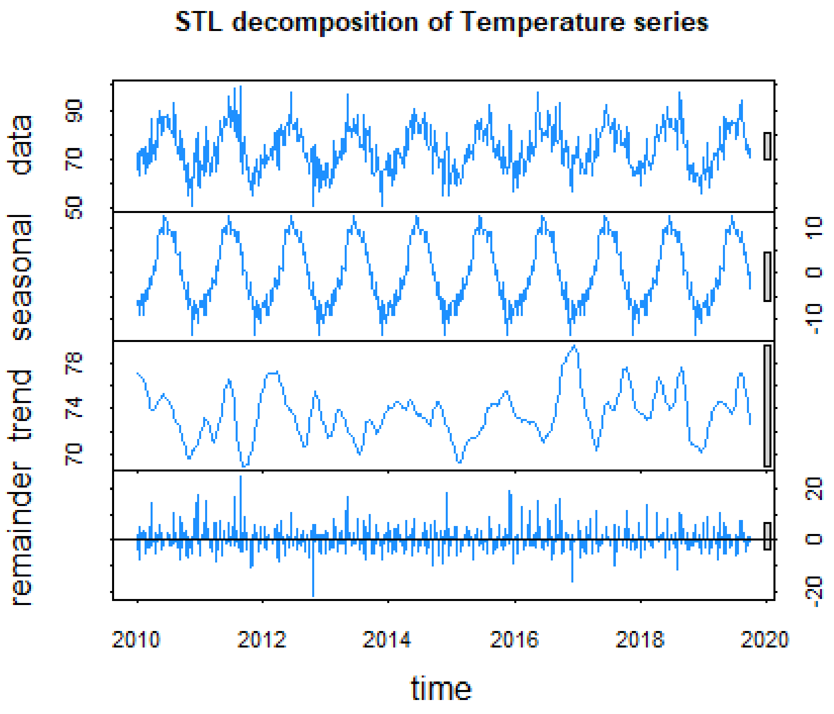
- STL decomposition, developed by Hyndman and Athanasopoulos in 2014, outperforms classical decomposition methods in several ways.It is good at handling different types of repeating patterns, not just monthly or quarterly ones. One key benefit is its adaptability to diverse seasonality patterns. In contrast to traditional methods that often work best with monthly or quarterly dat, the scenario is refraining from introducing any differencing because the time series data is already stationary.

```
mortality_stl_dc <- stl(mort_ts, t.window=15, s.window ="periodic", robust=TRUE)
plot(mortality_stl_dc, main = "STL decomposition of Mortality series", col="#1E90FF")
```



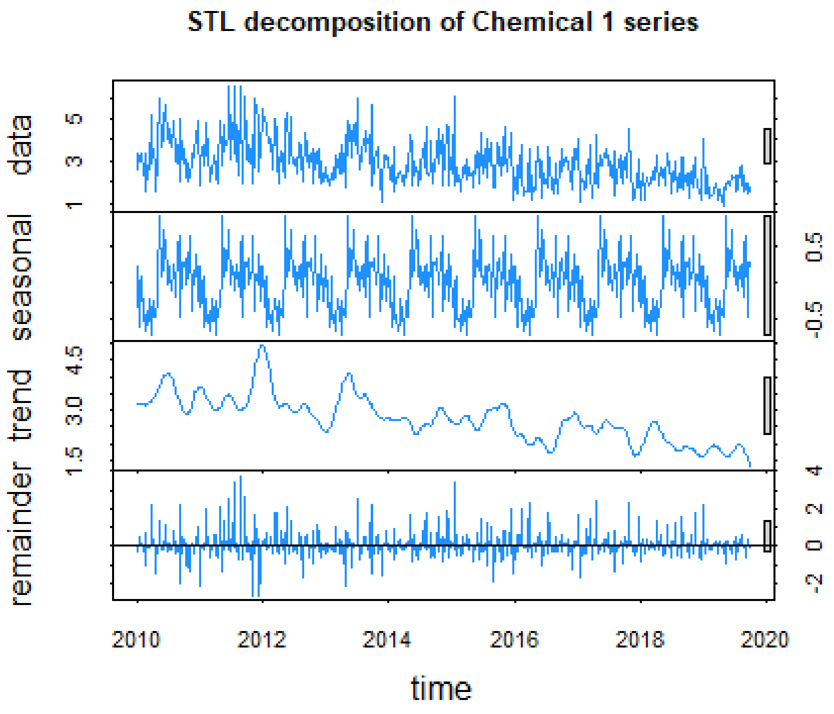
 * Analyzing the STL decomposition, it becomes evident that there is no discernible trend. Additionally, there is no indication of seasonality in the data set. However, it does appear that there have been interventions or anomalies occurring during certain weeks in the years 2013, 2015, and 2016.

```
temp_stl_dc <- stl(temp_ts, t.window=15, s.window ="periodic", robust=TRUE)
plot(temp_stl_dc, main = "STL decomposition of Temperature series", col="#1E90FF")
```



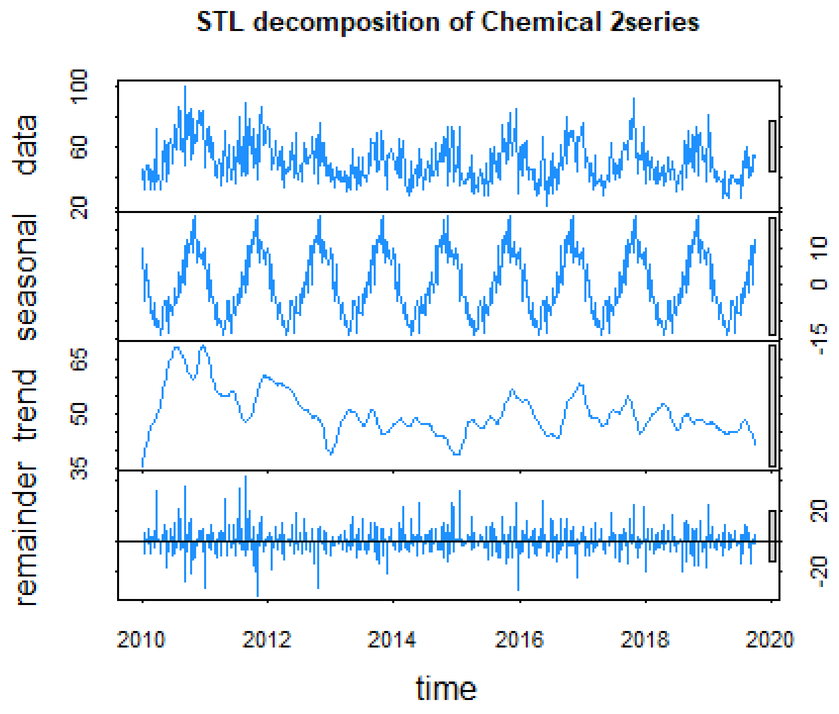
* Upon examining the STL decomposition, it’s clear that there is no noticeable trend. Moreover, there is a strong signal of seasonality in the data set. Nevertheless, it does appear that there have been instances of interventions or unusual occurrences during specific weeks in the years 2011, 2013, 2016, and 2017.

```
chem_1_stl_dc <- stl(chem_1, t.window=15, s.window ="periodic", robust=TRUE)
plot(chem_1_stl_dc, main = "STL decomposition of Chemical 1 series", col="#1E90FF")
```



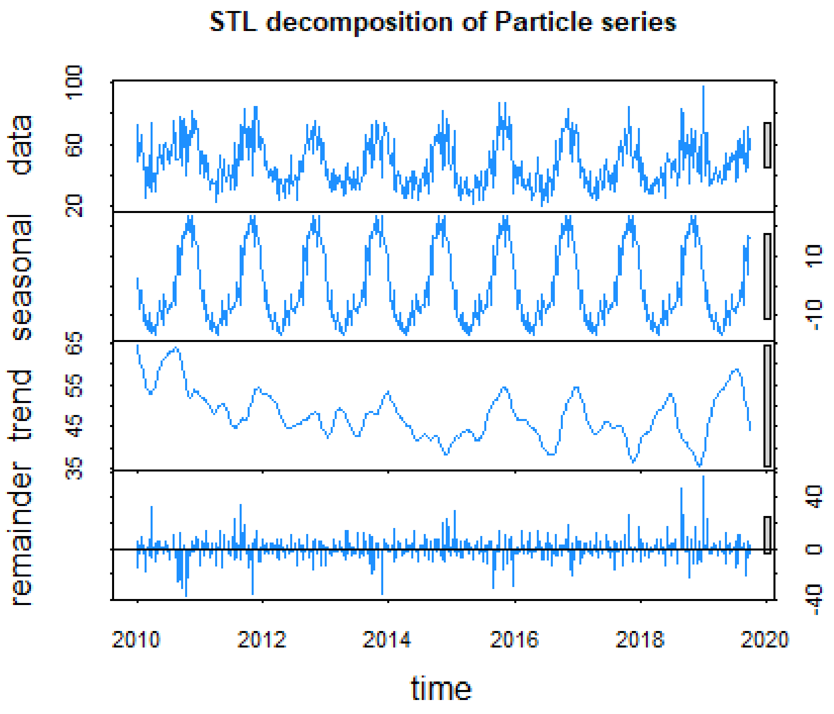
* After analyzing the STL decomposition, it’s evident that there is a discernible trend, aligning with our observations in the ACF plot. Additionally, there is a relatively weak signal of seasonality in the dataset. However, it does seem that there have been instances of interventions or unusual events occurring over various weeks in multiple years.

```
chem_2_stl_dc <- stl(chem_2, t.window=15, s.window ="periodic", robust=TRUE)
plot(chem_2_stl_dc, main = "STL decomposition of Chemical 2series", col="#1E90FF")
```



* After analyzing the STL decomposition, it’s evident that there is no discernible trend. Additionally, there is no sign of seasonality in the dataset. However, it does seem that there have been instances of interventions or unusual events occurring over various weeks in multiple years.

```
particle_stl_dc <- stl(particle, t.window=15, s.window ="periodic", robust=TRUE)
plot(particle_stl_dc, main = "STL decomposition of Particle series", col="#1E90FF")
```



* Upon reviewing the STL decomposition, it’s clear that there isn’t a noticeable trend. Furthermore, there are indications of seasonality in the data set, although not very strong. Nonetheless, there are instances of interventions or unusual events happening during different weeks across multiple years.

Modelling utlizing distributed Lag Models

Finite DLM

```
finiteDLMauto(x=as.vector(chem_1 + chem_2 + temp_ts + particle), y= as.vector(mort_ts), 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(chem_1 + chem_2 + temp_ts + particle), y= as.vector(mort_ts),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
```

```
finiteDLMauto(x=as.vector(chem_1 + chem_2 + temp_ts + particle), y= as.vector(mort_ts),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
```

* Given that the lag length of 10 has been identified as the optimal choice based on all two criteria—AIC, BIC, and MASE, the next step is to proceed with fitting a finite Dynamic Linear Model (DLM).

```
model_1_particle <- dlm(x=as.vector(particle), y=as.vector(mort_ts), q=10)

model_2_chem_1 <- dlm(x=as.vector(chem_1), y=as.vector(mort_ts), q=10)

model_3_chem_2 <- dlm(x=as.vector(chem_2), y=as.vector(mort_ts), q=10)

model_4_temp <- dlm(x=as.vector(temp_ts), y=as.vector(mort_ts), q=10)

model_5 <- dlm(x=as.vector(chem_1 + chem_2 + temp_ts + particle), y=as.vector(mort_ts), q=10)

model_6 <- dlm(x=as.vector(chem_1 + chem_2 + temp_ts), y=as.vector(mort_ts), q=10)

model_7 <- dlm(x=as.vector(chem_1 + chem_2 + particle), y=as.vector(mort_ts), q=10)

model_8<- dlm(x=as.vector(chem_1 + chem_2), y=as.vector(mort_ts), q=10)

model_9 <- dlm(x=as.vector(chem_1 + temp_ts), y=as.vector(mort_ts), q=10)

model_10 <- dlm(x=as.vector(chem_1 + temp_ts + particle), y=as.vector(mort_ts), q=10)

model_11 <- dlm(x=as.vector(chem_2 + temp_ts), y=as.vector(mort_ts), q=10)

model_12 <- dlm(x=as.vector(chem_2 + temp_ts + particle), y=as.vector(mort_ts), q=10)
```

```
model_13 <- dlm(x=as.vector(chem_1 + particle), y=as.vector(mort_ts), q=10)

model_14 <- dlm(x=as.vector(chem_2 + particle), y=as.vector(mort_ts), q=10)

model_15 <- dlm(x=as.vector(temp_ts + particle), y=as.vector(mort_ts), q=10)

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

sort.score(AIC(model_1_particle$model,model_2_chem_1$model,
model_3_chem_2$model,model_4_temp$model, model_5$model, model_6$model, model_7$model,
model_8$model, model_9$model,
model_10$model,model_11$model,model_12$model,model_13$model,
model_14$model,model_15$model), score = "aic")

##                        df      AIC
## model_1_particle$model 13 2318.164
## model_13$model         13 2323.363
## model_14$model         13 2341.546
## model_7$model          13 2344.918
## model_4_temp$model     13 2380.059
## model_12$model         13 2380.687
## model_5$model          13 2383.868
## model_9$model          13 2384.980
## model_3_chem_2$model   13 2393.253
## model_15$model         13 2397.244
## model_8$model          13 2398.708
## model_10$model         13 2401.098
## model_11$model         13 2455.059
## model_6$model          13 2456.223
## model_2_chem_1$model   13 2483.558

sort.score(BIC(model_1_particle$model,model_2_chem_1$model,
model_3_chem_2$model,model_4_temp$model, model_5$model, model_6$model, model_7$model,
model_8$model, model_9$model,
model_10$model,model_11$model,model_12$model,model_13$model,
model_14$model,model_15$model), score = "bic")

##                        df      BIC
## model_1_particle$model 13 2372.902
## model_13$model         13 2378.101
## model_14$model         13 2396.284
## model_7$model          13 2399.656
## model_4_temp$model     13 2434.796
## model_12$model         13 2435.425
## model_5$model          13 2438.606
## model_9$model          13 2439.717
## model_3_chem_2$model   13 2447.991
## model_15$model         13 2451.982
## model_8$model          13 2453.446
## model_10$model         13 2455.836
## model_11$model         13 2509.797
## model_6$model          13 2510.961
## model_2_chem_1$model   13 2538.296
```

```
sort.score(MASE(model_1_particle$model,model_2_chem_1$model,
model_3_chem_2$model,model_4_temp$model, model_5$model, model_6$model, model_7$model,
model_8$model, model_9$model,
model_10$model,model_11$model,model_12$model,model_13$model, model_14$model), score =
"mase")
```

```
##                          n     MASE
## model_1_particle$model 498 1.087711
## model_13$model         498 1.097456
## model_14$model         498 1.101983
## model_7$model          498 1.107963
## model_12$model         498 1.148070
## model_3_chem_2$model   498 1.150336
## model_5$model          498 1.154767
## model_8$model          498 1.163328
## model_4_temp$model     498 1.169444
## model_9$model          498 1.170722
## model_10$model         498 1.180025
## model_11$model         498 1.235505
## model_6$model          498 1.238762
## model_2_chem_1$model   498 1.294917
```

 * Considering the result for AIC, BIC and MASE model model_1_particle is the best model. The model with model_1_particle as the predictor stands out as the most suitable choice according to AIC, BIC and MASE values. Therefore, this model will undergo further analysis for diagnostic checking.

```
model_1_particle <- dlm(x=as.vector(particle), y=as.vector(mort_ts), q=10)
summary(model_1_particle)
```

```
##
## Call:
## lm(formula = model.formula, data = design)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7002 -1.3059 -0.1555  0.9361 20.0489
##
```

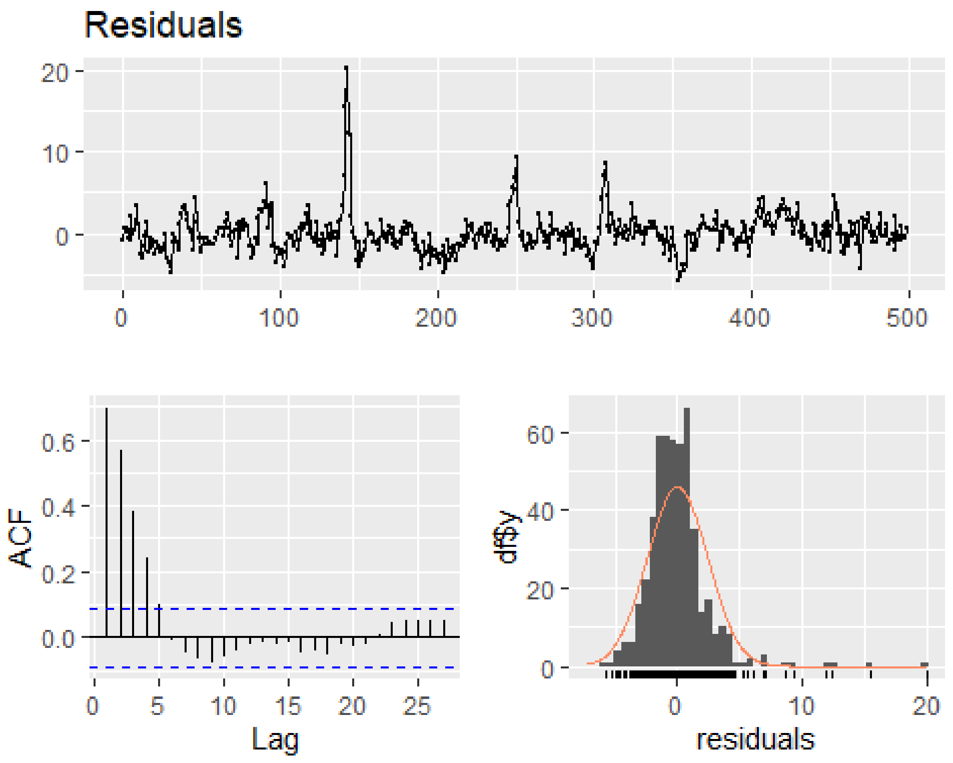
```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  2.0116427  0.4927144   4.083  5.2e-05 ***
## x.t          0.0105446  0.0101416   1.040  0.29898    
## x.1          0.0002472  0.0102817   0.024  0.98083    
## x.2          0.0012723  0.0106010   0.120  0.90452    
## 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
```

```
vif(model_1_particle$model)>10
```

```
##   x.t   x.1   x.2   x.3   x.4   x.5   x.6   x.7   x.8   x.9  x.10 
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

* The model_1_particle fit is inadequate, as most of the lag variables are not statistically significant. Additionally, the adjusted R-squared value is low, indicating that the model can only explain 28.19% of the variation in the dependent variable. The AIC, BIC, and MASE values for the model are 2318.164, 2372.902, and 1.087711, respectively. However, despite having numerous lag variables, the gold copper model shows overall significance at the 5% level based on the F-Test. Furthermore, examining the VIF (Variance Inflation Factor) values for the lag variables within model1_finite, they are all well below the threshold of 10, suggesting that multicollinearity is not a significant issue in the model. The Residual Standard Error (RSE) is 2.446, and the range of residuals spans from a minimum value of -5.7002 to a maximum value of 20.0489. A smaller RSE indicates a more precise fit as the data points are closer to the models prediction.

```
checkresiduals(model_1_particle$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
```

 * Taking into account the results of the Breusch-Godfrey test, the p-value falls below the 5% significance level, leading to the rejection of the null hypothesis. This implies the existence of serial correlation within the residuals. Further examination of the residuals resulting from the finite dynamic linear model (DLM) fit reveals that they do not exhibit a random distribution. Specifically, the autocorrelation function (ACF) demonstrates a declining pattern, confirming the presence of serial correlation in the residuals, which aligns with the findings of the Breusch-Godfrey test. Additionally, the histograms indicate a departure from normal distribution. In summary, the overall conclusion is that the model is not a suitable fit for the data.

Polynomial DLM

```
finiteDLMauto(x = as.vector(chem_1 + chem_2 + temp_ts + particle), y = as.vector(mort_ts), 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 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(chem_1 + chem_2 + temp_ts + particle), y = as.vector(mort_ts), 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 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
```

 * Considering the findings presented in the previous code, a lag length of 10 will be employed, and the polynomial order will be modified to 2.

```
model_poly_1 <- polyDlm(x=as.vector(chem_1 + chem_2 + temp_ts + particle), y=as.vector(mort_ts), 
q=10, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##          Estimate Std. Error t value  P(>|t|)
## beta.0  -0.004140    0.00326  -1.270 2.05e-01
## beta.1  -0.002820    0.00192  -1.470 1.43e-01
## beta.2  -0.001310    0.00125  -1.050 2.96e-01
## beta.3   0.000394    0.00134   0.295 7.68e-01
## beta.4   0.002290    0.00162   1.420 1.57e-01
## beta.5   0.004380    0.00173   2.520 1.19e-02
## beta.6   0.006660    0.00162   4.120 4.50e-05
## beta.7   0.009140    0.00134   6.830 2.49e-11
## beta.8   0.011800    0.00125   9.430 1.61e-19
## beta.9   0.014700    0.00192   7.630 1.22e-13
## beta.10  0.017700    0.00326   5.440 8.40e-08
```

```
model_poly_2 <- polyDlm(x=as.vector(particle), y=as.vector(mort_ts), q=10, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value  P(>|t|)
## beta.0   0.00565    0.00697   0.811 4.18e-01
## beta.1   0.00370    0.00368   1.010 3.15e-01
## beta.2   0.00283    0.00201   1.410 1.59e-01
## beta.3   0.00306    0.00261   1.170 2.42e-01
## beta.4   0.00438    0.00351   1.250 2.13e-01
## beta.5   0.00679    0.00385   1.760 7.84e-02
## beta.6   0.01030    0.00352   2.920 3.61e-03
## beta.7   0.01490    0.00263   5.660 2.52e-08
## beta.8   0.02060    0.00201  10.200 2.26e-22
## beta.9   0.02740    0.00364   7.510 2.89e-13
## beta.10  0.03520    0.00692   5.090 5.07e-07
```

```
model_poly_3 <- polyDlm(x=as.vector(chem_1), y=as.vector(mort_ts), q=10, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value P(>|t|)
## beta.0  -0.10700     0.1040 -1.0200 0.30700
## beta.1  -0.11300     0.0639 -1.7700 0.07810
## beta.2  -0.10900     0.0432 -2.5300 0.01180
## beta.3  -0.09570     0.0439 -2.1800 0.02980
## beta.4  -0.07260     0.0513 -1.4200 0.15700
## beta.5  -0.03980     0.0545 -0.7290 0.46600
## beta.6   0.00276     0.0514  0.0537 0.95700
## beta.7   0.05500     0.0442  1.2400 0.21400
## beta.8   0.11700     0.0436  2.6800 0.00757
## beta.9   0.18900     0.0643  2.9300 0.00350
## beta.10  0.27000     0.1050  2.5800 0.01020
```

```
model_poly_4 <- polyDlm(x=as.vector(chem_2), y=as.vector(mort_ts), q=10, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##          Estimate Std. Error t value  P(>|t|)
## beta.0  -0.002560    0.00770  -0.332 7.40e-01
## beta.1   0.000702    0.00445   0.158 8.75e-01
## beta.2   0.003820    0.00284   1.350 1.79e-01
## beta.3   0.006790    0.00312   2.180 3.00e-02
## beta.4   0.009630    0.00385   2.500 1.27e-02
## beta.5   0.012300    0.00414   2.970 3.08e-03
## beta.6   0.014900    0.00384   3.870 1.24e-04
## beta.7   0.017300    0.00311   5.550 4.65e-08
## beta.8   0.019500    0.00282   6.930 1.37e-11
## beta.9   0.021700    0.00444   4.880 1.44e-06
## beta.10  0.023700    0.00769   3.080 2.21e-03
```

```
model_poly_5 <- polyDlm(x=as.vector(temp_ts), y=as.vector(mort_ts), q=10, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value  P(>|t|)
## beta.0  -0.03390    0.01200  -2.820 5.00e-03
## beta.1  -0.03370    0.00639  -5.280 1.93e-07
## beta.2  -0.03230    0.00360  -8.980 5.81e-18
## beta.3  -0.02970    0.00462  -6.430 3.13e-10
## beta.4  -0.02580    0.00613  -4.210 3.03e-05
## beta.5  -0.02070    0.00668  -3.100 2.06e-03
## beta.6  -0.01440    0.00609  -2.360 1.87e-02
## beta.7  -0.00681    0.00456  -1.490 1.36e-01
## beta.8   0.00199    0.00360   0.553 5.81e-01
## beta.9   0.01200    0.00652   1.840 6.58e-02
## beta.10  0.02330    0.01220   1.900 5.75e-02
```

```
model_poly_6 <- polyDlm(x=as.vector(chem_1 + chem_2 + temp_ts), y=as.vector(mort_ts), q=10, k =
2)

## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value  P(>|t|)
## beta.0  -0.01340    0.00565  -2.360 1.85e-02
## beta.1  -0.01110    0.00351  -3.160 1.65e-03
## beta.2  -0.00850    0.00242  -3.510 4.96e-04
## beta.3  -0.00557    0.00245  -2.270 2.36e-02
## beta.4  -0.00231    0.00283  -0.818 4.14e-01
## beta.5   0.00129    0.00299   0.431 6.67e-01
## beta.6   0.00522    0.00281   1.850 6.42e-02
## beta.7   0.00948    0.00243   3.900 1.09e-04
## beta.8   0.01410    0.00240   5.860 8.73e-09
## beta.9   0.01900    0.00351   5.420 9.38e-08
## beta.10  0.02430    0.00567   4.280 2.23e-05

model_poly_7 <- polyDlm(x=as.vector(chem_1 + chem_2 + particle), y=as.vector(mort_ts), q=10, k =
2)

## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value  P(>|t|)
## beta.0  0.000198    0.00368  0.0538 9.57e-01
## beta.1  0.000745    0.00203  0.3670 7.14e-01
## beta.2  0.001520    0.00121  1.2600 2.08e-01
## beta.3  0.002530    0.00144  1.7600 7.88e-02
## beta.4  0.003770    0.00185  2.0300 4.25e-02
## beta.5  0.005240    0.00201  2.6000 9.63e-03
## beta.6  0.006940    0.00185  3.7400 2.06e-04
## beta.7  0.008870    0.00144  6.1600 1.52e-09
## beta.8  0.011000    0.00121  9.1400 1.69e-18
## beta.9  0.013400    0.00202  6.6300 8.92e-11
## beta.10 0.016000    0.00367  4.3700 1.55e-05

model_poly_8 <- polyDlm(x=as.vector(chem_1 + chem_2), y=as.vector(mort_ts), q=10, k = 2)

## Estimates and t-tests for beta coefficients:
##          Estimate Std. Error t value  P(>|t|)
## beta.0  -0.002960    0.00731 -0.4040 6.86e-01
## beta.1   0.000128    0.00427  0.0299 9.76e-01
## beta.2   0.003110    0.00274  1.1300 2.58e-01
## beta.3   0.005980    0.00298  2.0000 4.57e-02
## beta.4   0.008740    0.00365  2.3900 1.70e-02
## beta.5   0.011400    0.00392  2.9100 3.83e-03
## beta.6   0.014000    0.00365  3.8300 1.47e-04
## beta.7   0.016400    0.00298  5.5100 5.85e-08
## beta.8   0.018700    0.00273  6.8600 2.14e-11
## beta.9   0.021000    0.00426  4.9200 1.16e-06
## beta.10  0.023100    0.00731  3.1600 1.68e-03

model_poly_9 <- polyDlm(x=as.vector(chem_1 + temp_ts), y=as.vector(mort_ts), q=10, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value  P(>|t|)
## beta.0  -0.03220    0.01110  -2.910 3.83e-03
## beta.1  -0.03220    0.00598  -5.380 1.18e-07
## beta.2  -0.03090    0.00347  -8.920 9.21e-18
## beta.3  -0.02850    0.00429  -6.630 8.79e-11
## beta.4  -0.02480    0.00562  -4.420 1.24e-05
## beta.5  -0.01990    0.00611  -3.260 1.20e-03
## beta.6  -0.01380    0.00558  -2.470 1.38e-02
## beta.7  -0.00647    0.00424  -1.530 1.27e-01
## beta.8   0.00207    0.00347   0.597 5.51e-01
## beta.9   0.01180    0.00609   1.940 5.27e-02
## beta.10  0.02280    0.01120   2.030 4.31e-02
```

```
model_poly_10 <- polyDlm(x=as.vector(chem_1 + temp_ts + particle), y=as.vector(mort_ts), q=10, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value  P(>|t|)
## beta.0  -0.00989    0.00530  -1.870 6.27e-02
## beta.1  -0.00816    0.00313  -2.610 9.30e-03
## beta.2  -0.00588    0.00203  -2.900 3.89e-03
## beta.3  -0.00305    0.00216  -1.410 1.60e-01
## beta.4   0.00033    0.00262   0.126 9.00e-01
## beta.5   0.00426    0.00282   1.510 1.32e-01
## beta.6   0.00873    0.00263   3.320 9.66e-04
## beta.7   0.01380    0.00218   6.320 5.88e-10
## beta.8   0.01930    0.00204   9.470 1.24e-19
## beta.9   0.02540    0.00313   8.120 3.86e-15
## beta.10  0.03210    0.00531   6.050 2.85e-09
```

```
model_poly_11 <- polyDlm(x=as.vector(chem_2 + temp_ts), y=as.vector(mort_ts), q=10, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value  P(>|t|)
## beta.0  -0.01420    0.00590  -2.400 1.67e-02
## beta.1  -0.01170    0.00366  -3.200 1.46e-03
## beta.2  -0.00890    0.00253  -3.520 4.66e-04
## beta.3  -0.00577    0.00256  -2.250 2.46e-02
## beta.4  -0.00230    0.00295  -0.779 4.36e-01
## beta.5   0.00151    0.00312   0.484 6.29e-01
## beta.6   0.00565    0.00293   1.920 5.48e-02
## beta.7   0.01010    0.00253   3.990 7.47e-05
## beta.8   0.01490    0.00251   5.960 4.84e-09
## beta.9   0.02010    0.00366   5.490 6.56e-08
## beta.10  0.02560    0.00592   4.320 1.90e-05
```

```
model_poly_12 <- polyDlm(x=as.vector(chem_1 + particle), y=as.vector(mort_ts), q=10, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value  P(>|t|)
## beta.0   0.00452    0.00661   0.684 4.94e-01
## beta.1   0.00307    0.00353   0.871 3.84e-01
## beta.2   0.00259    0.00197   1.320 1.89e-01
## beta.3   0.00308    0.00249   1.240 2.16e-01
## beta.4   0.00454    0.00331   1.370 1.71e-01
## beta.5   0.00697    0.00363   1.920 5.53e-02
## beta.6   0.01040    0.00333   3.120 1.93e-03
## beta.7   0.01470    0.00251   5.870 8.09e-09
## beta.8   0.02010    0.00198  10.200 4.03e-22
## beta.9   0.02640    0.00350   7.540 2.23e-13
## beta.10  0.03370    0.00656   5.130 4.12e-07
```

```
model_poly_13 <- polyDlm(x=as.vector(chem_2 + particle), y=as.vector(mort_ts), q=10, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value  P(>|t|)
## beta.0  0.000408    0.00379   0.108 9.14e-01
## beta.1  0.000897    0.00208   0.432 6.66e-01
## beta.2  0.001630    0.00122   1.340 1.82e-01
## beta.3  0.002610    0.00147   1.780 7.62e-02
## beta.4  0.003840    0.00191   2.010 4.46e-02
## beta.5  0.005320    0.00208   2.560 1.08e-02
## beta.6  0.007040    0.00191   3.690 2.54e-04
## beta.7  0.009020    0.00148   6.110 2.02e-09
## beta.8  0.011200    0.00122   9.190 1.11e-18
## beta.9  0.013700    0.00207   6.620 9.72e-11
## beta.10 0.016400    0.00378   4.340 1.75e-05
```

```
model_poly_14 <- polyDlm(x=as.vector(temp_ts + particle), y=as.vector(mort_ts), q=10, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##          Estimate Std. Error t value  P(>|t|)
## beta.0  -0.010400    0.00552  -1.880 6.07e-02
## beta.1  -0.008560    0.00323  -2.650 8.42e-03
## beta.2  -0.006170    0.00208  -2.960 3.25e-03
## beta.3  -0.003200    0.00224  -1.430 1.54e-01
## beta.4   0.000352    0.00273   0.129 8.97e-01
## beta.5   0.004480    0.00293   1.530 1.27e-01
## beta.6   0.009190    0.00273   3.360 8.43e-04
## beta.7   0.014500    0.00225   6.420 3.23e-10
## beta.8   0.020300    0.00210   9.670 2.42e-20
## beta.9   0.026800    0.00325   8.250 1.53e-15
## beta.10  0.033800    0.00552   6.120 1.91e-09
```

```
sort.score(MASE(model_poly_1$model,model_poly_2$model,
model_poly_3$model,model_poly_4$model, model_poly_5$model, model_poly_6$model,
model_poly_7$model, model_poly_8$model, model_poly_9$model,
model_poly_10$model,model_poly_11$model,model_poly_12$model,model_poly_13$model,
model_poly_14$model), score = "mase")
```

```
##                       n     MASE
## model_poly_2$model  498 1.091676
## model_poly_12$model 498 1.100952
## model_poly_13$model 498 1.105860
## model_poly_7$model  498 1.112082
## model_poly_4$model  498 1.157062
## model_poly_1$model  498 1.160041
## model_poly_8$model  498 1.170545
## model_poly_14$model 498 1.180714
## model_poly_5$model  498 1.186847
## model_poly_9$model  498 1.187319
## model_poly_10$model 498 1.187372
## model_poly_11$model 498 1.246725
## model_poly_6$model  498 1.249934
## model_poly_3$model  498 1.300309
```

```
sort.score(AIC(model_poly_1$model,model_poly_2$model,
model_poly_3$model,model_poly_4$model, model_poly_5$model, model_poly_6$model,
model_poly_7$model, model_poly_8$model, model_poly_9$model,
model_poly_10$model,model_poly_11$model,model_poly_12$model,model_poly_13$model,
model_poly_14$model), score = "aic")
```

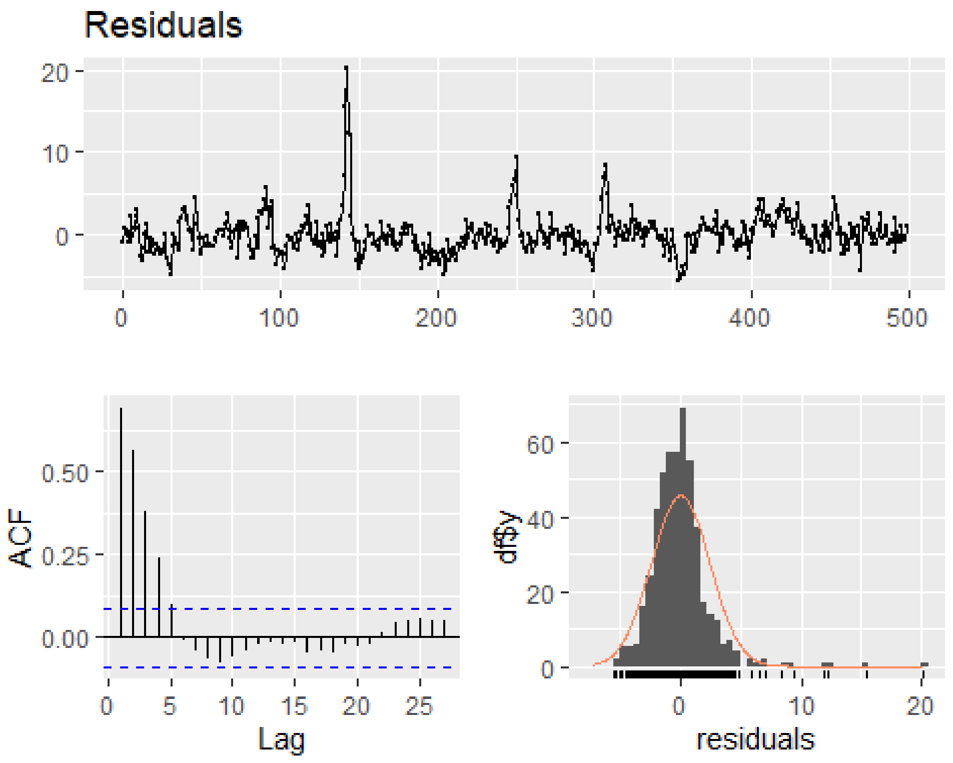
```
##                     df      AIC
## model_poly_2$model   5 2303.729
## model_poly_12$model  5 2308.926
## model_poly_13$model  5 2327.700
## model_poly_7$model   5 2331.071
## model_poly_5$model   5 2369.024
## model_poly_1$model   5 2370.609
## model_poly_9$model   5 2373.583
## model_poly_4$model   5 2379.735
## model_poly_14$model  5 2383.690
## model_poly_8$model   5 2385.176
## model_poly_10$model  5 2387.510
## model_poly_11$model  5 2442.407
## model_poly_6$model   5 2443.494
## model_poly_3$model   5 2469.016
```

```
sort.score(BIC(model_poly_1$model,model_poly_2$model,
model_poly_3$model,model_poly_4$model, model_poly_5$model, model_poly_6$model,
model_poly_7$model, model_poly_8$model, model_poly_9$model,
model_poly_10$model,model_poly_11$model,model_poly_12$model,model_poly_13$model,
model_poly_14$model), score = "bic")
```

```
##                     df      BIC
## model_poly_2$model   5 2324.782
## model_poly_12$model  5 2329.979
## model_poly_13$model  5 2348.753
## model_poly_7$model   5 2352.124
## model_poly_5$model   5 2390.077
## model_poly_1$model   5 2391.662
## model_poly_9$model   5 2394.636
## model_poly_4$model   5 2400.788
## model_poly_14$model  5 2404.743
## model_poly_8$model   5 2406.229
## model_poly_10$model  5 2408.563
## model_poly_11$model  5 2463.460
## model_poly_6$model   5 2464.547
## model_poly_3$model   5 2490.069
```

- Taking into account the AIC, BIC, and MASE values, the “model_poly_2” emerges as the preferred predictive model. Now, let’s delve into a comprehensive analysis of this specific model.

```
summary(model_poly_2$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

vif(model_poly_2$model) > 10

## z.t0 z.t1 z.t2
## TRUE TRUE TRUE

checkresiduals(model_poly_2$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
```

* The model "model_poly_2" is poorly fitted and almost all lag variables are statistically insignificant, except for "z.t0." The lower adjusted R-squared value indicates that it can only explain 29.13% of the variation in the dependent variable. Despite having lag variables, the model demonstrates statistical significance at the 5% level according to the F-Test for overall model significance.

- As seen in the preceding data, the VIF values exceed 10 for every lag variable within "model_poly_2." This indicates a significant issue of multi collinearity in the model.

- Based on the results of the Breusch-Godfrey test, the p-value is below the 5% significance level, leading to the rejection of the null hypothesis, signifying the presence of serial correlation in the residuals. Examination of residuals from the polynomial DLM reveals a lack of randomness. Notably, the autocorrelation function (ACF) displays a decreasing pattern, confirming the presence of serial correlation in the residuals, which aligns with the results of the Breusch-Godfrey test. In summary, due to the presence of multi collinearity, the fitted polynomial DLM is inadequate.

Koyck Transformation DLM

```
model_Koyock_1 <- koyckDlm(x=as.vector(particle), y = as.vector(mort_ts))
model_Koyock_2 <- koyckDlm(x=as.vector(chem_1), y = as.vector(mort_ts))
model_Koyock_3 <- koyckDlm(x=as.vector(chem_2), y = as.vector(mort_ts))
model_Koyock_4 <- koyckDlm(x=as.vector(temp_ts), y = as.vector(mort_ts))
model_Koyock_5 <- koyckDlm(x=as.vector(chem_1 + chem_2 + temp_ts + particle), y =
as.vector(mort_ts))
model_Koyock_6 <- koyckDlm(x=as.vector(chem_1 + chem_2 + temp_ts), y = as.vector(mort_ts))
model_Koyock_7 <- koyckDlm(x=as.vector(chem_1 + chem_2 + particle), y = as.vector(mort_ts))
model_Koyock_8 <- koyckDlm(x=as.vector(chem_1 + chem_2), y = as.vector(mort_ts))
model_Koyock_9 <- koyckDlm(x=as.vector(chem_1 + temp_ts), y = as.vector(mort_ts))
model_Koyock_10 <- koyckDlm(x=as.vector(chem_1 + temp_ts + particle), y = as.vector(mort_ts))
model_Koyock_11 <- koyckDlm(x=as.vector(chem_2 + temp_ts), y = as.vector(mort_ts))
model_Koyock_12<- koyckDlm(x=as.vector(chem_2 + temp_ts + particle), y = as.vector(mort_ts))
model_Koyock_13 <- koyckDlm(x=as.vector(chem_1 + particle), y = as.vector(mort_ts))
model_Koyock_14 <- koyckDlm(x=as.vector(chem_2 + particle), y = as.vector(mort_ts))
model_Koyock_15 <- koyckDlm(x=as.vector(temp_ts + particle), y = as.vector(mort_ts))
```

```
MASE(model_Koyock_1, model_Koyock_2,model_Koyock_3, model_Koyock_4,model_Koyock_5,
model_Koyock_6, model_Koyock_7, model_Koyock_8, model_Koyock_9, model_Koyock_10,
model_Koyock_11, model_Koyock_12, model_Koyock_13, model_Koyock_14,model_Koyock_15)
```

```
##                  n      MASE
## model_Koyock_1  507 0.9221834
## model_Koyock_2  507 0.9718873
## model_Koyock_3  507 0.9255483
## model_Koyock_4  507 1.0024611
## model_Koyock_5  507 0.9411883
## model_Koyock_6  507 1.1075274
## model_Koyock_7  507 0.9201017
## model_Koyock_8  507 0.9281007
## model_Koyock_9  507 1.0132028
## model_Koyock_10 507 0.9581315
## model_Koyock_11 507 1.1093516
## model_Koyock_12 507 0.9392184
## model_Koyock_13 507 0.9228863
## model_Koyock_14 507 0.9194400
## model_Koyock_15 507 0.9549093
```

```
AIC(model_Koyock_1)
```

```
## [1] 2025.937
```

```
AIC(model_Koyock_2)
```

```
## [1] 2067.979
```

```
AIC(model_Koyock_3)
```

```
## [1] 2031.724
```

```
AIC(model_Koyock_4)
```

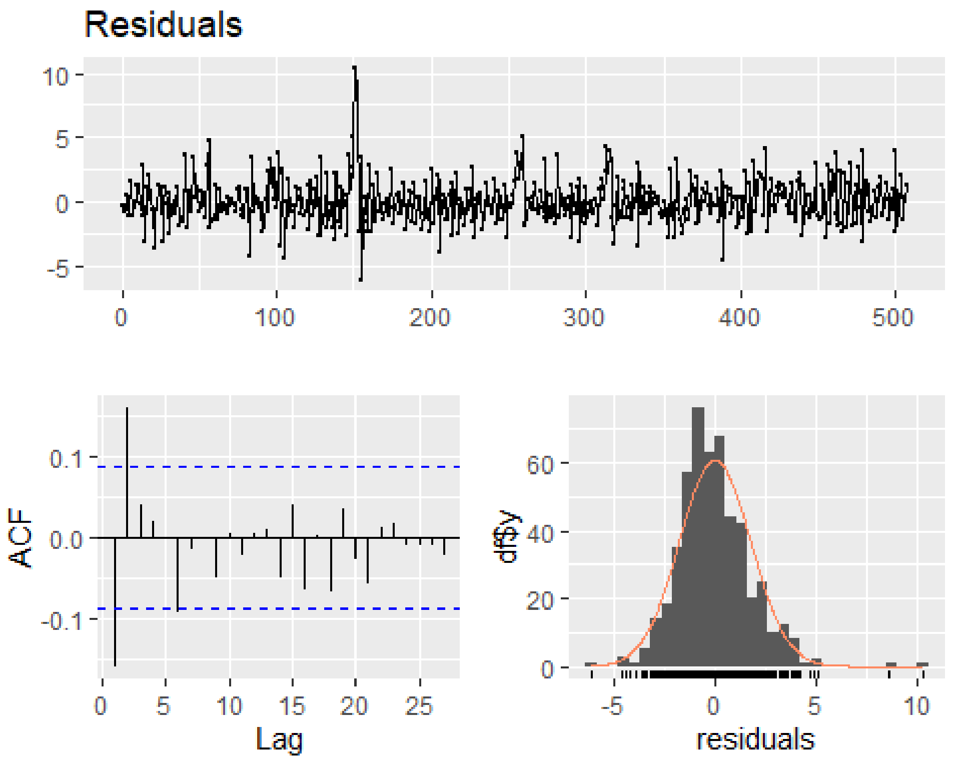
```
## [1] 2099.204

AIC(model_Koyock_5)

## [1] 2044.127

AIC(model_Koyock_6)

## [1] 2187.022

AIC(model_Koyock_7)

## [1] 2025.25

AIC(model_Koyock_8)

## [1] 2034.002

AIC(model_Koyock_9)

## [1] 2109.41

AIC(model_Koyock_10)

## [1] 2056.069

AIC(model_Koyock_11)

## [1] 2188.264

AIC(model_Koyock_12)

## [1] 2042.771

AIC(model_Koyock_13)

## [1] 2026.724

AIC(model_Koyock_14)

## [1] 2024.521

AIC(model_Koyock_15)

## [1] 2053.76

BIC(model_Koyock_1)

## [1] 2042.851

BIC(model_Koyock_2)

## [1] 2084.893

BIC(model_Koyock_3)

## [1] 2048.638
```

```
BIC(model_Koyock_4)

## [1] 2116.118

BIC(model_Koyock_5)

## [1] 2061.041

BIC(model_Koyock_6)

## [1] 2203.936

BIC(model_Koyock_7)

## [1] 2042.164

BIC(model_Koyock_8)

## [1] 2050.916

BIC(model_Koyock_9)

## [1] 2126.324

BIC(model_Koyock_10)

## [1] 2072.983

BIC(model_Koyock_11)

## [1] 2205.178

BIC(model_Koyock_12)

## [1] 2059.685

BIC(model_Koyock_13)

## [1] 2043.638

BIC(model_Koyock_14)

## [1] 2041.435

BIC(model_Koyock_15)

## [1] 2070.674
```

 * * Taking into account the AIC, BIC, and MASE values, the “model_Koyock_14” emerges as the preferred predictive model because it has the lowest MASE, AIC and BIC values. Now, let’s delve into a comprehensive analysis of this specific model.

```
summary(model_Koyock_14$model)

##
## 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

checkresiduals(model_Koyock_14$model)
```



```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 33.071, df = 10, p-value = 0.000265
##
## Model df: 0.   Total lags used: 10
```

```
vif(model_Koyock_14$model)> 10

##   Y.1   X.t
## FALSE FALSE
```

 * “model_Koyock_14” is a better fitted model compared to the other models as indicated by higher Adjusted R-squared, and the significant relationship between the independent variable and the error term based on the Wu-Hausman test. Furthermore, the Root Standard Error (RSE) 1.773 for the Koyock transformation has significantly reduced compared to previous models.

- Additionally, there is no significant multicollinearity issue in “model_Koyock_14” as evidenced by the Variance Inflation Factor (VIF) values for all lag variables being below 10.

- However, the Breusch-Godfrey test results suggest the presence of serial correlation in the residuals of this model, supported by significant lags in the autocorrelation function (ACF) plot and a deviation from a normal distribution in the histogram. Despite this issue, the model remains superior due to its other favorable characteristics of multi-collinearity, higher adjusted R-square and lower RSE.

Autoregressive DLM

```
for (i in 1:5){
 for(j in 1:5){
  model_1_ADLM <- ardlDlm(x= as.vector(chem_1 + chem_2 + temp_ts +
particle),y=as.vector(mort_ts), p = i , q = j )
  cat("p =", i, "q =", j, "AIC =", AIC(model_1_ADLM $model), "BIC =", BIC(model_1_ADLM
$model), "MASE =", MASE(model_1_ADLM )$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
```

* Looking at the above output p= 5 and q =5 has the lowest MASE, therefore further exploration will be conducted using summary and diagnostic checking.

```
model_ard_1 <- ardlDlm(x=as.vector(particle), y=as.vector(mort_ts),p = 5, q = 5)
model_ard_2 <- ardlDlm(x=as.vector(chem_1), y=as.vector(mort_ts),p = 5, q = 5)
model_ard_3 <- ardlDlm(x=as.vector(chem_2), y=as.vector(mort_ts),p = 5, q = 5)
model_ard_4 <- ardlDlm(x=as.vector(temp_ts), y=as.vector(mort_ts),p = 5, q = 5)
model_ard_5 <- ardlDlm(x=as.vector(chem_1 + chem_2 + temp_ts + particle), y=as.vector(mort_ts),p =
5, q = 5)
model_ard_6 <- ardlDlm(x=as.vector(chem_1 + chem_2 + temp_ts), y=as.vector(mort_ts),p = 5, q = 5)
model_ard_7 <- ardlDlm(x=as.vector(chem_1 + chem_2 + particle), y=as.vector(mort_ts),p = 5, q = 5)
model_ard_8 <- ardlDlm(x=as.vector(chem_1 + chem_2), y=as.vector(mort_ts),p = 5, q = 5)
model_ard_9 <- ardlDlm(x=as.vector(chem_1 + temp_ts), y=as.vector(mort_ts),p = 5, q = 5)
model_ard_10 <- ardlDlm(x=as.vector(chem_1 + temp_ts + particle), y=as.vector(mort_ts),p = 5, q = 5)
model_ard_11 <- ardlDlm(x=as.vector(chem_2 + temp_ts), y=as.vector(mort_ts),p = 5, q = 5)
model_ard_12 <- ardlDlm(x=as.vector(chem_2 + temp_ts + particle), y=as.vector(mort_ts),p = 5, q = 5)
model_ard_13 <- ardlDlm(x=as.vector(chem_1 + particle), y=as.vector(mort_ts),p = 5, q = 5)
model_ard_14 <- ardlDlm(x=as.vector(chem_2 + particle), y=as.vector(mort_ts),p = 5, q = 5)
model_ard_15 <- ardlDlm(x=as.vector(temp_ts + particle), y=as.vector(mort_ts),p = 5, q = 5)

sort.score(AIC(model_ard_1$model,model_ard_2$model,model_ard_3$model,model_ard_4$model,
model_ard_5$model, model_ard_6$model, model_ard_7$model, model_ard_8$model,
model_ard_9$model, model_ard_10$model, model_ard_11$model,
model_ard_12$model,model_ard_13$model, model_ard_14$model), score = "aic")

##                   df      AIC
## model_ard_4$model  13 1984.211
## model_ard_9$model  13 1985.261
## model_ard_1$model  13 1987.645
## model_ard_14$model 13 1987.803
## model_ard_7$model  13 1988.432
## model_ard_13$model 13 1988.437
## model_ard_3$model  13 1993.320
## model_ard_12$model 13 1994.392
## model_ard_8$model  13 1994.394
## model_ard_5$model  13 1994.843
## model_ard_10$model 13 1999.157
## model_ard_11$model 13 2002.008
## model_ard_6$model  13 2002.184
## model_ard_2$model  13 2008.913

sort.score(BIC(model_ard_1$model,model_ard_2$model,model_ard_3$model,model_ard_4$model,
model_ard_5$model, model_ard_6$model, model_ard_7$model, model_ard_8$model,
model_ard_9$model, model_ard_10$model, model_ard_11$model,
model_ard_12$model,model_ard_13$model, model_ard_14$model), score = "bic")
```

```
##                     df      BIC
## model_ard_4$model  13 2039.078
## model_ard_9$model  13 2040.129
## model_ard_1$model  13 2042.512
## model_ard_14$model 13 2042.671
## model_ard_7$model  13 2043.299
## model_ard_13$model 13 2043.304
## model_ard_3$model  13 2048.188
## model_ard_12$model 13 2049.259
## model_ard_8$model  13 2049.261
## model_ard_5$model  13 2049.711
## model_ard_10$model 13 2054.025
## model_ard_11$model 13 2056.876
## model_ard_6$model  13 2057.052
## model_ard_2$model  13 2063.780
```

```
MASE(model_ard_1, model_ard_2, model_ard_3, model_ard_4, model_ard_5, model_ard_6, model_ard_7,
     model_ard_8, model_ard_9, model_ard_10, model_ard_11, model_ard_12, model_ard_13, model_ard_14)
```

```
##                n      MASE
## model_ard_1  503 0.8833224
## model_ard_2  503 0.8991869
## model_ard_3  503 0.8870239
## model_ard_4  503 0.8829339
## model_ard_5  503 0.8863831
## model_ard_6  503 0.8917270
## model_ard_7  503 0.8828656
## model_ard_8  503 0.8881683
## model_ard_9  503 0.8837403
## model_ard_10 503 0.8881237
## model_ard_11 503 0.8915162
## model_ard_12 503 0.8859138
## model_ard_13 503 0.8837951
## model_ard_14 503 0.8823001
```

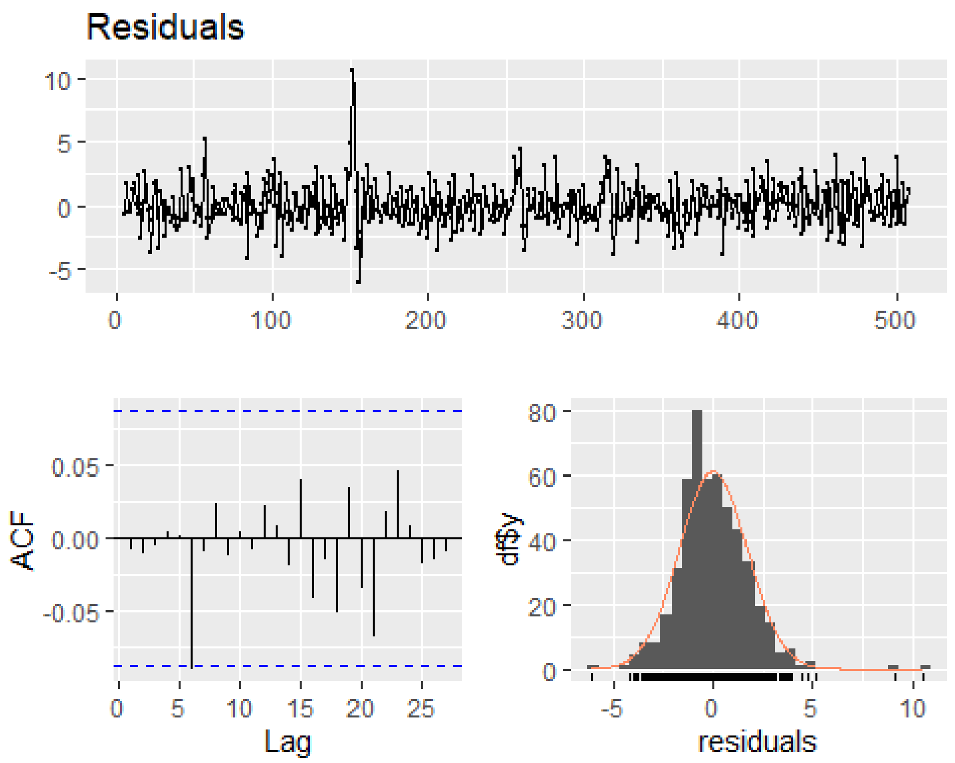
* Considering the lowest MASE, model_ard_14 is the best model. Now, let’s delve into a comprehensive analysis of this specific model.

```
summary(model_ard_14$model)
```

```
## 
## 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
```

```
checkresiduals(model_ard_14$model)
```



```
##
##  Breusch-Godfrey test for serial correlation of order up to 15
##
```

```
## data:  Residuals
## LM test = 12.02, df = 15, p-value = 0.6775

vif(model_ard_14$model)>10

##      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)
##    FALSE     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE
## L(y.t, 3) L(y.t, 4) L(y.t, 5)
##    FALSE     FALSE     FALSE
```

* The model labeled "model_ard_14" appears to be a better fit when compared to other models. This is evident from its higher Adjusted R-squared value of 64.1%, indicating a significant relationship between the independent variable and the error term according to the Wu-Hausman test. Additionally, the Root Standard Error (RSE) for the Koyock transformation has slightly decreased to 1.72 compared to previous models.

- Furthermore, there is no significant issue of multicollinearity in "model_ard_14" as indicated by Variance Inflation Factor (VIF) values for all lag variables staying below 10.
- However, the results of the Breusch-Godfrey test suggest the absence of serial correlation in the residuals of this model. There are no significant lags in the autocorrelation function (ACF) plot, and the histogram does not show a deviation from a normal distribution. Taking all these factors into account, "model_ard_14" is considered a better fit, and it's worth noting that it also has the lowest mean absolute scaled error (mase) among all the models.

Dynamic Linear Modelling

```
Y.t <- log(mort_ts)
t <- 450
S.t <- 1*(seq(Y.t)==T)
S.t.1 <- lag(S.t, +1)

dynlm_1<- dynlm(Y.t~ L(Y.t, k=1) + S.t + trend(Y.t)+ season(Y.t))
dynlm_2<- dynlm(Y.t~ L(Y.t, k=2) + S.t + trend(Y.t)+ season(Y.t))
dynlm_3<- dynlm(Y.t~ L(Y.t, k=1) + S.t + S.t.1 +trend(Y.t)+ season(Y.t))
dynlm_4 <- dynlm(Y.t~ L(Y.t, k=1) + S.t + trend(Y.t))
dynlm_5 <- dynlm(Y.t~ particle + L(Y.t, k=1) + S.t + trend(Y.t))
dynlm_6 <- dynlm(Y.t~ chem_1+ L(Y.t, k=1) + S.t + trend(Y.t))
dynlm_7 <- dynlm(Y.t~ chem_2+ L(Y.t, k=1) + S.t + trend(Y.t))
dynlm_8 <- dynlm(Y.t~ temp_ts+ L(Y.t, k=1) + S.t + trend(Y.t))
dynlm_9 <- dynlm(Y.t~ chem_1 + chem_2 + temp_ts + particle+ L(Y.t, k=1) + S.t + trend(Y.t))
dynlm_10 <- dynlm(Y.t~ chem_1 + chem_2 + temp_ts + L(Y.t, k=1) + S.t + trend(Y.t))
dynlm_11 <- dynlm(Y.t~ chem_1 + chem_2 + particle + L(Y.t, k=1) + S.t + trend(Y.t))
dynlm_12 <- dynlm(Y.t~ chem_1 + chem_2 + L(Y.t, k=1) + S.t + trend(Y.t))
dynlm_13 <- dynlm(Y.t~ chem_1 + temp_ts + L(Y.t, k=1) + S.t + trend(Y.t))
dynlm_14 <- dynlm(Y.t~ chem_1 + temp_ts + particle + L(Y.t, k=1) + S.t + trend(Y.t))
dynlm_15 <- dynlm(Y.t~ chem_2 + temp_ts + L(Y.t, k=1) + S.t + trend(Y.t))
dynlm_16 <- dynlm(Y.t~ chem_2 + temp_ts + particle + L(Y.t, k=1) + S.t + trend(Y.t))
dynlm_17 <- dynlm(Y.t~ chem_1 + particle + L(Y.t, k=1) + S.t + trend(Y.t))
dynlm_18 <- dynlm(Y.t~ chem_2 + particle + L(Y.t, k=1) + S.t + trend(Y.t))
```

```
dynlm_19 <- dynlm(Y.t~ temp_ts + particle + L(Y.t, k=1) + S.t + trend(Y.t))
dynlm_20<- dynlm(Y.t~ L(Y.t, k=1))
dynlm_21<- dynlm(Y.t~ L(Y.t, k=1) + L(Y.t, k=2))
dynlm_22 <- dynlm(Y.t~ particle + L(Y.t, k=1))

MASE(lm(dynlm_1),lm(dynlm_2),lm(dynlm_3),lm(dynlm_4),lm(dynlm_5),lm(dynlm_6),lm(dynlm_7),
lm(dynlm_8),
lm(dynlm_9),lm(dynlm_10),lm(dynlm_11),lm(dynlm_12),lm(dynlm_13),lm(dynlm_14),lm(dynlm_15),l
m(dynlm_16),lm(dynlm_17),lm(dynlm_18),lm(dynlm_19),
lm(dynlm_20),lm(dynlm_21),lm(dynlm_22))

##                n      MASE
## lm(dynlm_1)  507 0.8266331
## lm(dynlm_2)  506 0.8297852
## lm(dynlm_3)  507 0.8263200
## lm(dynlm_4)  507 0.9220813
## lm(dynlm_5)  507 0.8980311
## lm(dynlm_6)  507 0.9181876
## lm(dynlm_7)  507 0.8967984
## lm(dynlm_8)  507 0.9179619
## lm(dynlm_9)  507 0.8830838
## lm(dynlm_10) 507 0.8827862
## lm(dynlm_11) 507 0.8873709
## lm(dynlm_12) 507 0.8896369
## lm(dynlm_13) 507 0.9070736
## lm(dynlm_14) 507 0.8931002
## lm(dynlm_15) 507 0.8848019
## lm(dynlm_16) 507 0.8851251
## lm(dynlm_17) 507 0.8968902
## lm(dynlm_18) 507 0.8945977
## lm(dynlm_19) 507 0.8931000
## lm(dynlm_20) 507 0.9221145
## lm(dynlm_21) 506 0.8814864
## lm(dynlm_22) 507 0.8989073
```

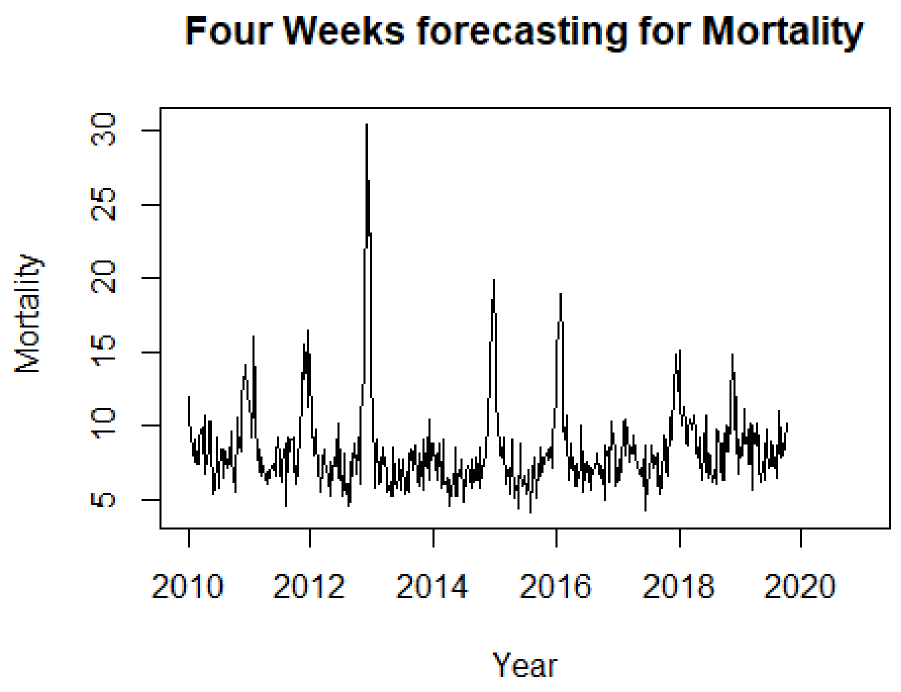
- Considering the lowest MASE, dynlm_3 is the best model. Now, let's delve into a comprehensive analysis of this specific model.

```
summary(dynlm_1)

##
## 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
## -0.61311 -0.11413 -0.00144  0.12438  0.54019
##
## Coefficients: (1 not defined because of singularities)
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.1377278  0.1172362   9.705  < 2e-16 ***
## L(Y.t, k = 1)  0.5180955  0.0402711  12.865  < 2e-16 ***
## S.t                   NA         NA      NA       NA
## trend(Y.t)     0.0016891  0.0030875   0.547  0.58460
## season(Y.t)2  -0.0695875  0.0896537  -0.776  0.43805
## season(Y.t)3  -0.1390516  0.0897162  -1.550  0.12186
## season(Y.t)4  -0.0675428  0.0899958  -0.751  0.45334
## season(Y.t)5   0.0443480  0.0899481   0.493  0.62222
## season(Y.t)6  -0.1323856  0.0896852  -1.476  0.14061
## season(Y.t)7  -0.1700069  0.0899335  -1.890  0.05935 .
## season(Y.t)8  -0.1726647  0.0903281  -1.912  0.05657 .
## season(Y.t)9  -0.1231372  0.0906136  -1.359  0.17485
## season(Y.t)10 -0.1635149  0.0904878  -1.807  0.07142 .
## season(Y.t)11 -0.2369358  0.0906530  -2.614  0.00926 **
## season(Y.t)12 -0.0910600  0.0912494  -0.998  0.31885
## season(Y.t)13 -0.1557785  0.0905831  -1.720  0.08616 .
## season(Y.t)14 -0.2316003  0.0906584  -2.555  0.01096 *
## season(Y.t)15 -0.1211816  0.0912123  -1.329  0.18466
## season(Y.t)16 -0.2896734  0.0907481  -3.192  0.00151 **
## season(Y.t)17 -0.2314405  0.0917369  -2.523  0.01198 *
## season(Y.t)18 -0.2489709  0.0918306  -2.711  0.00696 **
## season(Y.t)19 -0.1605706  0.0920371  -1.745  0.08173 .
## season(Y.t)20 -0.1979290  0.0913886  -2.166  0.03085 *
## season(Y.t)21 -0.2821022  0.0913709  -3.087  0.00214 **
## season(Y.t)22 -0.1288685  0.0920763  -1.400  0.16232
## season(Y.t)23 -0.2082815  0.0911645  -2.285  0.02279 *
## season(Y.t)24 -0.1839622  0.0913320  -2.014  0.04458 *
## season(Y.t)25 -0.2146893  0.0912348  -2.353  0.01904 *
## season(Y.t)26 -0.2388662  0.0914203  -2.613  0.00928 **
## season(Y.t)27 -0.2644414  0.0917214  -2.883  0.00413 **
## season(Y.t)28 -0.2484090  0.0921181  -2.697  0.00727 **
## season(Y.t)29 -0.1706420  0.0921839  -1.851  0.06481 .
## season(Y.t)30 -0.2148576  0.0915338  -2.347  0.01934 *
## season(Y.t)31 -0.2391287  0.0915815  -2.611  0.00932 **
## season(Y.t)32 -0.2941650  0.0918128  -3.204  0.00145 **
## season(Y.t)33 -0.0764021  0.0924550  -0.826  0.40903
## season(Y.t)34 -0.1132510  0.0909396  -1.245  0.21365
## season(Y.t)35 -0.2725343  0.0905794  -3.009  0.00277 **
## season(Y.t)36 -0.1448735  0.0914737  -1.584  0.11394
## season(Y.t)37 -0.1764971  0.0910199  -1.939  0.05311 .
## season(Y.t)38 -0.1496598  0.0910220  -1.644  0.10083
## season(Y.t)39 -0.2490097  0.0908420  -2.741  0.00637 **
## season(Y.t)40 -0.1283501  0.0914629  -1.403  0.16121
## season(Y.t)41 -0.2205187  0.0934033  -2.361  0.01865 *
## season(Y.t)42 -0.1149764  0.0936841  -1.227  0.22036
## season(Y.t)43 -0.1362844  0.0930886  -1.464  0.14388
## season(Y.t)44 -0.1154598  0.0929491  -1.242  0.21481
## season(Y.t)45  0.0344011  0.0927669   0.371  0.71093
## season(Y.t)46 -0.0721129  0.0921296  -0.783  0.43419
## season(Y.t)47  0.0885969  0.0921803   0.961  0.33700
## season(Y.t)48  0.0933119  0.0919707   1.015  0.31085
## season(Y.t)49  0.1164319  0.0920250   1.265  0.20644
## season(Y.t)50  0.0162078  0.0921546   0.176  0.86047
## season(Y.t)51 -0.0032997  0.0920246  -0.036  0.97141
## season(Y.t)52 -0.0003321  0.0919756  -0.004  0.99712
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1951 on 453 degrees of freedom
## Multiple R-squared:  0.5683, Adjusted R-squared:  0.5177
## F-statistic: 11.25 on 53 and 453 DF,  p-value: < 2.2e-16
```

* The model labeled “dynlm_1” does not appear to be a better fit when compared to other models. This is evident from its slightly lower Adjusted R-squared value of 51.7% compared to the previous model, indicating a significant relationship between the independent variable and the error term according to the Wu-Hausman test. Additionally, the Root Standard Error (RSE) slightly increased to 1.73 compared to previous models.

- However, the results of the Breusch-Godfrey test suggest the presence of serial correlation in the residuals of this model. There are significant lags in the autocorrelation function (ACF) plot, and the histogram does not show a deviation from a normal distribution. Taking all these factors into account, “dynlm_3” is considered not a better fit, and it’s worth noting that it also has the lowest mean absolute scaled error (mase) among all the models but the lower RSE and slightly lower MASE value makes model not a better fit.

```
q = 4
n = nrow(dynlm_1$model)
model_1.frc = array(NA , (n + q))
model_1.frc[1:n] = Y.t[2:length(Y.t)]
trend = array(NA,q)
trend.start = dynlm_1$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,model_1.frc[n-1+i],1,trend[i],weeks)
model_1.frc[n+i] = as.vector(dynlm_1$coefficients) %*% data.new
}

par(mfrow=c(1,1))
plot(mort_ts,xlim=c(2010,2021),ylab=' Mortality',xlab='Year',main =
"Four Weeks forecasting for Mortality")
lines(ts(model_1.frc[(n+1):(n+q)],start=c(2019,42),frequency = 52),col="blue")
```



Exponential Smoothing

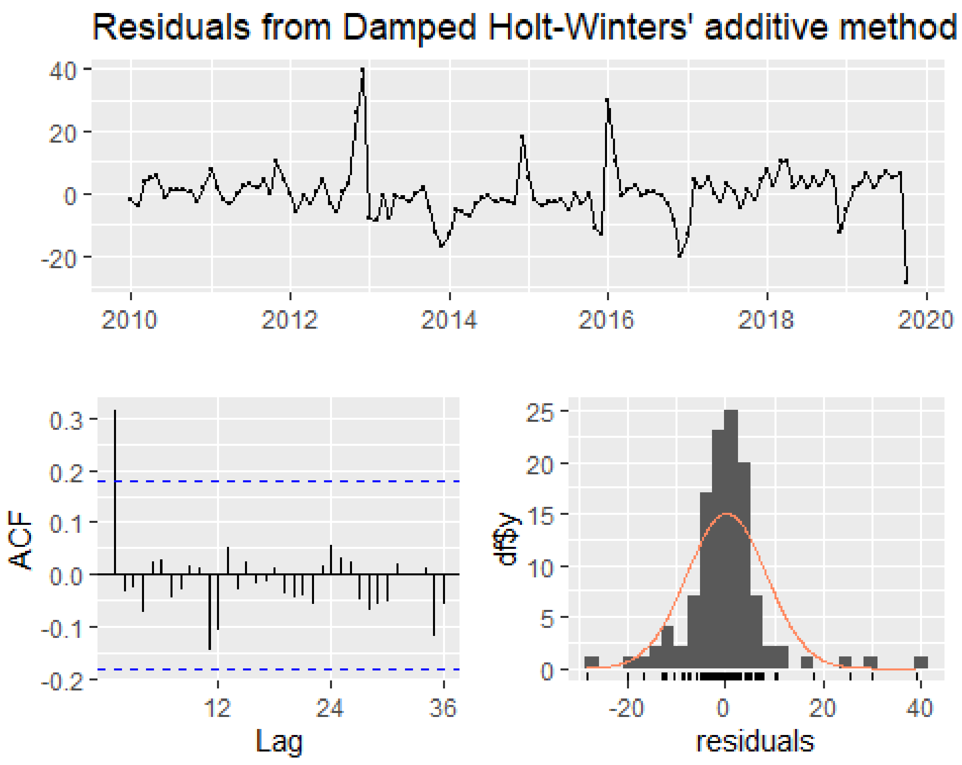
```
mort_zoo <- zoo(mort_ts)
mort_monthly <- aggregate(mort_zoo, as.yearmon, sum)
mort_monthly_ts <- ts(coredata(mort_monthly), start = start(mort_monthly), frequency = 12)
print(mort_monthly_ts)
```

```
##        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 10.11
```

```
seasonal <- c("additive", "multiplicative")
damped <- c(TRUE, FALSE)
expand <- expand.grid(seasonal, damped)
AIC <- numeric(4)
BIC <- numeric(4)
MASE <- numeric(4)
levels <- matrix(NA, nrow = 4, ncol = 2)
```

```
for (i in 1:4) {
  AIC[i] <- hw(mort_monthly_ts,
                  seasonal = as.character(expand[i, 1]),
                  damped = expand[i, 2])$model$aic
  BIC[i] <- accuracy(hw(mort_monthly_ts,
                           seasonal = as.character(expand[i, 1]),
                           damped = expand[i, 2]))[, 6]
  MASE[i] <- accuracy(hw(mort_monthly_ts,
                           seasonal = as.character(expand[i, 1]),
                           damped = expand[i, 2]))[, 6]
  levels[i, 1] <- as.character(expand[i, 1])
  levels[i, 2] <- expand[i, 2]
}

results <- data.frame(levels, MASE, AIC, BIC)
colnames(results) <- c("seasonal", "damped", "MASE", "AIC","BIC")
results

##         seasonal damped      MASE      AIC       BIC
## 1       additive   TRUE 0.7196122 1092.572 0.7196122
## 2 multiplicative   TRUE 0.7288054 1048.812 0.7288054
## 3       additive  FALSE 0.7435300 1091.411 0.7435300
## 4 multiplicative  FALSE 0.7376868 1045.149 0.7376868

model_hw_1 <- hw(mort_monthly_ts, seasonal = "additive", damped = TRUE)
model_hw_2 <- hw(mort_monthly_ts, seasonal = "multiplicative")
model_hw_3 <- hw(mort_monthly_ts, seasonal = "multiplicative", damped = TRUE)
model_hw_4 <- hw(mort_monthly_ts, seasonal = "multiplicative", damped = TRUE)
model_hw_5 <- hw(mort_monthly_ts, seasonal = "multiplicative", exponential =  TRUE)

summary(model_hw_1)

##
## Forecast method: Damped Holt-Winters' additive method
##
## Model Information:
## Damped Holt-Winters' additive method
##
## Call:
##  hw(y = mort_monthly_ts, seasonal = "additive", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 1e-04
##     beta  = 1e-04
##     gamma = 1e-04
##     phi   = 0.969
##
##   Initial states:
##     l = 37.4387
##     b = -0.0588
##     s = 14.027 7.7066 2.8124 -6.5881 -7.2838 -2.3644
##           -8.0405 -8.0556 -0.5173 -5.1942 -1.2036 14.7016
## 
##   sigma:  8.7534
## 
##      AIC     AICc      BIC 
## 1092.572 1099.482 1142.445 
## 
## Error measures:
##                     ME     RMSE     MAE       MPE     MAPE      MASE      ACF1
## Training set 0.1807267 8.098346 5.15222 -3.974345 14.97567 0.7196122 0.3147791
## 
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Nov 2019       43.37619 32.15826 54.59412 26.21985 60.53253
## Dec 2019       49.69709 38.47916 60.91502 32.54074 66.85343
## Jan 2020       50.37197 39.15404 61.58990 33.21563 67.52831
## Feb 2020       34.46531 23.24738 45.68324 17.30896 51.62166
## Mar 2020       30.47577 19.25783 41.69370 13.31942 47.63211
## Apr 2020       35.15242 23.93448 46.37035 17.99607 52.30877
## May 2020       27.61441 16.39647 38.83235 10.45805 44.77076
## Jun 2020       27.62971 16.41177 38.84765 10.47335 44.78607
## Jul 2020       33.30549 22.08755 44.52344 16.14913 50.46186
## Aug 2020       28.38615 17.16820 39.60410 11.22978 45.54253
## Sep 2020       29.08180 17.86385 40.29976 11.92542 46.23818
## Oct 2020       38.47882 27.26086 49.69678 21.32243 55.63521
## Nov 2020       43.37595 32.15798 54.59392 26.21955 60.53236
## Dec 2020       49.69685 38.47888 60.91483 32.54044 66.85327
## Jan 2021       50.37174 39.15376 61.58973 33.21531 67.52818
## Feb 2021       34.46509 23.24709 45.68309 17.30864 51.62154
## Mar 2021       30.47555 19.25754 41.69356 13.31909 47.63202
## Apr 2021       35.15221 23.93419 46.37023 17.99573 52.30869
## May 2021       27.61421 16.39618 38.83224 10.45771 44.77071
## Jun 2021       27.62952 16.41147 38.84757 10.47300 44.78604
## Jul 2021       33.30531 22.08725 44.52337 16.14876 50.46185
## Aug 2021       28.38597 17.16790 39.60405 11.22940 45.54254
## Sep 2021       29.08163 17.86353 40.29972 11.92503 46.23822
## Oct 2021       38.47865 27.26054 49.69676 21.32203 55.63527
```

```
checkresiduals(model_hw_1)
```

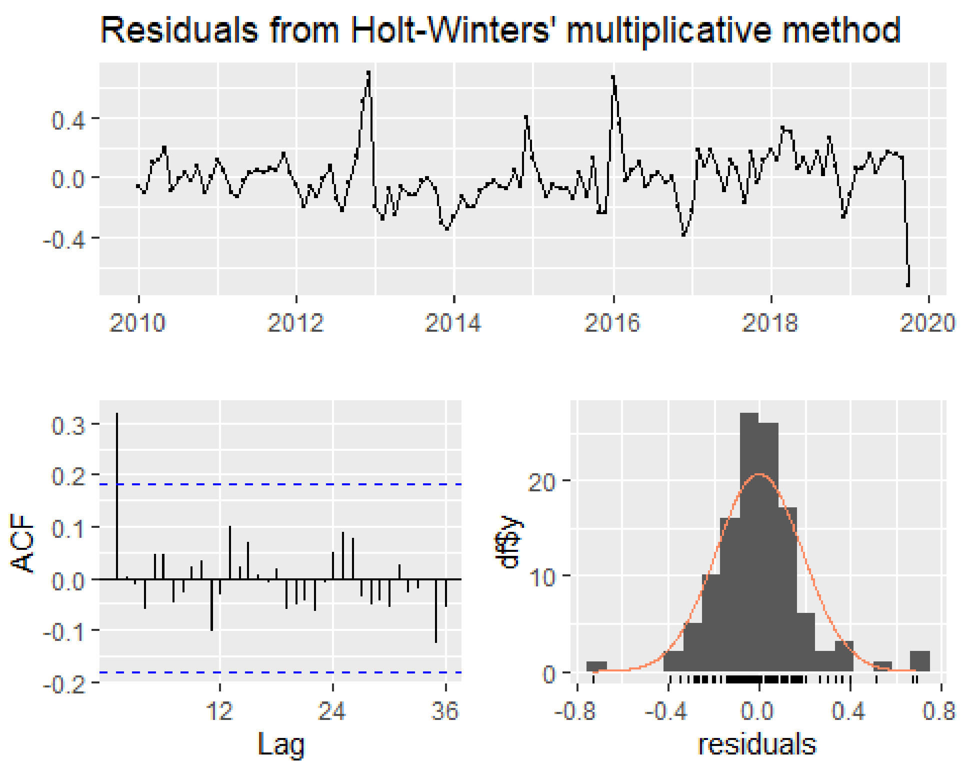


```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt-Winters' additive method
## Q* = 20.323, df = 24, p-value = 0.6783
##
## Model df: 0.   Total lags used: 24
```

```
summary(model_hw_2)
```

```
##
## Forecast method: Holt-Winters' multiplicative method
##
## Model Information:
## Holt-Winters' multiplicative method
##
## Call:
##  hw(y = mort_monthly_ts, seasonal = "multiplicative")
##
##   Smoothing parameters:
##     alpha = 0.0316
##     beta  = 1e-04
##     gamma = 1e-04
##
##   Initial states:
##     l = 37.8454
##     b = -0.0086
##     s = 1.396 1.234 0.9902 0.8493 0.7938 0.9472
##            0.8046 0.774 0.9868 0.8671 0.9499 1.4071
##
##   sigma:  0.2029
##
##      AIC     AICc      BIC
## 1045.149 1051.269 1092.251
##
## Error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.07068833 8.108542 5.281629 -4.671699 15.30341 0.7376868
##                   ACF1
## Training set 0.3016914
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Nov 2019       44.70616 33.08100 56.33133 26.92700 62.48532
## Dec 2019       50.56436 37.40897 63.71975 30.44492 70.68379
## Jan 2020       50.95541 37.69129 64.21952 30.66969 71.24112
## Feb 2020       34.39032 25.43349 43.34715 20.69203 48.08861
## Mar 2020       31.38218 23.20445 39.55991 18.87542 43.88893
## Apr 2020       35.70499 26.39582 45.01417 21.46783 49.94215
## May 2020       28.00019 20.69591 35.30447 16.82926 39.17112
## Jun 2020       29.09585 21.50163 36.69006 17.48150 40.71020
## Jul 2020       34.24719 25.30357 43.19082 20.56910 47.92529
## Aug 2020       28.69352 21.19612 36.19092 17.22724 40.15980
## Sep 2020       30.69048 22.66687 38.71409 18.41942 42.96153
## Oct 2020       35.77142 26.41428 45.12856 21.46091 50.08193
## Nov 2020       44.56583 32.90169 56.22997 26.72707 62.40459
## Dec 2020       50.40559 37.20562 63.60556 30.21798 70.59320
## Jan 2021       50.79537 37.48583 64.10492 30.44018 71.15056
## Feb 2021       34.28229 25.29445 43.27012 20.53658 48.02799
## Mar 2021       31.28357 23.07724 39.48990 18.73307 43.83406
## Apr 2021       35.59277 26.25069 44.93484 21.30530 49.88024
## May 2021       27.91215 20.58180 35.24251 16.70134 39.12297
## Jun 2021       29.00435 21.38274 36.62595 17.34811 40.66059
## Jul 2021       34.13946 25.16326 43.11567 20.41154 47.86739
## Aug 2021       28.60324 21.07825 36.12823 17.09476 40.11171
## Sep 2021       30.59389 22.54046 38.64731 18.27723 42.91054
## Oct 2021       35.65881 26.26655 45.05107 21.29459 50.02303
```

```
checkresiduals(model_hw_2)
```

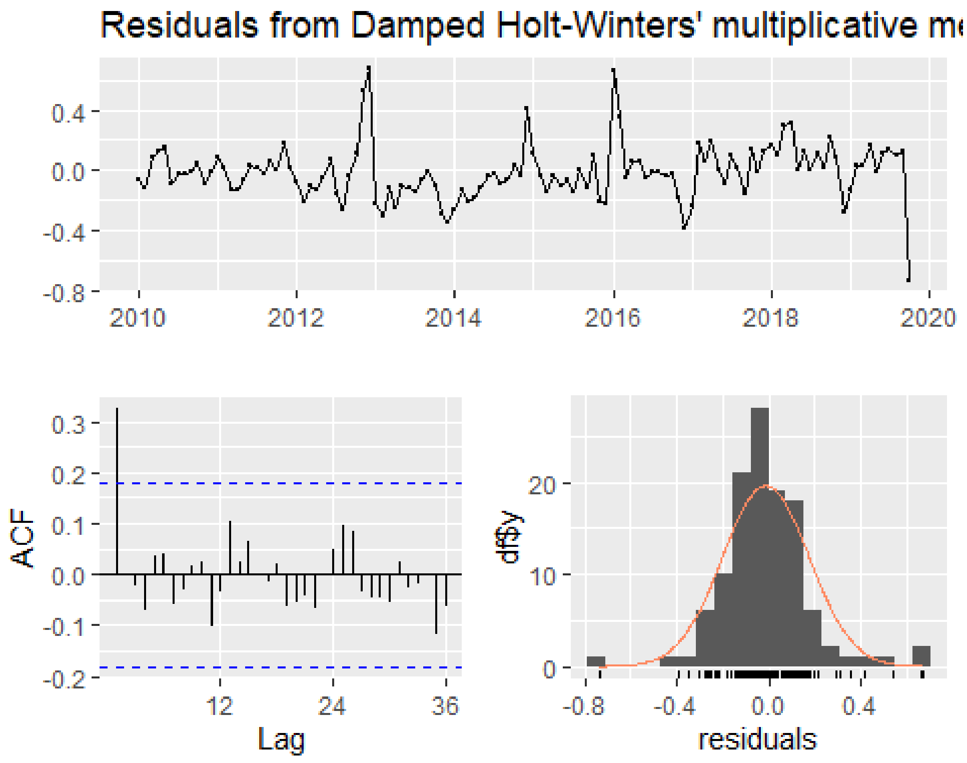


```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' multiplicative method
## Q* = 19.774, df = 24, p-value = 0.7096
##
## Model df: 0.   Total lags used: 24
```

```
summary(model_hw_3)
```

```
##
## Forecast method: Damped Holt-Winters' multiplicative method
##
## Model Information:
## Damped Holt-Winters' multiplicative method
##
## Call:
##  hw(y = mort_monthly_ts, seasonal = "multiplicative", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.0413
##     beta  = 1e-04
##     gamma = 1e-04
##     phi   = 0.954
##
##   Initial states:
##     l = 37.811
##     b = 0.0774
##     s = 1.3823 1.1964 1.009 0.8349 0.8181 0.9537
##            0.7948 0.7963 0.9679 0.8832 0.9514 1.4121
##
##   sigma:  0.2028
##
##      AIC     AICc      BIC
## 1048.812 1055.721 1098.685
##
## Error measures:
##                     ME     RMSE     MAE       MPE     MAPE      MASE     ACF1
## Training set -0.4802064 8.143085 5.21804 -6.011649 15.27702 0.7288054 0.303279
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Nov 2019       43.85266 32.45536 55.24996 26.42200 61.28332
## Dec 2019       50.66859 37.48812 63.84906 30.51080 70.82638
## Jan 2020       51.76217 38.28521 65.23913 31.15094 72.37341
## Feb 2020       34.87751 25.78857 43.96645 20.97717 48.77785
## Mar 2020       32.37411 23.92998 40.81824 19.45992 45.28830
## Apr 2020       35.48361 26.22011 44.74712 21.31631 49.65092
## May 2020       29.19067 21.56316 36.81817 17.52541 40.85593
## Jun 2020       29.13639 21.51620 36.75658 17.48232 40.79046
## Jul 2020       34.96247 25.81029 44.11465 20.96542 48.95952
## Aug 2020       29.99277 22.13441 37.85113 17.97445 42.01109
## Sep 2020       30.60895 22.58188 38.63602 18.33260 42.88530
## Oct 2020       36.99176 27.28202 46.70151 22.14199 51.84154
## Nov 2020       43.86168 32.33818 55.38517 26.23801 61.48534
## Dec 2020       50.67853 37.35198 64.00508 30.29733 71.05973
## Jan 2021       51.77186 38.14541 65.39831 30.93201 72.61171
## Feb 2021       34.88374 25.69391 44.07356 20.82911 48.93836
## Mar 2021       32.37963 23.84172 40.91754 19.32202 45.43723
## Apr 2021       35.48938 26.12296 44.85581 21.16467 49.81410
## May 2021       29.19519 21.48291 36.90747 17.40028 40.99011
## Jun 2021       29.14070 21.43579 36.84561 17.35706 40.92434
## Jul 2021       34.96741 25.71346 44.22135 20.81472 49.12009
## Aug 2021       29.99681 22.05106 37.94255 17.84484 42.14878
## Sep 2021       30.61288 22.49655 38.72922 18.20002 43.02574
## Oct 2021       36.99630 27.17859 46.81400 21.98142 52.01118
```

```
checkresiduals(model_hw_3)
```

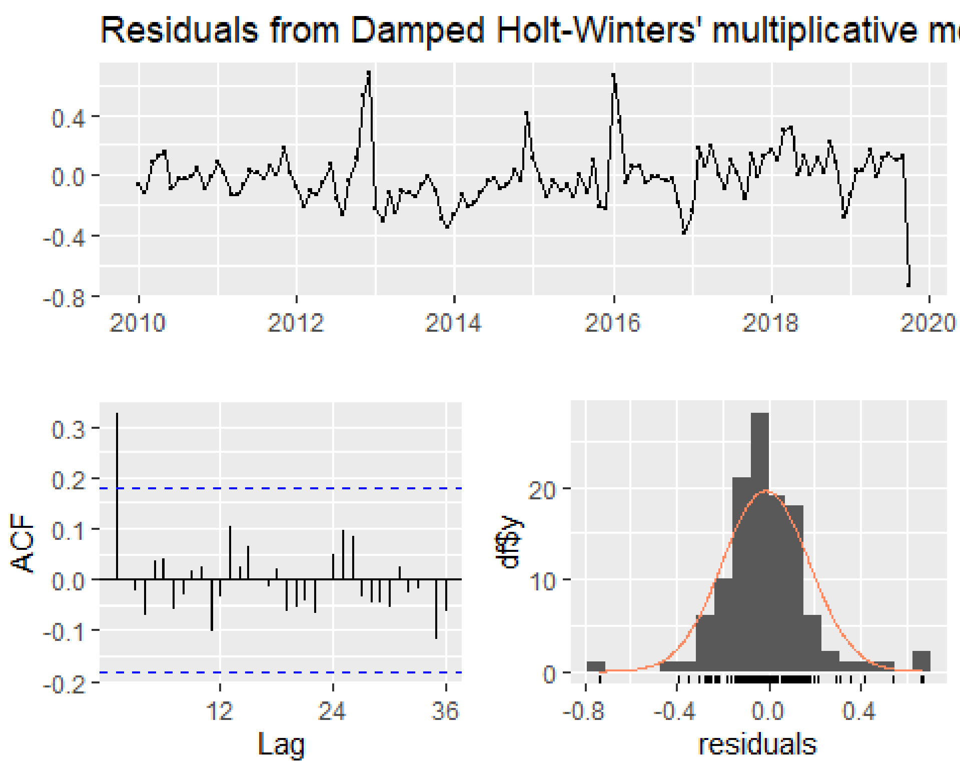


```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt-Winters' multiplicative method
## Q* = 20.654, df = 24, p-value = 0.6591
##
## Model df: 0.   Total lags used: 24
```

```
summary(model_hw_4)
```

```
##
## Forecast method: Damped Holt-Winters' multiplicative method
##
## Model Information:
## Damped Holt-Winters' multiplicative method
##
## Call:
##  hw(y = mort_monthly_ts, seasonal = "multiplicative", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.0413
##     beta  = 1e-04
##     gamma = 1e-04
##     phi   = 0.954
##
##   Initial states:
##     l = 37.811
##     b = 0.0774
##     s = 1.3823 1.1964 1.009 0.8349 0.8181 0.9537
##            0.7948 0.7963 0.9679 0.8832 0.9514 1.4121
##
##   sigma:  0.2028
##
##      AIC     AICc      BIC
## 1048.812 1055.721 1098.685
##
## Error measures:
##                      ME     RMSE     MAE       MPE     MAPE      MASE     ACF1
## Training set -0.4802064 8.143085 5.21804 -6.011649 15.27702 0.7288054 0.303279
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Nov 2019       43.85266 32.45536 55.24996 26.42200 61.28332
## Dec 2019       50.66859 37.48812 63.84906 30.51080 70.82638
## Jan 2020       51.76217 38.28521 65.23913 31.15094 72.37341
## Feb 2020       34.87751 25.78857 43.96645 20.97717 48.77785
## Mar 2020       32.37411 23.92998 40.81824 19.45992 45.28830
## Apr 2020       35.48361 26.22011 44.74712 21.31631 49.65092
## May 2020       29.19067 21.56316 36.81817 17.52541 40.85593
## Jun 2020       29.13639 21.51620 36.75658 17.48232 40.79046
## Jul 2020       34.96247 25.81029 44.11465 20.96542 48.95952
## Aug 2020       29.99277 22.13441 37.85113 17.97445 42.01109
## Sep 2020       30.60895 22.58188 38.63602 18.33260 42.88530
## Oct 2020       36.99176 27.28202 46.70151 22.14199 51.84154
## Nov 2020       43.86168 32.33818 55.38517 26.23801 61.48534
## Dec 2020       50.67853 37.35198 64.00508 30.29733 71.05973
## Jan 2021       51.77186 38.14541 65.39831 30.93201 72.61171
## Feb 2021       34.88374 25.69391 44.07356 20.82911 48.93836
## Mar 2021       32.37963 23.84172 40.91754 19.32202 45.43723
## Apr 2021       35.48938 26.12296 44.85581 21.16467 49.81410
## May 2021       29.19519 21.48291 36.90747 17.40028 40.99011
## Jun 2021       29.14070 21.43579 36.84561 17.35706 40.92434
## Jul 2021       34.96741 25.71346 44.22135 20.81472 49.12009
## Aug 2021       29.99681 22.05106 37.94255 17.84484 42.14878
## Sep 2021       30.61288 22.49655 38.72922 18.20002 43.02574
## Oct 2021       36.99630 27.17859 46.81400 21.98142 52.01118
```

```
checkresiduals(model_hw_4)
```

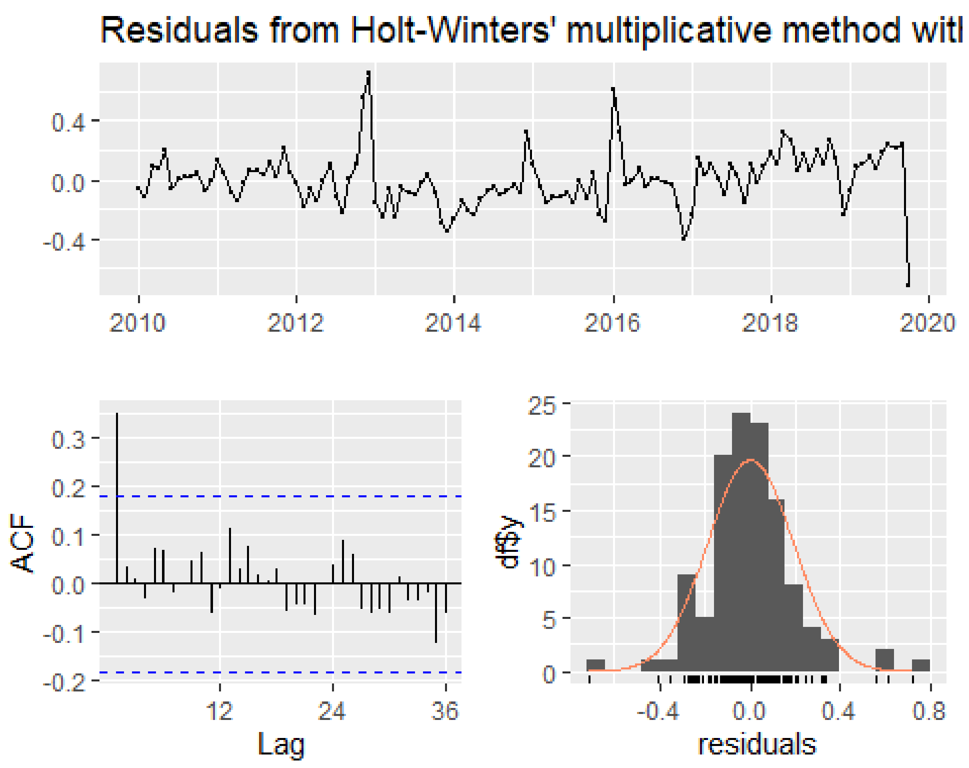


```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt-Winters' multiplicative method
## Q* = 20.654, df = 24, p-value = 0.6591
##
## Model df: 0.   Total lags used: 24
```

```
summary(model_hw_5)
```

```
##
## Forecast method: Holt-Winters' multiplicative method with exponential trend
##
## Model Information:
## Holt-Winters' multiplicative method with exponential trend
##
## Call:
##  hw(y = mort_monthly_ts, seasonal = "multiplicative", exponential = TRUE)
##
##   Smoothing parameters:
##     alpha = 1e-04
##     beta  = 1e-04
##     gamma = 1e-04
##
##   Initial states:
##     l = 37.8676
##     b = 0.9993
##     s = 1.4089 1.2042 1.0228 0.8177 0.7991 0.9387
##            0.792 0.7787 1.0213 0.8711 0.9535 1.3919
##
##   sigma:  0.204
##
##      AIC     AICc      BIC
## 1046.004 1052.124 1093.106
##
## Error measures:
##                     ME     RMSE      MAE       MPE    MAPE      MASE      ACF1
## Training set -0.0159365 8.068006 5.304868 -4.474637 15.4081 0.7409326 0.3246759
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Nov 2019       41.72554 30.83412 52.69191 24.91089 58.11681
## Dec 2019       48.78172 36.02198 61.92234 29.21561 68.94126
## Jan 2020       48.16561 35.45375 60.77860 29.02015 67.35370
## Feb 2020       32.97184 24.46814 41.52949 19.59660 45.93560
## Mar 2020       30.10130 22.15641 37.87372 18.18004 41.91792
## Apr 2020       35.26559 25.95121 44.62367 21.27419 49.64845
## May 2020       26.87314 19.93230 33.99080 15.90713 37.75338
## Jun 2020       27.31382 20.26199 34.58716 16.35954 38.47974
## Jul 2020       32.35194 23.93749 40.61952 19.42552 45.20546
## Aug 2020       27.51954 20.25303 34.79971 16.61545 38.47643
## Sep 2020       28.14159 20.70571 35.27018 16.71708 39.48994
## Oct 2020       35.17754 25.99895 44.40983 20.79587 49.22973
## Nov 2020       41.39015 30.75213 51.86658 24.56434 57.64525
## Dec 2020       48.38961 35.61974 61.31988 29.59975 68.21157
## Jan 2021       47.77846 35.44874 60.12598 28.71014 66.41056
## Feb 2021       32.70681 24.20826 41.36888 19.57320 45.93238
## Mar 2021       29.85935 22.23561 37.68350 18.26932 41.67688
## Apr 2021       34.98213 26.07123 43.82590 20.94331 49.54781
## May 2021       26.65714 19.75086 33.70641 15.98040 37.78762
## Jun 2021       27.09427 19.85188 34.16744 16.10569 37.89172
## Jul 2021       32.09189 23.63177 40.64055 19.15324 45.10439
## Aug 2021       27.29834 20.41123 34.40503 16.80602 38.55449
## Sep 2021       27.91538 20.81741 35.16694 16.92632 39.05299
## Oct 2021       34.89478 25.33986 44.06797 20.65993 49.05365
```

```
checkresiduals(model_hw_5)
```

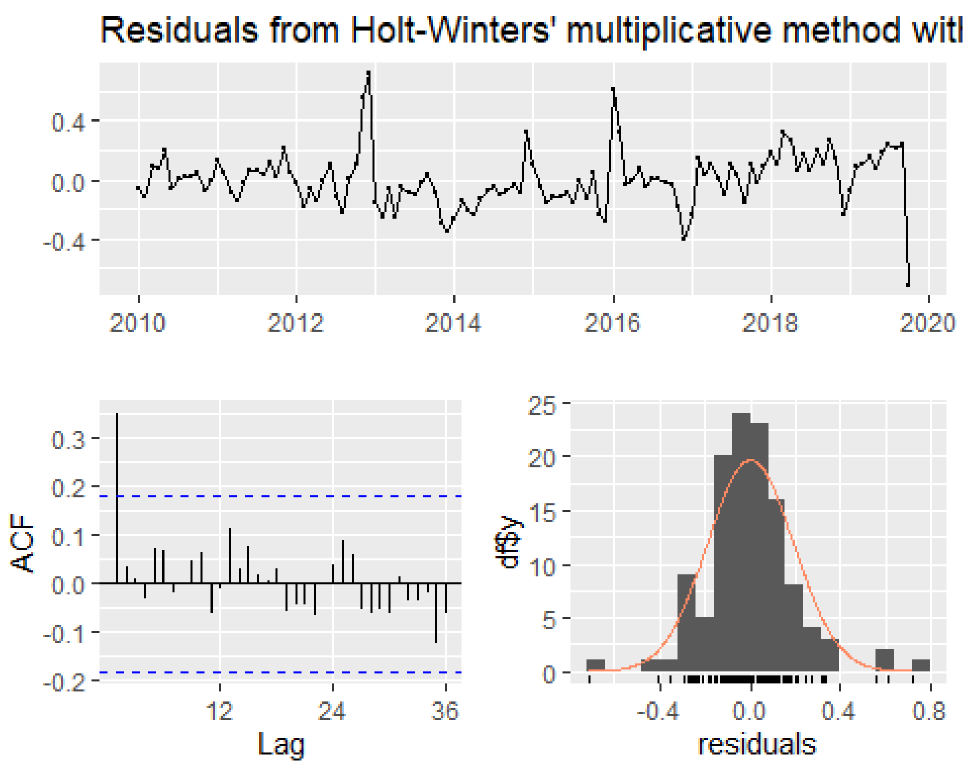


```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' multiplicative method with exponential trend
## Q* = 22.424, df = 24, p-value = 0.554
##
## Model df: 0.   Total lags used: 24
```

```
summary(model_hw_5)
```

```
##
## Forecast method: Holt-Winters' multiplicative method with exponential trend
##
## Model Information:
## Holt-Winters' multiplicative method with exponential trend
##
## Call:
##  hw(y = mort_monthly_ts, seasonal = "multiplicative", exponential = TRUE)
##
##   Smoothing parameters:
##     alpha = 1e-04
##     beta  = 1e-04
##     gamma = 1e-04
##
##   Initial states:
##     l = 37.8676
##     b = 0.9993
##     s = 1.4089 1.2042 1.0228 0.8177 0.7991 0.9387
##            0.792 0.7787 1.0213 0.8711 0.9535 1.3919
## 
##   sigma:  0.204
## 
##      AIC     AICc      BIC 
## 1046.004 1052.124 1093.106 
## 
## Error measures:
##                      ME     RMSE      MAE       MPE    MAPE      MASE      ACF1
## Training set -0.0159365 8.068006 5.304868 -4.474637 15.4081 0.7409326 0.3246759
## 
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Nov 2019       41.72554 30.83412 52.69191 24.91089 58.11681
## Dec 2019       48.78172 36.02198 61.92234 29.21561 68.94126
## Jan 2020       48.16561 35.45375 60.77860 29.02015 67.35370
## Feb 2020       32.97184 24.46814 41.52949 19.59660 45.93560
## Mar 2020       30.10130 22.15641 37.87372 18.18004 41.91792
## Apr 2020       35.26559 25.95121 44.62367 21.27419 49.64845
## May 2020       26.87314 19.93230 33.99080 15.90713 37.75338
## Jun 2020       27.31382 20.26199 34.58716 16.35954 38.47974
## Jul 2020       32.35194 23.93749 40.61952 19.42552 45.20546
## Aug 2020       27.51954 20.25303 34.79971 16.61545 38.47643
## Sep 2020       28.14159 20.70571 35.27018 16.71708 39.48994
## Oct 2020       35.17754 25.99895 44.40983 20.79587 49.22973
## Nov 2020       41.39015 30.75213 51.86658 24.56434 57.64525
## Dec 2020       48.38961 35.61974 61.31988 29.59975 68.21157
## Jan 2021       47.77846 35.44874 60.12598 28.71014 66.41056
## Feb 2021       32.70681 24.20826 41.36888 19.57320 45.93238
## Mar 2021       29.85935 22.23561 37.68350 18.26932 41.67688
## Apr 2021       34.98213 26.07123 43.82590 20.94331 49.54781
## May 2021       26.65714 19.75086 33.70641 15.98040 37.78762
## Jun 2021       27.09427 19.85188 34.16744 16.10569 37.89172
## Jul 2021       32.09189 23.63177 40.64055 19.15324 45.10439
## Aug 2021       27.29834 20.41123 34.40503 16.80602 38.55449
## Sep 2021       27.91538 20.81741 35.16694 16.92632 39.05299
## Oct 2021       34.89478 25.33986 44.06797 20.65993 49.05365
```

```
checkresiduals(model_hw_5)
```



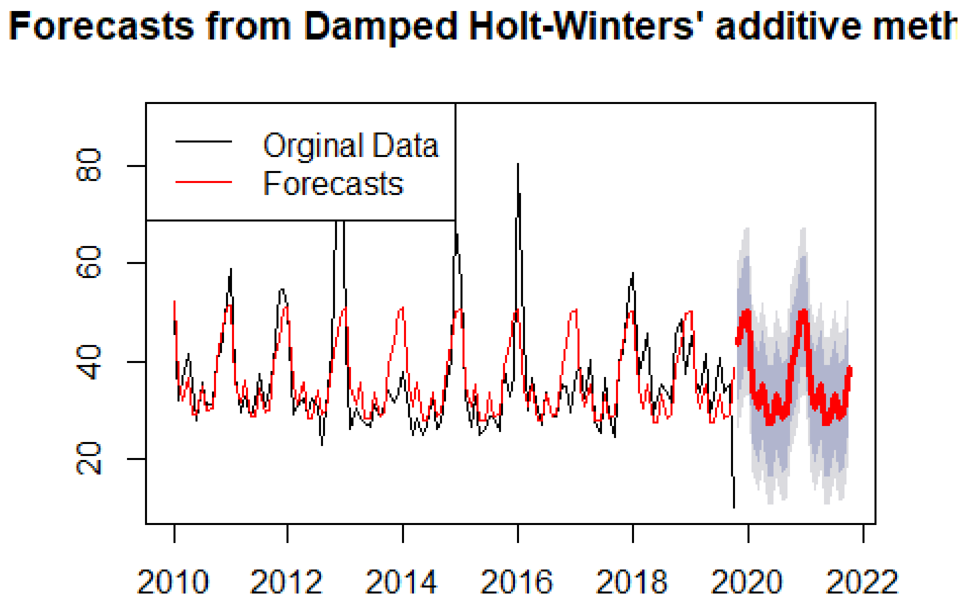
```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' multiplicative method with exponential trend
## Q* = 22.424, df = 24, p-value = 0.554
##
## Model df: 0.   Total lags used: 24
```

 * Considering all the models, model_hw_1 is considered to be the best model with comparatively lower MASE value and p-value greater than 5% level of significance, indicating that there is no presence of serial correlation. All the lags are below the 95% confidence interval except one.

```
model_hw_1_forecast<- hw(mort_monthly_ts, seasonal = "additive", damped = TRUE)

forecast_model_mean<- model_hw_1_forecast$mean

forecast_model_upper <- model_hw_1_forecast$upper
forecast_model_lower <- model_hw_1_forecast$lower

forecast_mortal_model <- ts.intersect(ts(forecast_model_lower, start=c(2019,1),frequency =
12),ts(forecast_model_mean,start=c(2019,1),frequency =
12),ts(forecast_model_upper,start=c(2019,1),frequency = 12))
```

Forecasting For Exponential Model

```
plot(model_hw_1_forecast, fcol = "red")
lines(fitted(model_hw_1_forecast), col = "red")
lines(model_hw_1_forecast$mean, col = "red", lwd= 3)
legend("topleft", lty=1, col= c("black","red"),c("Orginal Data","Forecasts"))
```



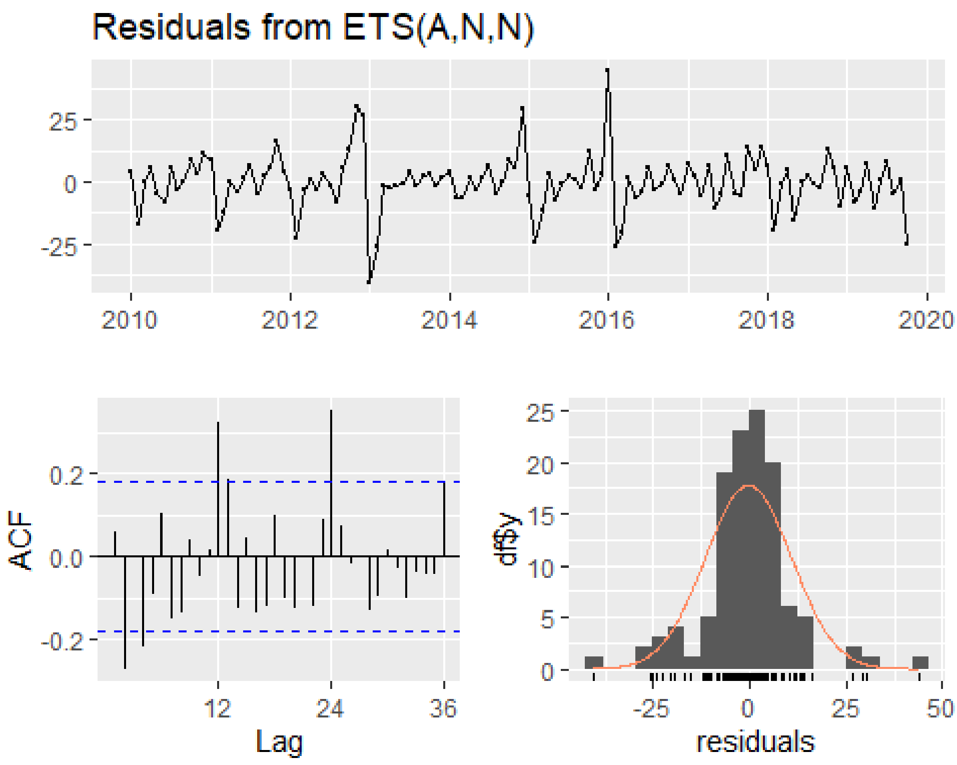
Modelling with state Space Models

- State-space models offer greater flexibility in defining the parametric structure. Within the ETS model, the initial element characterizes the trend, the second element captures the seasonality, and the third element accounts for the error.

```
variable <- c("AAA", "MAA", "MAM", "MMM")
damped <- c(TRUE, FALSE)
models <- expand.grid(variable, damped)
aic <- array(NA,8)
bic <- array(NA,8)
mase <- array(NA,8)
new_model<- array(NA,dim =c(8,2))
for(i in 1:8){
  ets_1<- ets(mort_monthly_ts, model = toString(models[i,1]), damped = models[i,2])
  aic[i] <- ets_1$aic
  bic[i] <- ets_1$bic
  mase[i] <- accuracy(ets_1)[6]
  new_model[i,1] <- toString(models[i,1])
  new_model[i,2] <- models[i,2]
}
```

```
new_1 <- data.frame(new_model, mase, aic,bic)
new_1$X2 <- factor(new_1$X2, levels = c(T,F), labels = c("Damped","N"))
new_1 <- unite(new_1, "ETS_Model", c("X1","X2"))
colnames(new_1) <- c("ETS_Model", "MASE", "AIC", "BIC")

accuracy <- arrange(new_1, MASE)

accuracy

##    ETS_Model      MASE      AIC      BIC
## 1 MAM_Damped 0.7054746 1041.467 1091.339
## 2 MMM_Damped 0.7066792 1044.393 1094.266
## 3 MAA_Damped 0.7109471 1043.327 1093.199
## 4      MMM_N 0.7129231 1040.336 1087.437
## 5      MAM_N 0.7193936 1039.421 1086.523
## 6 AAA_Damped 0.7196122 1092.572 1142.445
## 7      MAA_N 0.7379681 1044.805 1091.906
## 8      AAA_N 0.7435300 1091.411 1138.513

new_1 <- ets(mort_monthly_ts,model = "ANN")
new_2 <- ets(mort_monthly_ts,model = "AAN")
new_3 <- ets(mort_monthly_ts,model = "AAN", damped = TRUE)
new_4 <- ets(mort_monthly_ts,model = "AAA")
new_5 <- ets(mort_monthly_ts,model = "AAA", damped = TRUE)
new_6 <- ets(mort_monthly_ts, model="MNN", beta = 0.0001)
new_7 <- ets(mort_monthly_ts, model="MAN", damped = TRUE)
new_8 <- ets(mort_monthly_ts, model="MMM", damped = TRUE)
new_9 <- ets(mort_monthly_ts, model="MAM", damped = TRUE)

summary(new_1)

## ETS(A,N,N)
##
## Call:
##  ets(y = mort_monthly_ts, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.7823
##
##   Initial states:
##     l = 46.1443
##
##   sigma:  11.2618
##
##      AIC     AICc      BIC
## 1138.377 1138.588 1146.689
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE     MASE
## Training set -0.3312389 11.16592 7.613011 -6.262377 21.67542 1.063312
##                    ACF1
## Training set 0.06051682
```

```r
checkresiduals(new_1)
```



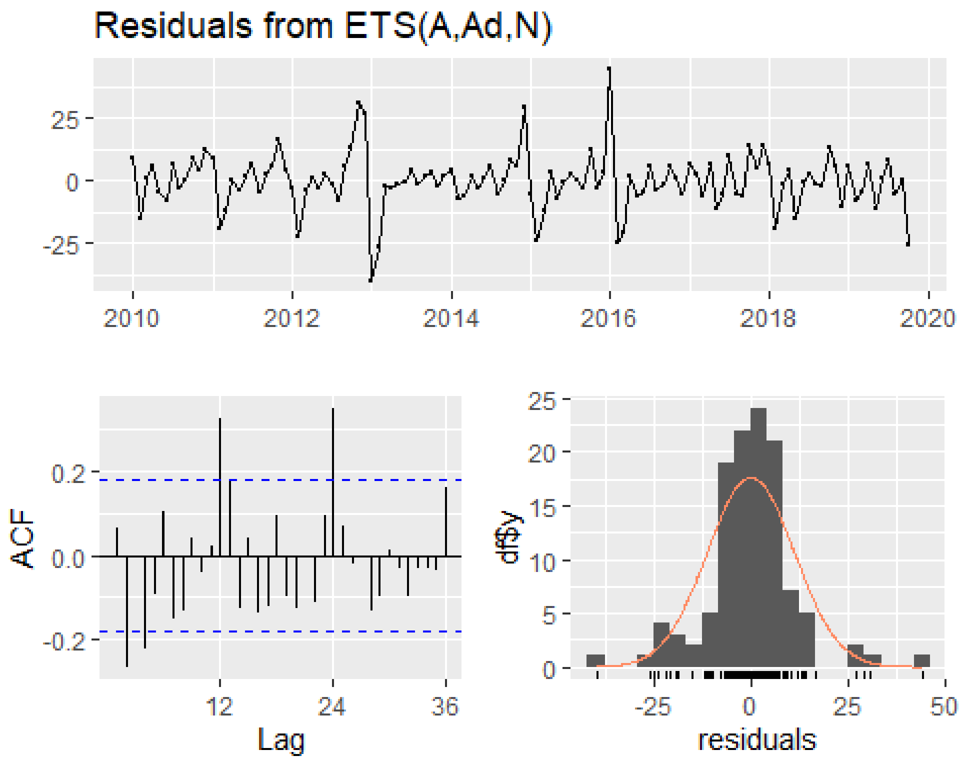
```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,N)
## Q* = 75.903, df = 24, p-value = 2.697e-07
##
## Model df: 0.   Total lags used: 24
```

```r
summary(new_2)
```

```
## ETS(A,Ad,N)
##
## Call:
##  ets(y = mort_monthly_ts, model = "AAN")
##
##   Smoothing parameters:
##     alpha = 0.7689
##     beta  = 1e-04
##     phi   = 0.9589
##
##   Initial states:
##     l = 40.943
##     b = -0.572
##
##   sigma:  11.4204
##
```

```
##      AIC     AICc      BIC 
## 1144.586 1145.343 1161.210 
## 
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE     MASE
## Training set -0.1301646 11.17581 7.666093 -5.766419 21.73666 1.070726
##                    ACF1
## Training set 0.06400021
```

```
checkresiduals(new_2)
```

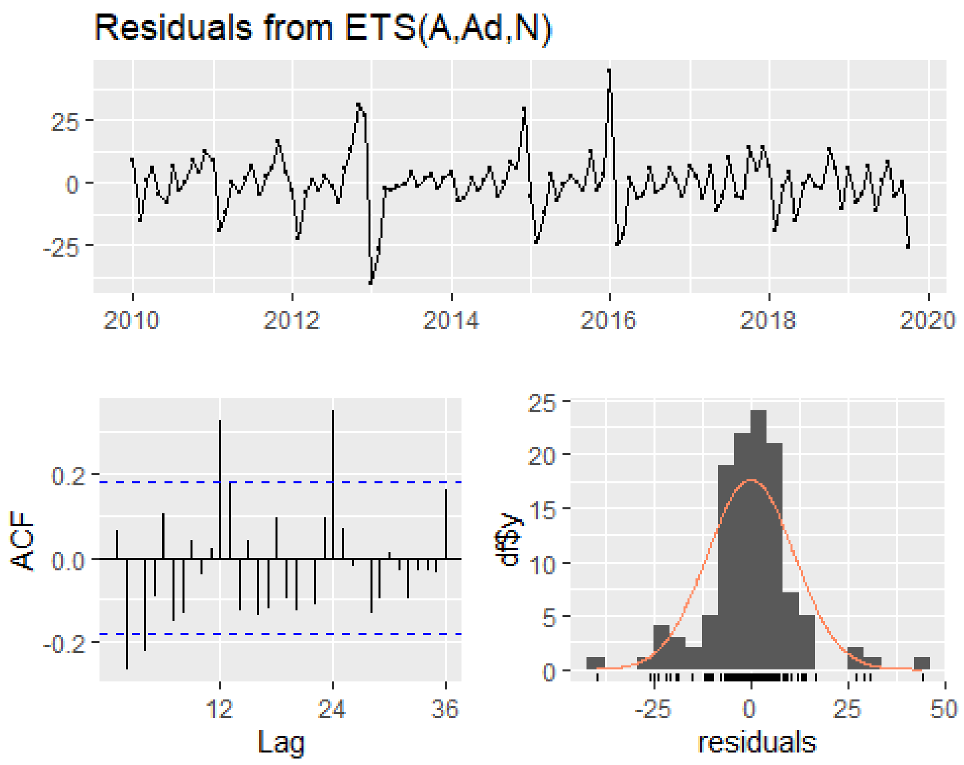


```
## 
##  Ljung-Box test
## 
## data:  Residuals from ETS(A,Ad,N)
## Q* = 75.353, df = 24, p-value = 3.287e-07
## 
## Model df: 0.   Total lags used: 24
```

```
summary(new_3)
```

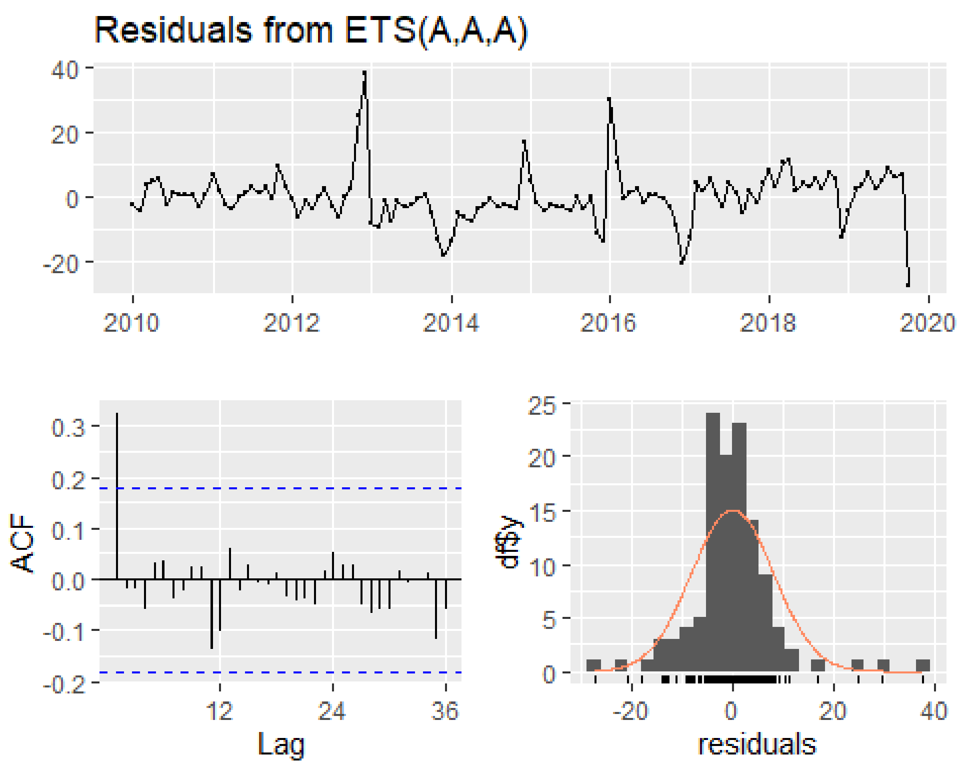
```
## ETS(A,Ad,N) 
## 
## Call:
##  ets(y = mort_monthly_ts, model = "AAN", damped = TRUE) 
## 
##   Smoothing parameters:
##     alpha = 0.7689
##     beta  = 1e-04
##     phi   = 0.9589
##
##   Initial states:
##     l = 40.943
##     b = -0.572
##
##   sigma:  11.4204
##
##      AIC     AICc      BIC
## 1144.586 1145.343 1161.210
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE     MASE
## Training set -0.1301646 11.17581 7.666093 -5.766419 21.73666 1.070726
##                    ACF1
## Training set 0.06400021
```

```
checkresiduals(new_3)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,Ad,N)
## Q* = 75.353, df = 24, p-value = 3.287e-07
##
## Model df: 0.   Total lags used: 24
```

```
summary(new_4)

## ETS(A,A,A)
##
## Call:
##  ets(y = mort_monthly_ts, model = "AAA")
##
##   Smoothing parameters:
##     alpha = 1e-04
##     beta  = 1e-04
##     gamma = 1e-04
##
##   Initial states:
##     l = 37.6703
##     b = -0.0204
##     s = 14.7116 7.7807 2.5525 -6.3303 -7.384 -3.0706
##            -7.1237 -8.1961 -0.8828 -5.3273 -1.4008 14.6709
##
##   sigma:  8.7414
##
##      AIC     AICc      BIC
## 1091.411 1097.531 1138.513
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE    MASE      ACF1
## Training set -0.1315099 8.127185 5.323464 -4.883535 15.59896 0.74353 0.3244654

checkresiduals(new_4)
```



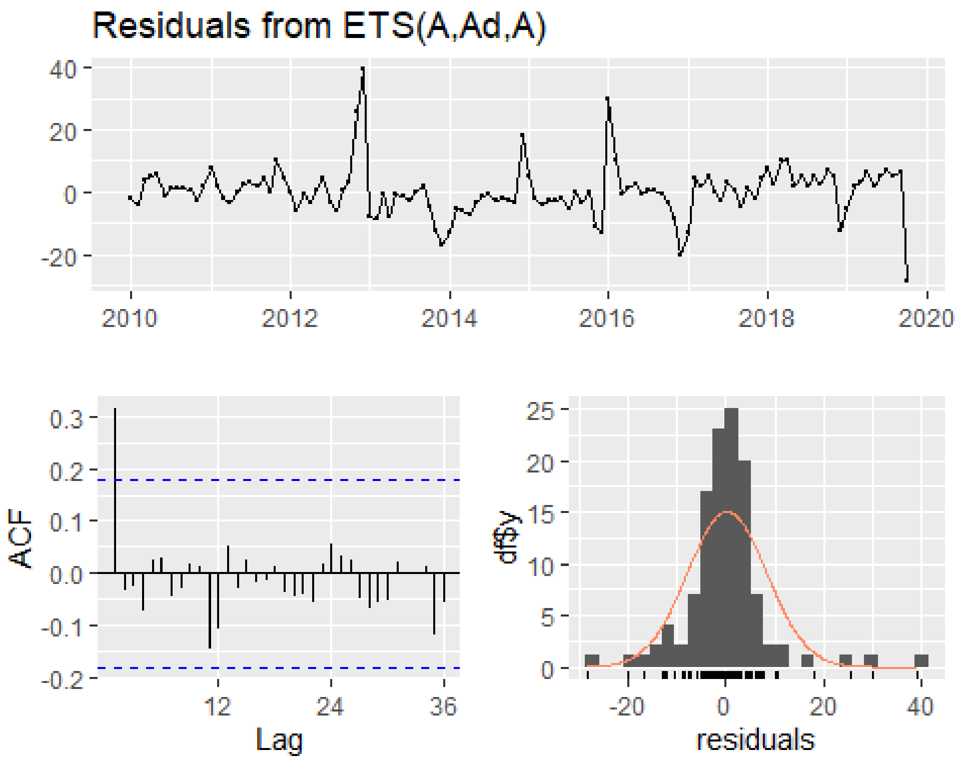
```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,A,A)
## Q* = 20.033, df = 24, p-value = 0.6949
##
## Model df: 0.   Total lags used: 24
```

```
summary(new_5)
```

```
## ETS(A,Ad,A)
##
## Call:
##  ets(y = mort_monthly_ts, model = "AAA", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 1e-04
##     beta  = 1e-04
##     gamma = 1e-04
##     phi   = 0.969
##
##   Initial states:
##     l = 37.4387
##     b = -0.0588
##     s = 14.027 7.7066 2.8124 -6.5881 -7.2838 -2.3644
##            -8.0405 -8.0556 -0.5173 -5.1942 -1.2036 14.7016
##
```

```
##   sigma:  8.7534
##
##      AIC     AICc      BIC
## 1092.572 1099.482 1142.445
##
## Training set error measures:
##                     ME     RMSE     MAE       MPE     MAPE      MASE      ACF1
## Training set 0.1807267 8.098346 5.15222 -3.974345 14.97567 0.7196122 0.3147791
```

```
checkresiduals(new_5)
```



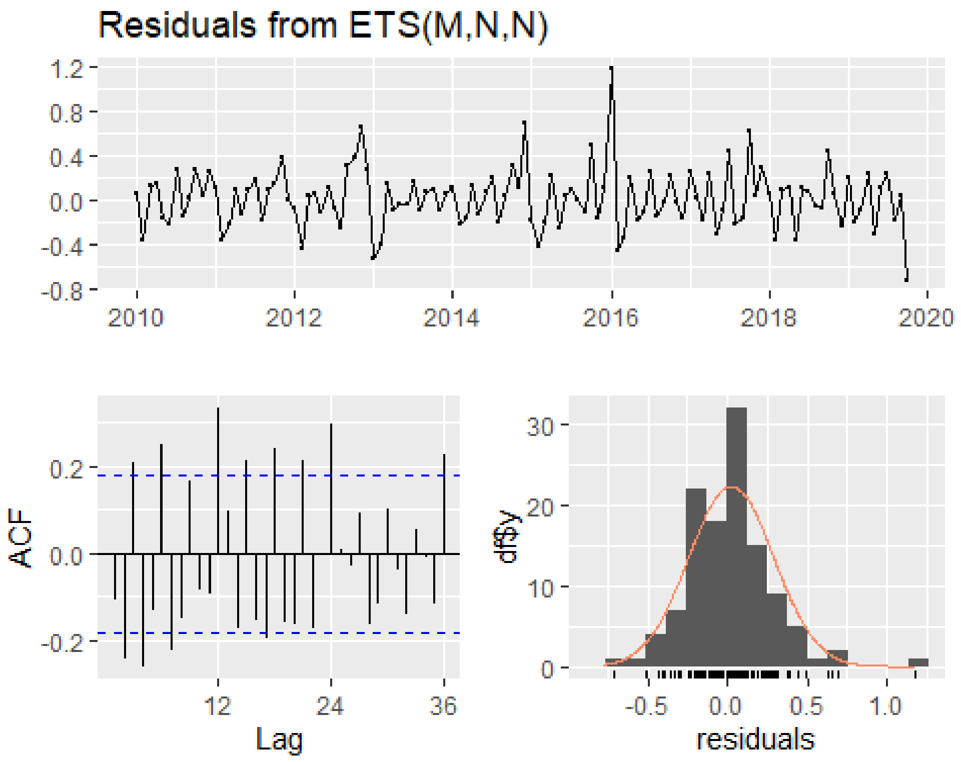
```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,Ad,A)
## Q* = 20.323, df = 24, p-value = 0.6783
##
## Model df: 0.   Total lags used: 24
```

```
summary(new_6)
```

```
## ETS(M,N,N)
##
## Call:
##  ets(y = mort_monthly_ts, model = "MNN", beta = 1e-04)
##
##   Smoothing parameters:
##     alpha = 0.9999
```

```
##
##   Initial states:
##     l = 46.7121
##
##   sigma:  0.2693
##
##      AIC     AICc      BIC
## 1096.314 1096.525 1104.626
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE     MASE      ACF1
## Training set -0.3101969 11.26928 7.802938 -5.542614 22.27066 1.089839 -0.084141
```

```
checkresiduals(new_6)
```



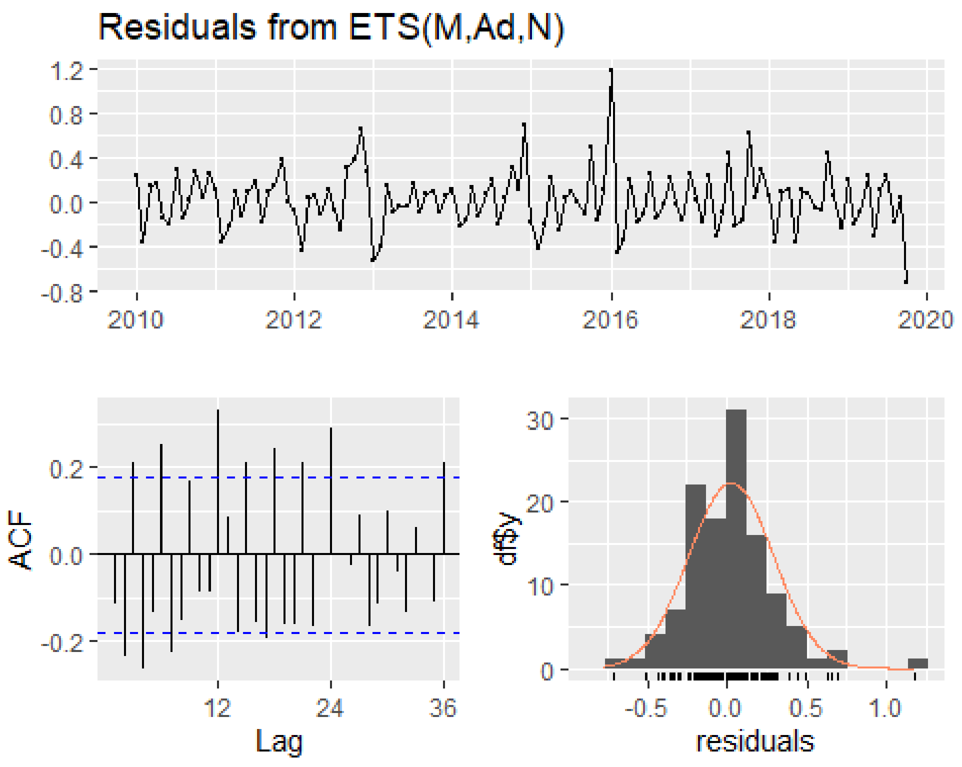
```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,N)
## Q* = 122.09, df = 24, p-value = 4.108e-15
##
## Model df: 0.   Total lags used: 24
```

```
summary(new_7)
```

```
## ETS(M,Ad,N)
##
## Call:
```

```
## ets(y = mort_monthly_ts, model = "MAN", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.9976
##     beta  = 1e-04
##     phi   = 0.8
##
##   Initial states:
##     l = 41.305
##     b = -1.3452
##
##   sigma:  0.2727
##
##      AIC     AICc      BIC
## 1102.665 1103.422 1119.289
##
## Training set error measures:
##                     ME     RMSE      MAE      MPE     MAPE     MASE       ACF1
## Training set -0.218779 11.28964 7.857587 -5.33363 22.36943 1.097472 -0.0895632
```

```r
checkresiduals(new_7)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,Ad,N)
## Q* = 121.95, df = 24, p-value = 4.33e-15
```

```
##
## Model df: 0.   Total lags used: 24
```

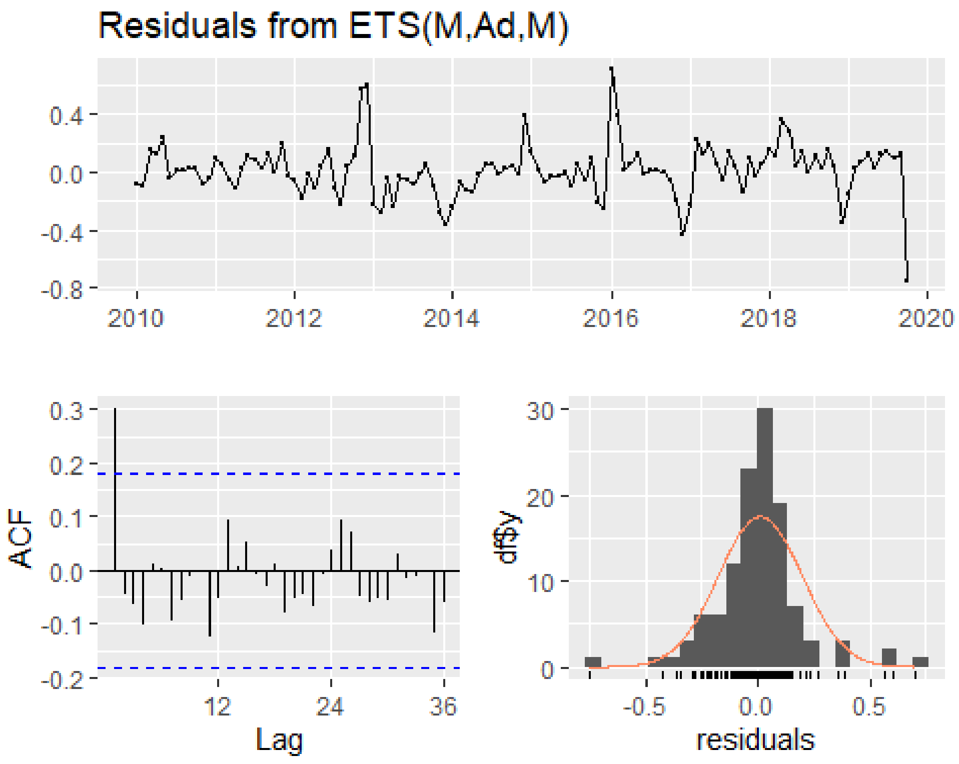
```
auto_ets <- ets(mort_monthly_ts)
summary(auto_ets)
```

```
## ETS(M,N,M)
##
## Call:
##  ets(y = mort_monthly_ts)
##
##   Smoothing parameters:
##     alpha = 0.0631
##     gamma = 1e-04
##
##   Initial states:
##     l = 38.9759
##     s = 1.521 1.2148 1.0169 0.7986 0.7847 0.9026
##            0.7496 0.748 0.9589 0.8438 0.9477 1.5133
##
##   sigma:  0.1958
##
##      AIC     AICc      BIC
## 1035.920 1040.626 1077.480
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.5774302 8.406571 5.279428 -5.382239 15.44174 0.7373794
##                   ACF1
## Training set 0.2995454
```

```
summary(new_9)
```

```
## ETS(M,Ad,M)
##
## Call:
##  ets(y = mort_monthly_ts, model = "MAM", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.0662
##     beta  = 2e-04
##     gamma = 1e-04
##     phi   = 0.9733
##
##   Initial states:
##     l = 37.8894
##     b = -0.1066
##     s = 1.4765 1.2143 1.0418 0.8094 0.8006 0.9295
##            0.7678 0.7573 0.9828 0.8408 0.9469 1.4324
##
##   sigma:  0.2014
##
```

```
##      AIC     AICc      BIC
## 1041.467 1048.376 1091.339
##
## Training set error measures:
##                    ME    RMSE      MAE       MPE     MAPE      MASE      ACF1
## Training set 0.180545 8.22399 5.050999 -3.690427 14.68262 0.7054746 0.2812261

checkresiduals(new_9)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,Ad,M)
## Q* = 21.35, df = 24, p-value = 0.618
##
## Model df: 0.   Total lags used: 24
```

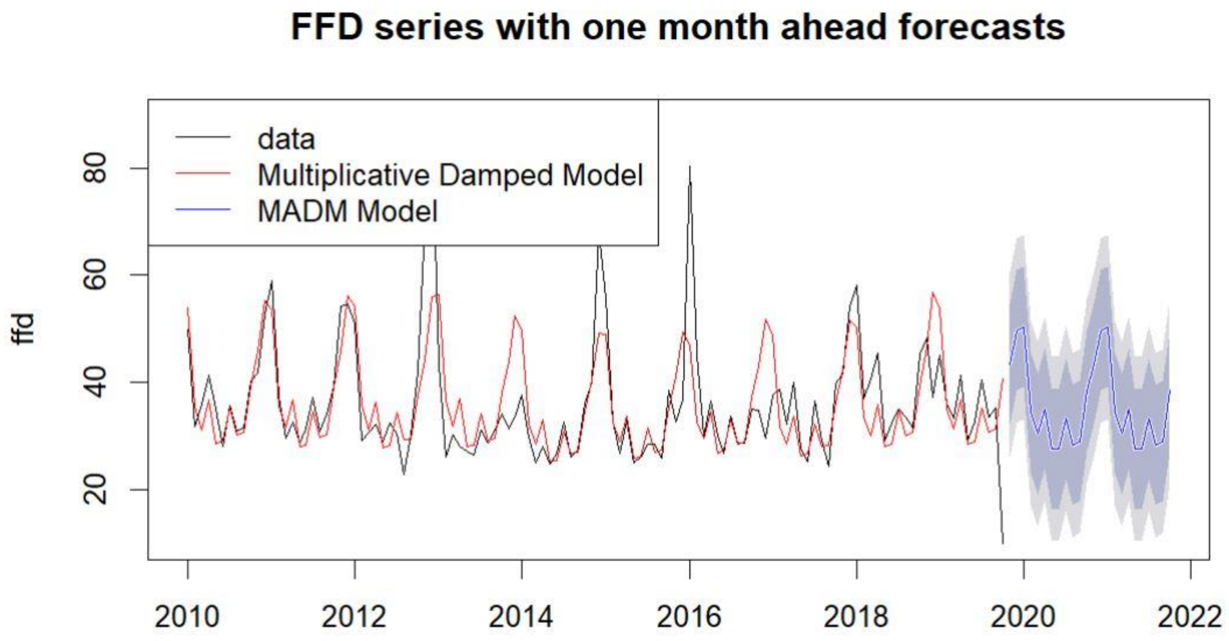
- The automated model fitting process suggests employing the ETS(M,N,M) model configuration. This recommendation is given because, in this setup, the exact values for smoothing and seasonality factors need to be estimated or determined based on the distinct features of the time series data being examined.

- Considering all the models from the State space model, it can be concluded that ETS(M,Ad,M) that is the model new_9 has the lowest MASE of all and outperformed the MNM model in this scenario

- Observing the summary statistics, it can be demonstrated that series is non-stationary but has the lowest MASE values among all the state space models. The residuals show signs of more randomness compared to the previous ones, ACF plot shows presence of a significant lag that demonstrates the presence of serial correlation and no seasonal effects. The histogram is normally distributed.

Considering all the models “new_9” is considered to be the best model with the lowest MASE values. Lets forecast with the new new_9 model.

```
plot(model_hw_1_forecast, type="l", main = "FFD series with one month ahead forecasts", ylab="ffd",
xlab="Year", fcol="white", plot.conf=FALSE)

lines(model_hw_1_forecast$mean, col="blue", type="l")

lines(fitted(new_9), col="red", lty=1)

lines(new_9$mean, col="red", type="l")

legend("topleft", lty=1, col=c("black","red","blue"),

c("data","Multiplicative Damped Model","MNM” Model"))
```



Conclusion

In the analysis through diverse modeling techniques for five distinct mortality series, the exponential damped model stood out as the best fit for monthly data, exhibiting superior performance in terms of R², AIC, BIC, and MASE. Meanwhile, for the weekly series, the dynamic linear model provided precise prediction.

Task 2

Data Description

This report explores the modeling and prediction of the initial flowering day (FFD) for a specific plant species in Australia, considering various climate factors such as rainfall, temperature, radiation, and relative humidity. The study covers a 31-year period from 1984 to 2014, with the goal of understanding how these climate variables affect the timing of FFD. Through the use of single-variable models, we investigate the relationship between each climate factor and FFD, allowing us to make forecasts about when this plant species will start flowering, with a particular emphasis on a four-year projection.

```
getwd()
```

```
## [1] "F:/2nd semester/Forecasting/Final_assignment"
```

```
setwd("F:/2nd semester/Forecasting/Final_assignment")
covariate <- read.csv("Covariate.csv")

ffd <- read.csv("FFD.csv")

ffd %>% is.na() %>% sum()
```

```
## [1] 0
```

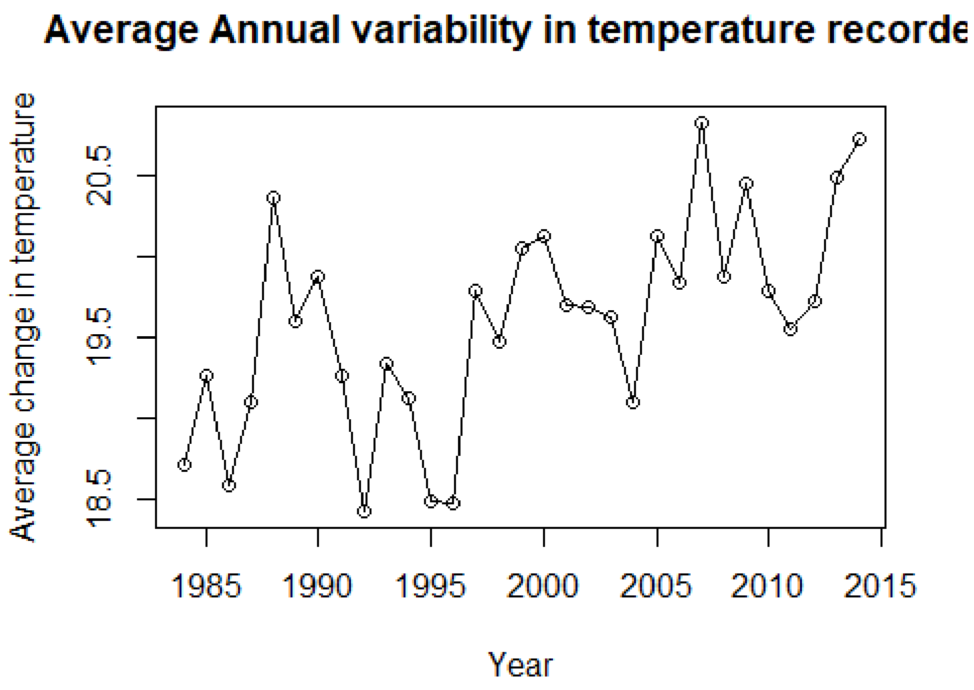
```
covariate %>% is.na() %>% sum()
```

```
## [1] 10
```

```
new_covariate <- na.omit(covariate)
new_covariate
```

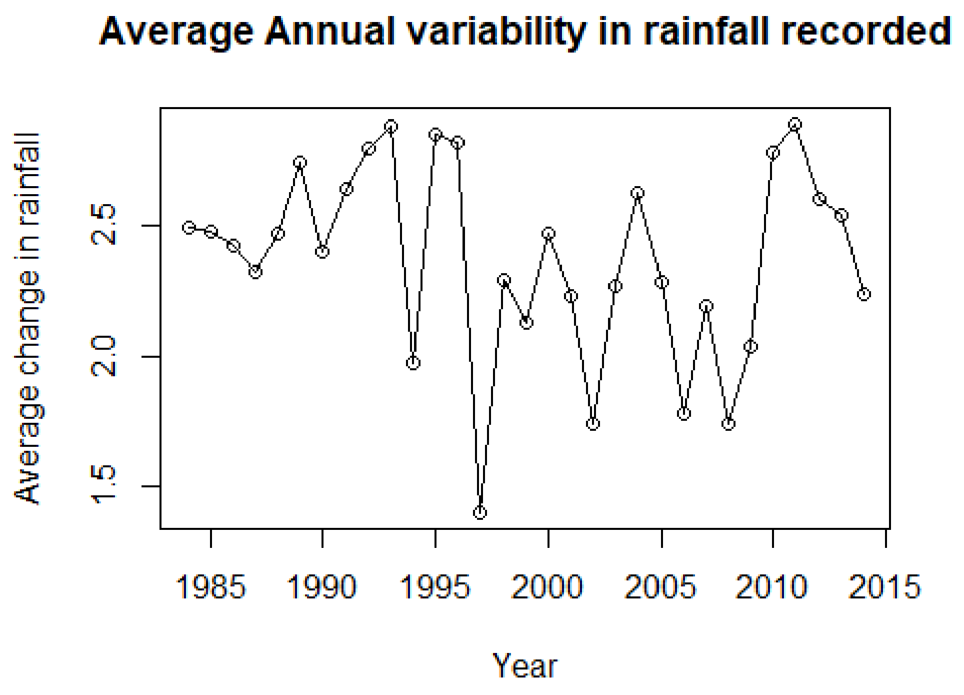
```
##   Year Temperature Rainfall Radiation RelHumidity
## 1 2015       20.74     2.27     14.60       94.45
## 2 2016       20.49     2.38     14.56       94.03
## 3 2017       20.52     2.26     14.79       95.04
## 4 2018       20.56     2.27     14.79       95.06
```

```
ffd_ts <- ts(ffd[,2:6], start = c(1984,1), frequency = 1)
temp <- ts(ffd$Temperature, start = c(1984,1) , frequency = 1)
rain <- ts(ffd$Rainfall, start = c(1984,1) , frequency = 1)
rad <- ts(ffd$Radiation, start = c(1984,1), frequency = 1)
hum <- ts(ffd$RelHumidity, start = c(1984,1), frequency = 1)
ffd <- ts(ffd$FFD, start = c(1984,1), frequency = 1)

par(mfrow=c(1,1))
plot(temp, ylab="Average change in temperature", xlab = "Year", main =
"Average Annual variability in temperature recorded", type ="o")
```



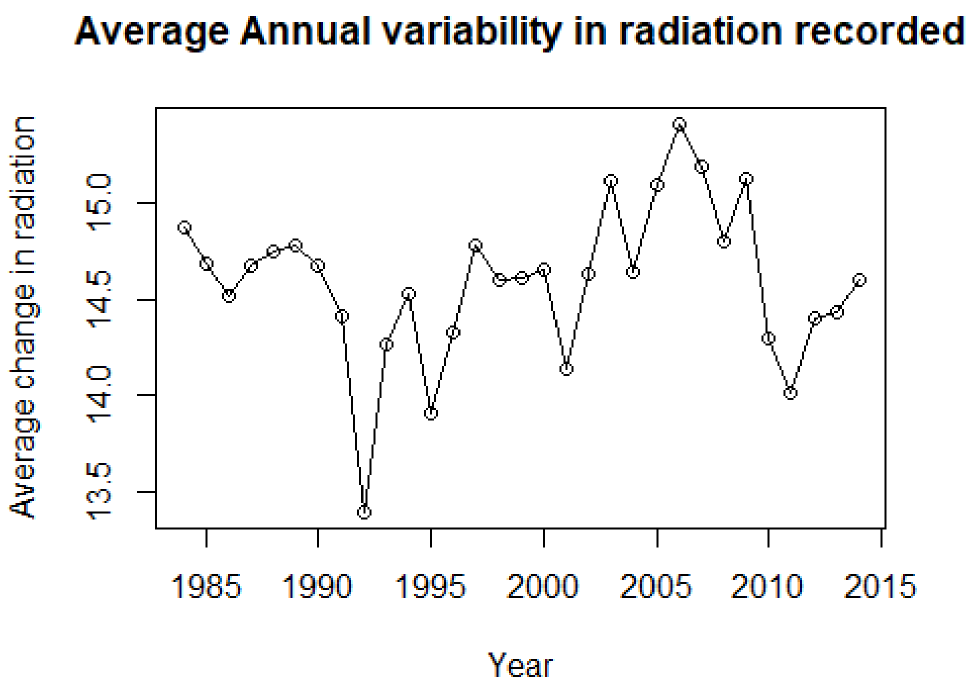
* The time series data shows no clear seasonal pattern with evidence of upward trend. The change of variance exist. However, there is a no notable intervention.

```
par(mfrow=c(1,1))
plot(rain, ylab="Average change in rainfall", xlab = "Year", main =
"Average Annual variability in rainfall recorded", type ="o")
```



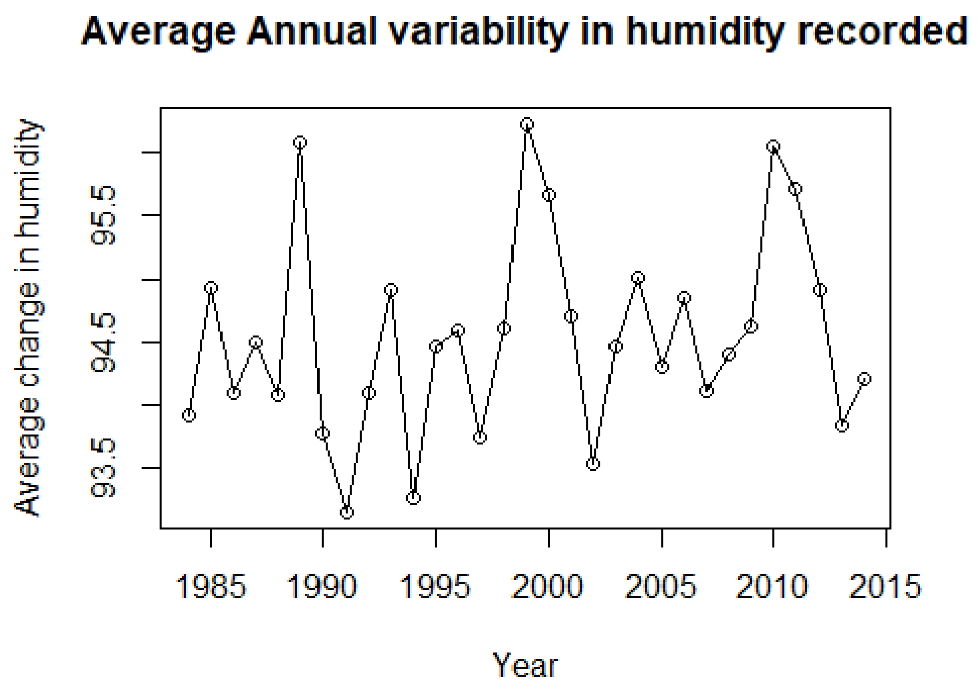
- The time series data shows no clear seasonal pattern with no evidence of trend or change of variance . However, there is a notable intervention in the year 1997, suggesting a significant external effect influenced the data

```
par(mfrow=c(1,1))
plot(rad, ylab="Average change in radiation", xlab = "Year", main =
"Average Annual variability in radiation recorded", type ="o")
```



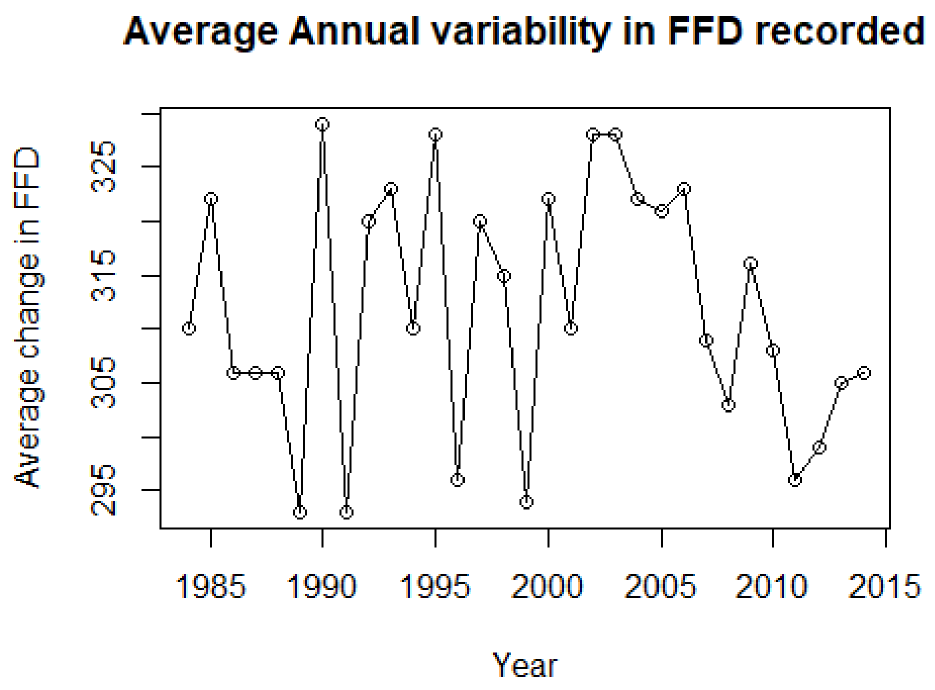
* The time series data shows no clear seasonal pattern with no evident trend but little change of variance 2000 onward. However, there is a notable intervention between the year 1992-193, suggesting a significant external effect influenced the data.

```
par(mfrow=c(1,1))
plot(hum, ylab="Average change in humidity", xlab = "Year", main =
"Average Annual variability in humidity recorded", type ="o")
```



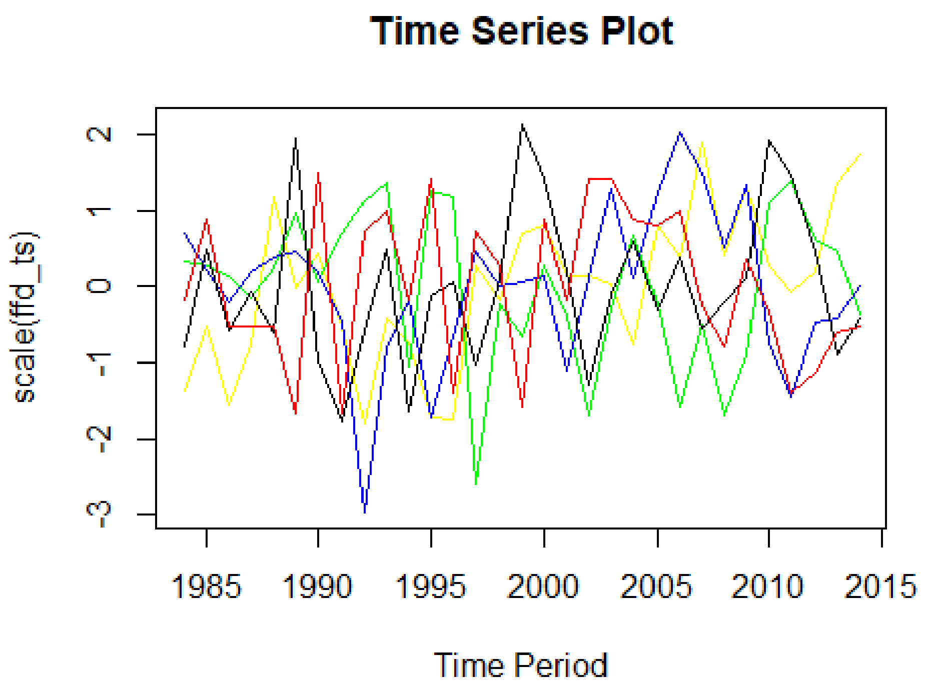
* The time series data shows no clear seasonal pattern with no evident trend or change of variance . However, there is a notable intervention between the year 1990, 2002 and 2013, suggesting a significant external effect influenced the data.

```
par(mfrow=c(1,1))
plot(ffd, ylab="Average change in FFD", xlab = "Year", main =
"Average Annual variability in FFD recorded", type ="o")
```



* The time series data shows no clear seasonal pattern with no evident trend or change of variance . However, there is a notable intervention starting from the year 1989 till 1999, suggesting a significant external effect influenced the data.

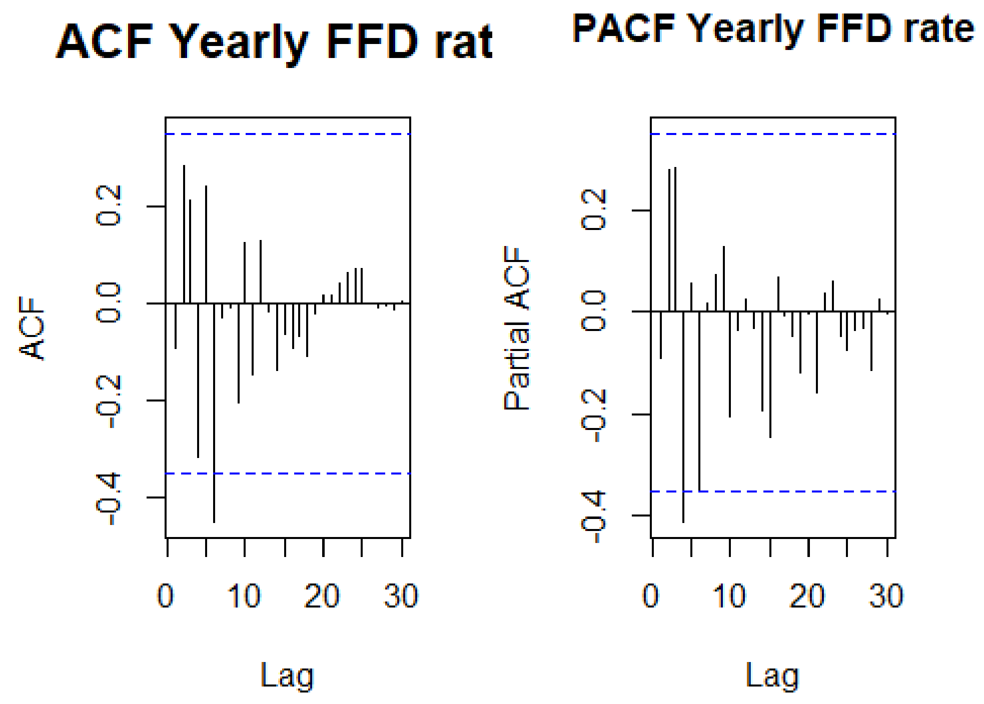
```
plot(scale(ffd_ts),plot.type = "s", col = c("yellow","green","blue","black","red"),main = "Time Series Plot", xlab = "Time Period")
```



* Looking at the all the time series plots it all the series are somewhat correlated.

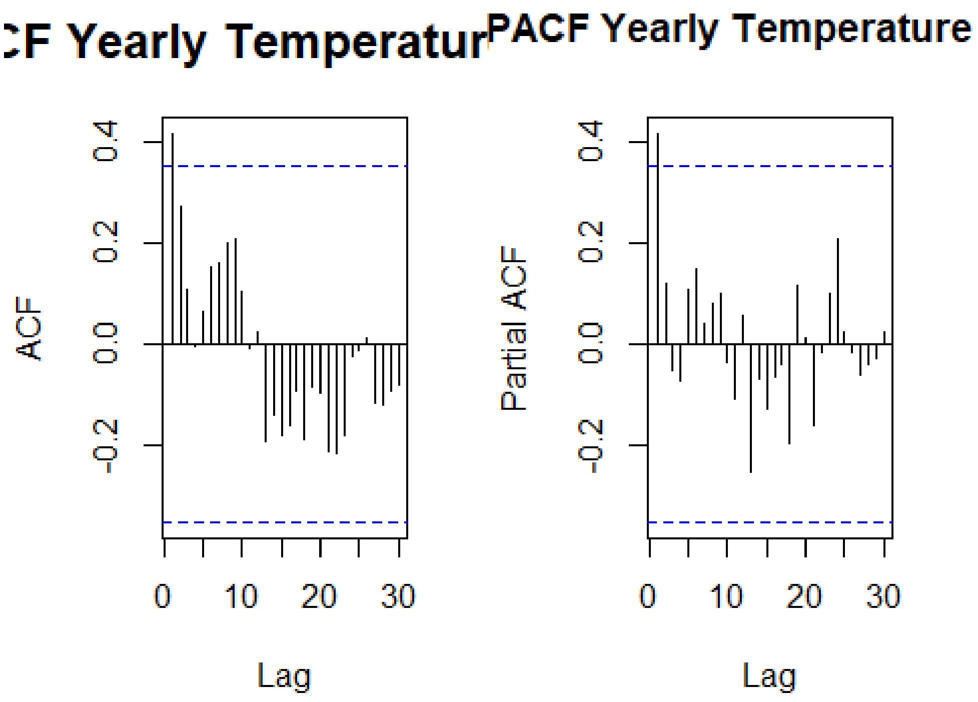
Analysing Non-Stationarity in dataset

```
par(mfrow=c(1,2))
acf(ffd, lag.max = 48, main = "ACF Yearly FFD rate", cex.main = 1.5)
pacf(ffd, lag.max = 48, main = "PACF Yearly FFD rate", cex.main = 1.5)
```



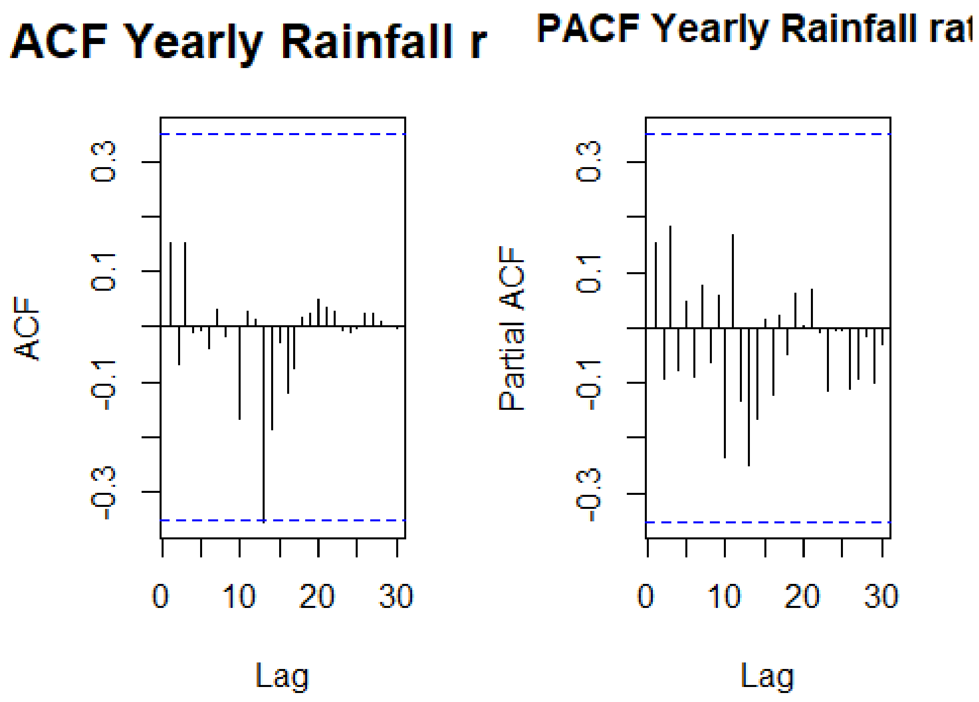
* The ACF plot shows signs of declining pattern, which infer that there is a trend in the series trend in the time series data that will dominate the series correlation. There is no sign of a seasonal pattern, and all the lag values are well below the 95% confidence interval except one. On the other hand, when looking at the PACF plot, there is only one lag that appears highly significant, suggesting that the series is not stationary.

```
par(mfrow=c(1,2))
acf(temp, lag.max = 48, main = "ACF Yearly Temperature rate", cex.main = 1.5)
pacf(temp, lag.max = 48, main = "PACF Yearly Temperature rate", cex.main = 1.5)
```



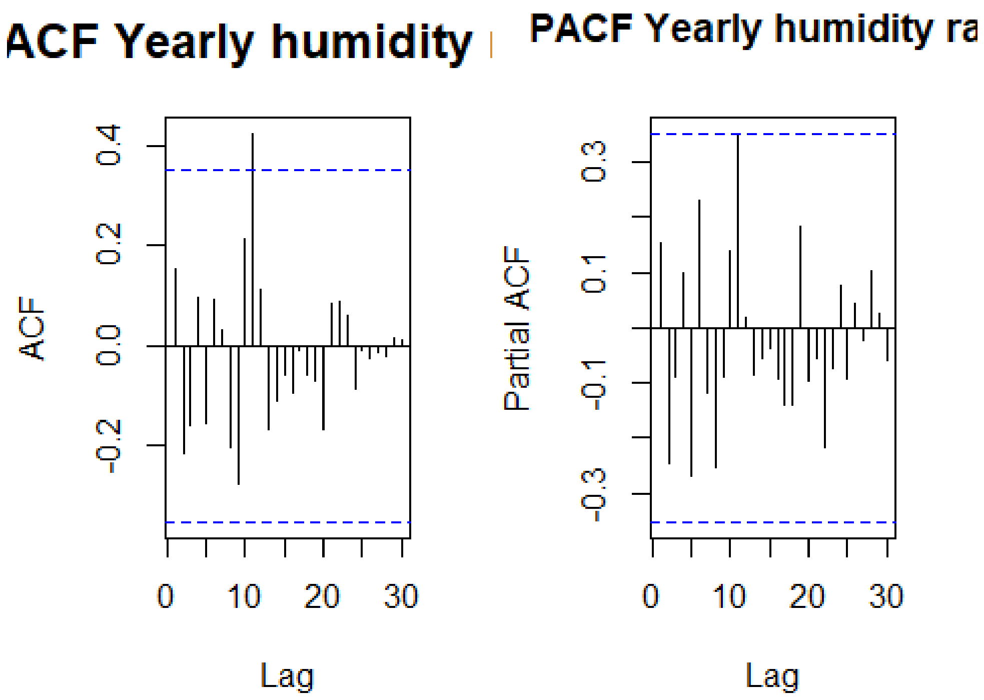
 * The ACF plot shows signs of declining pattern, which infer that there is a trend in the series trend in the time series data that will dominate the series correlation. All the lag values are well below the 95% confidence interval except the first lag. On the other hand, when looking at the PACF plot, there is only one lag that appears highly significant, suggesting that the series is not stationary.

```
par(mfrow=c(1,2))
acf(rain, lag.max = 48, main = "ACF Yearly Rainfall rate", cex.main = 1.5)
pacf(rain, lag.max = 48, main = "PACF Yearly Rainfall rate", cex.main = 1.5)
```



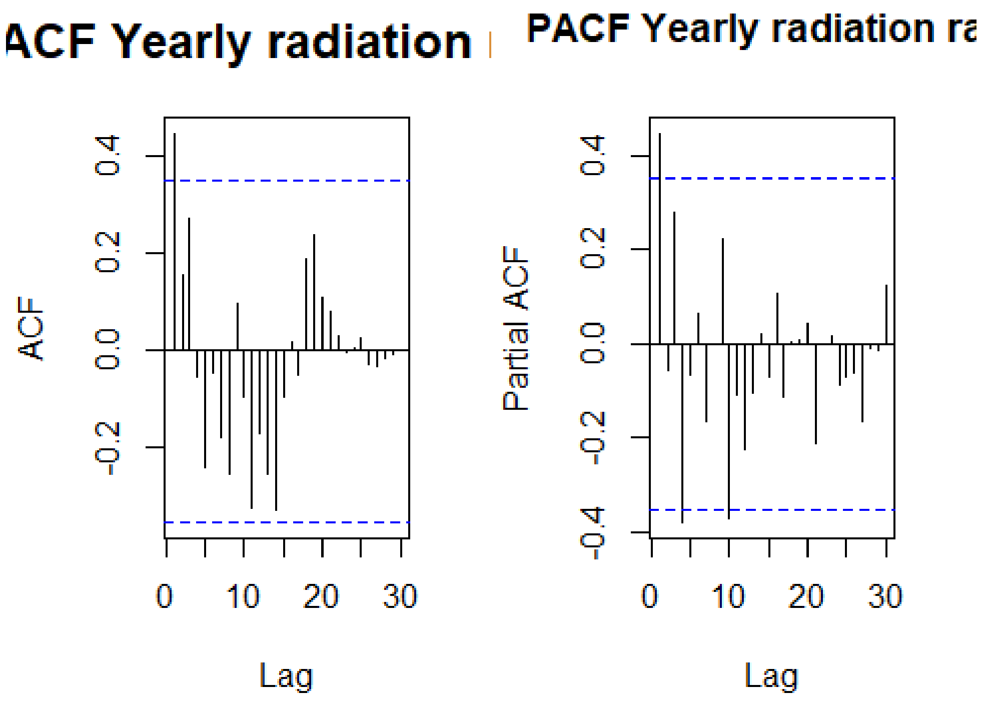
* The ACF plot shows no signs of declining pattern, which infer that there is no trend in the series trend in the time series data. All the lag values are well below the 95% confidence interval.On the other hand, when looking at the PACF plot, there is no lag that appears highly significant, suggesting that the series is stationary

```
par(mfrow=c(1,2))
acf(hum, lag.max = 48, main = "ACF Yearly humidity rate", cex.main = 1.5)
pacf(hum, lag.max = 48, main = "PACF Yearly humidity rate", cex.main = 1.5)
```



* The ACF plot shows signs of inclining and declining pattern, which infer that there is trend in the series trend in the time series data that will dominate the series correlation. All the lag values are well below the 95% confidence interval except the first lag. On the other hand, when looking at the PACF plot, there is no lag that appears highly significant, suggesting that the series is stationary.

```
par(mfrow=c(1,2))
acf(rad, lag.max = 48, main = "ACF Yearly radiation rate", cex.main = 1.5)
pacf(rad, lag.max = 48, main = "PACF Yearly radiation rate", cex.main = 1.5)
```



* The ACF plot shows signs of declining and inclining pattern, which infer that there is trend in the series trend in the time series data that will dominate the series correlation. All the lag values are well below the 95% confidence interval except the first lag. On the other hand, when looking at the PACF plot, there is are lags that appears highly significant, suggesting that the series is not stationary and does exhibit autocorrelation.

```
k <- ar(ffd)$order
adf.test(ffd, k = k)

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

 * In this situation, with a lag order of 6 and a p-value of 0.5977, we cannot reject the null hypothesis. This essentially means that the time series is not stationary. This demonstrates the time-dependent patterns and its variance is not uniform over time.

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

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

br> * In this situation, with a lag order of 1 and a p-value of 0.1902, we cannot reject the null hypothesis. This essentially means that all the time series are not stationary. This demonstrates the time-dependent patterns and its variance is not uniform over time.

```
k <- ar(rad)$order
adf.test(rad, k = k)

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

br> * In this situation, with a lag order of 4 and a p-value of 0.2911, we cannot reject the null hypothesis. This essentially means that all the time series are not stationary. This demonstrates the time-dependent patterns and its variance is not uniform over time

```
k <- ar(hum)$order
adf.test(hum, k = k)

## Warning in adf.test(hum, k = k): p-value smaller than printed p-value

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

br> * In this situation, with a lag order of 0 and a p-value of 0.01, we can reject the null hypothesis. This essentially means that all the time series are stationary. This demonstrates no time-dependent patterns and its variance is uniform over time

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

## Warning in adf.test(rain, k = k): p-value smaller than printed p-value

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

br> * In this situation, with a lag order of 0 and a p-value of 0.01, we can reject the null hypothesis. This essentially means that all the time series are stationary. This demonstrates no time-dependent patterns and its variance is uniform over time.

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

## Warning in adf.test(rain, k = k): p-value smaller than printed p-value

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

* In this situation, with a lag order of 0 and a p-value of 0.01, we can reject the null hypothesis. This essentially means that all the time series are stationary. This demonstrates no time-dependent patterns and its variance is uniform over time.

*It’s clear that the FFD, Temperature and radiation isn’t stationary. To remedy this, our first step is to handle non-stationarity, followed by addressing seasonality. This sequence aims to reduce the strong serial correlation within the series. The initial approach involves using the Box-Cox transformation to assess its impact on mitigating non-stationary behavior.

```
lambda_hum <- BoxCox.lambda(hum, method = "loglik")
lambda_rad <- BoxCox.lambda(rad, method = "loglik")
lambda_ffd <- BoxCox.lambda(ffd, method = "loglik")

rad_boxcox <- BoxCox(rad,lambda = lambda_rad )

ffd_boxcox <- BoxCox(ffd,lambda = lambda_ffd )

hum_boxcox <- BoxCox(ffd,lambda = lambda_hum )

adf.test(rad_boxcox)

##
##  Augmented Dickey-Fuller Test
##
## data:  rad_boxcox
## Dickey-Fuller = -2.7133, Lag order = 3, p-value = 0.2981
## alternative hypothesis: stationary

adf.test(ffd_boxcox)

##
##  Augmented Dickey-Fuller Test
##
## data:  ffd_boxcox
## Dickey-Fuller = -2.3313, Lag order = 3, p-value = 0.4452
## alternative hypothesis: stationary

adf.test(hum_boxcox)

##
##  Augmented Dickey-Fuller Test
```

```
##
## data:  hum_boxcox
## Dickey-Fuller = -2.3219, Lag order = 3, p-value = 0.4488
## alternative hypothesis: stationary
```

 * Considering the outcome of the ADF test, it is further validated that BoxCox Transformation did not significantly alter the series transformation in terms of its non-stationary in nature.

- Given that the series displays non-stationarity without any indications of seasonality, the next most appropriate course of action is to employ the ordinary differencing operation.

```
hum_boxcox_diff <- diff(hum_boxcox)

ffd_boxcox_diff <- diff(ffd_boxcox)

rad_boxcox_diff <- diff(rad_boxcox)

adf.test(hum_boxcox_diff)

##
##  Augmented Dickey-Fuller Test
##
## data:  hum_boxcox_diff
## Dickey-Fuller = -3.4758, Lag order = 3, p-value = 0.0653
## alternative hypothesis: stationary

adf.test(ffd_boxcox_diff)

##
##  Augmented Dickey-Fuller Test
##
## data:  ffd_boxcox_diff
## Dickey-Fuller = -3.4926, Lag order = 3, p-value = 0.06292
## alternative hypothesis: stationary

adf.test(rad_boxcox_diff)

##
##  Augmented Dickey-Fuller Test
##
## data:  rad_boxcox_diff
## Dickey-Fuller = -2.6703, Lag order = 3, p-value = 0.3152
## alternative hypothesis: stationary
```

 * 1st order difference did not seem to have much effect on stationarity, now lets try the second order difference.

```
hum_boxcox_diff_2 <- diff(hum_boxcox, differences = 2)
ffd_boxcox_diff_2 <- diff(ffd_boxcox, differences = 2)
rad_boxcox_diff_3 <- diff(rad_boxcox, differences = 3)

adf.test(hum_boxcox_diff_2)

## Warning in adf.test(hum_boxcox_diff_2): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  hum_boxcox_diff_2
## Dickey-Fuller = -5.2614, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(ffd_boxcox_diff_2)
```

```
## Warning in adf.test(ffd_boxcox_diff_2): p-value smaller than printed p-value
```

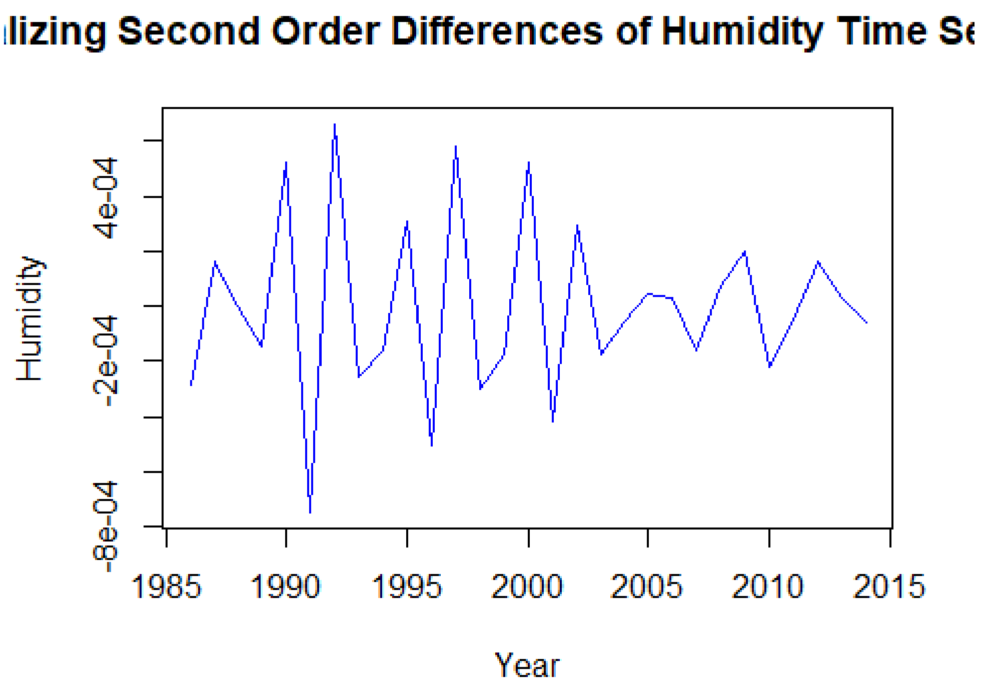
```
##
##  Augmented Dickey-Fuller Test
##
## data:  ffd_boxcox_diff_2
## Dickey-Fuller = -5.1715, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(rad_boxcox_diff_3)
```

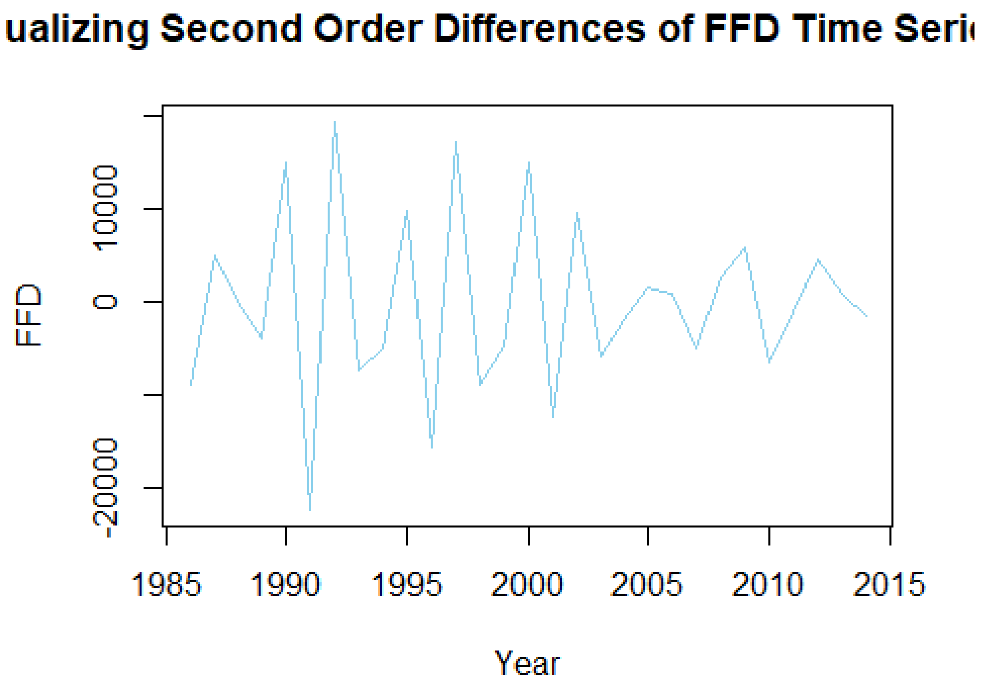
```
##
##  Augmented Dickey-Fuller Test
##
## data:  rad_boxcox_diff_3
## Dickey-Fuller = -4.0888, Lag order = 3, p-value = 0.01966
## alternative hypothesis: stationary
```

 * 2nd order difference seem to have much effect on stationarity, all the p-values are less than 0.05. This indicates, series is stationary.

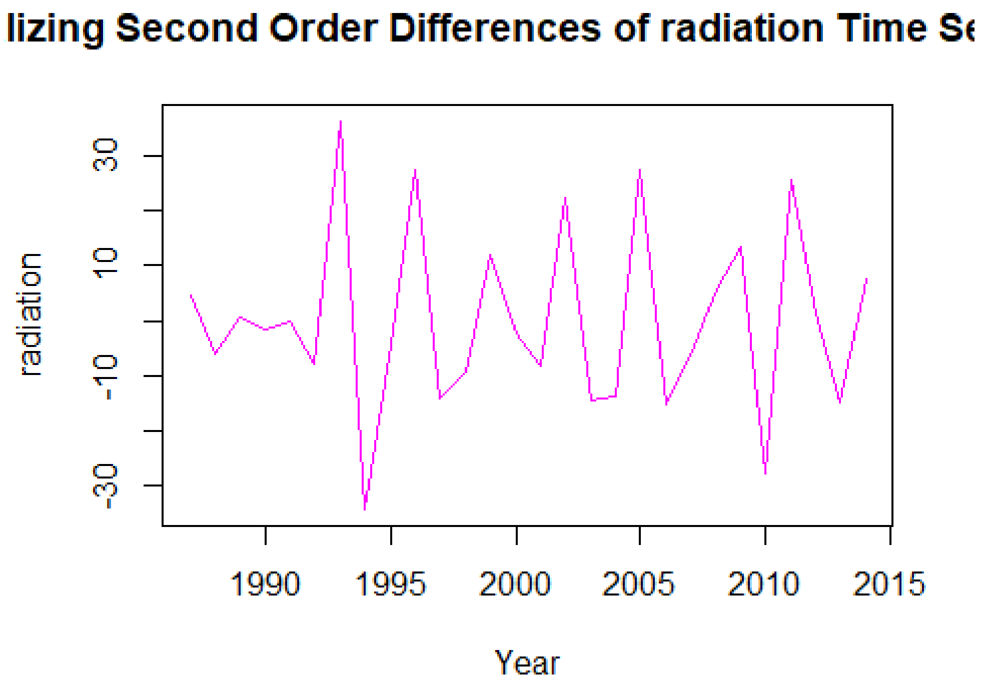
```
plot(hum_boxcox_diff_2, xlab = "Year", main =
"Visualizing Second Order Differences of Humidity Time Series Data", ylab = "Humidity", col = "blue")
```



```
plot(ffd_boxcox_diff_2, xlab = "Year", main =
"Visualizing Second Order Differences of FFD Time Series Data", ylab = "FFD", col = "skyblue")
```



```
plot(rad_boxcox_diff_3,xlab = "Year", main =
"Visualizing Second Order Differences of radiation Time Series Data", ylab = "radiation", col =
"magenta" )
```



* Considering the outcome of the ADF test and upon visualizing the data, after 1st order differencing did not seems to alter the series transformation in terms of its non-stationary in nature. Therefore second order differencing was taken into consideration the humidity and ffd series. In addition, the third order differencing seems to be more appropriate for radiation series to make the series stationary.

Modelling utlizing distributed Lag Models

Finite DLM

```
finiteDLMauto(x=as.vector(rain), 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 0.45542 159.5287 173.1075  1.92366  0.21114  0.35887 0.0119535960
## 9      9 0.57633 176.4695 189.5620  1.84139  0.61507  0.02093 0.0053610928
## 8      8 0.63147 183.4408 195.9313  2.04930  0.60794  0.02715 0.0009126001
## 7      7 0.64652 194.4994 206.2799  1.59045 -3.01761 -0.03019 0.0737059745
## 6      6 0.64294 202.8934 213.8633  1.92186  0.28658 -0.02790 0.5299790980
## 5      5 0.63057 210.6108 220.6755  1.89295  0.25105  0.02469 0.9076494683
## 4      4 0.62065 215.4913 224.5622  1.78164  0.25781  0.07739 0.9075279887
## 3      3 0.61948 220.7923 228.7855  3.63777 -0.76968  0.10634 0.8338787236
## 2      2 0.70509 227.8920 234.7284 10.96964  0.67281  0.07339 0.9694005848
## 1      1 0.73457 236.0701 241.6749 10.62325  0.47857  0.02278 0.6439623951
```

```
finiteDLMauto(x=as.vector(temp), 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
```

```
finiteDLMauto(x=as.vector(rad), 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 0.23190 134.8487 148.4274  0.88540  0.15359  0.80205 0.82991860
## 9      9 0.48289 170.8262 183.9187  1.34489 -0.09150  0.24245 0.05193651
## 8      8 0.63927 185.2035 197.6940  2.07409  0.39351 -0.05034 0.26617631
## 7      7 0.65526 195.3695 207.1500  1.89503  0.73656 -0.06822 0.83503825
## 6      6 0.65506 204.5765 215.5464  1.72456  2.05769 -0.09948 0.72930287
## 5      5 0.64497 212.4087 222.4735  1.76393  0.39153 -0.04515 0.34513069
## 4      4 0.72626 221.5765 230.6474  2.01730  0.68063 -0.15584 0.47571903
## 3      3 0.73563 226.6139 234.6072  4.31501  0.70260 -0.10020 0.43599041
## 2      2 0.75285 231.5493 238.3857 11.76070  0.67585 -0.05116 0.45318101
## 1      1 0.75824 238.2644 243.8691 10.05068  1.08447 -0.05138 0.49071929
```

```
finiteDLMauto(x=as.vector(hum), 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 0.57024 172.2492 185.8280  2.08228  0.06822 -0.17493 0.6161256
## 9      9 0.55036 177.5658 190.6583  1.72123 -0.27940 -0.02909 0.3965249
## 8      8 0.67306 187.0559 199.5463  2.44918 -0.71499 -0.13844 0.7241713
## 7      7 0.69275 199.2327 211.0132  1.48688  0.04426 -0.25478 0.7066725
## 6      6 0.67480 205.9846 216.9545  1.69696  0.59229 -0.16318 0.2418659
## 5      5 0.64039 212.2332 222.2980  1.31974  0.57567 -0.03811 0.1465822
## 4      4 0.70871 219.9171 228.9880  2.30614  0.34181 -0.08694 0.4106677
## 3      3 0.72218 224.7782 232.7714  5.69034  0.62192 -0.03038 0.5055109
## 2      2 0.75086 230.2895 237.1260 12.91953  1.09599 -0.00647 0.5780824
## 1      1 0.75547 237.0601 242.6649 10.62963  0.62485 -0.01001 0.4674482
```

- After conducting a comprehensive analysis that took into account factors like MASE, AIC, and BIC, we have determined that a lag length of 10 is the optimal choice. It is evident that as the lag length increases, there is consistent improvement in these metrics.

Based on this insight, we can conclude that implementing a finite Dynamic Linear Model (DLM) is a favorable decision.

Modelling with intercept

```
model1_finite = dlm(x = as.vector(rain), y = as.vector(ffd),
q =10)

model2_finite = dlm(x = as.vector(hum), y = as.vector(ffd),
q =10)

model2_finite = dlm(x = as.vector(temp), y = as.vector(ffd),
q =10)

model3_finite = dlm(x = as.vector(rad), y = as.vector(ffd),
q =10)
```

Modelling without intercept

```
original_names<- colnames(ffd_ts)
original_names

## [1] "Temperature" "Rainfall"    "Radiation"   "RelHumidity" "FFD"

new_names<- c("Temperature", "Rainfall","Radiation","Humidity","FFD")
colnames(ffd_ts)<- new_names
updated <- colnames(ffd_ts)
updated

## [1] "Temperature" "Rainfall"    "Radiation"   "Humidity"    "FFD"

model4_finite = dlm(formula= FFD~Rainfall-1, data = data.frame(ffd_ts), q = 10)

model5_finite = dlm(formula= FFD~Temperature -1, data = data.frame(ffd_ts), q = 10)

model6_finite = dlm(formula= FFD~Radiation -1, data = data.frame(ffd_ts), q = 10)

model7_finite = dlm(formula= FFD~Humidity -1, data = data.frame(ffd_ts), q = 10)

sort.score <- function(x, score = c("bic", "aic", "mase")) {
  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")')
  }
}
```

```
sort.score(AIC(model1_finite$model,model2_finite$model,
model3_finite$model,model4_finite$model,model5_finite$model,
model6_finite$model,model7_finite$model), score = "aic")

##                     df      AIC
## model3_finite$model 13 134.8487
## model1_finite$model 13 159.5287
## model2_finite$model 13 170.8439
## model6_finite$model 12 174.0916
## model7_finite$model 12 174.1069
## model5_finite$model 12 189.0983
## model4_finite$model 12 202.8882

sort.score(BIC(model1_finite$model,model2_finite$model,
model3_finite$model,model4_finite$model,model5_finite$model,
model6_finite$model,model7_finite$model), score = "bic")

##                     df      BIC
## model3_finite$model 13 148.4275
## model1_finite$model 13 173.1074
## model2_finite$model 13 184.4227
## model6_finite$model 12 186.6259
## model7_finite$model 12 186.6411
## model5_finite$model 12 201.6326
## model4_finite$model 12 215.4225

sort.score(MASE(model1_finite$model,model2_finite$model,
model3_finite$model,model4_finite$model,model5_finite$model,
model6_finite$model,model7_finite$model), score = "mase")

##                      n      MASE
## model3_finite$model 21 0.2319041
## model1_finite$model 21 0.4554174
## model2_finite$model 21 0.5495688
## model6_finite$model 21 0.6007618
## model7_finite$model 21 0.6153355
## model5_finite$model 21 0.9011990
## model4_finite$model 21 1.2496951
```

- Considering the result for AIC,BIC and MASE model3_finite model is the best model. The model with model3_finite as the predictor stands out as the most suitable choice according to AIC and BIC values. Therefore, this model will undergo further analysis for diagnostic checking.

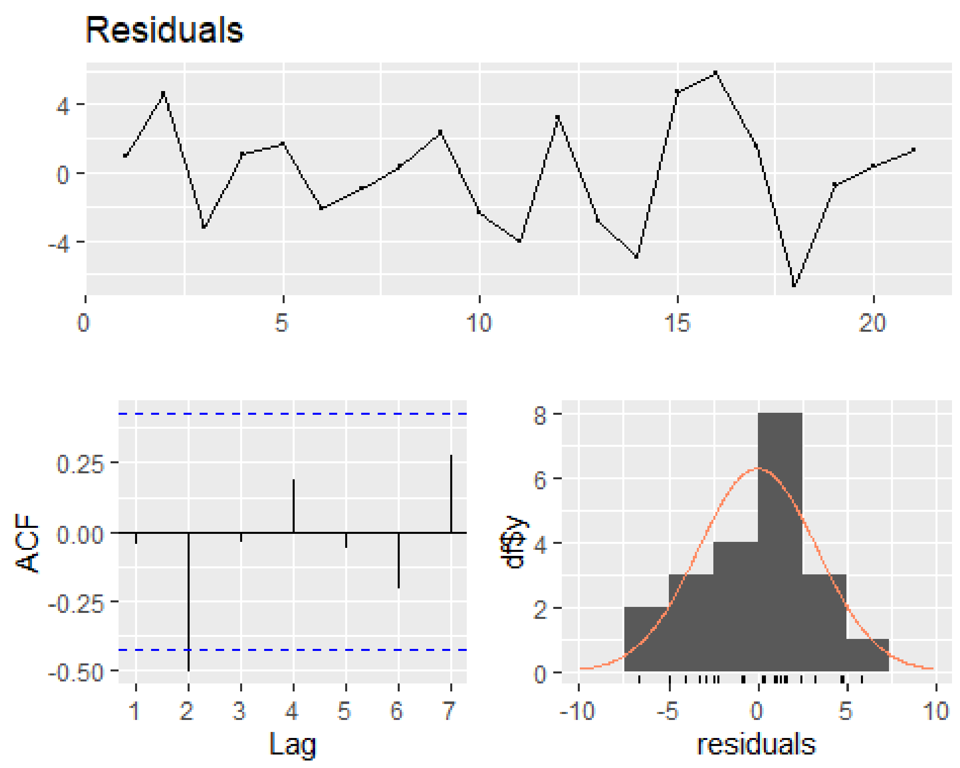
```
model3_finite = dlm(x = as.vector(rad), y = as.vector(ffd),
q =10)

summary(model3_finite)

##
## 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
```

* The model3_finite fit is quite a good fit, as most of the lag variables are statistically significant. Additionally, the adjusted R-squared value is quite high, indicating that the model can explain 80.21% and considering the Multiple R-squared it can explain 91.1% of the variation in the dependent variable. The AIC, BIC, and MASE values for the model are 134.8487 148.4275 and 0.2319041, respectively. The Residual Standard Error (RSE) is 4.935, and the range of residuals spans from a minimum value of -6.6469 to a maximum value of 5.7969. This signifies that, on average, the residuals deviate from the model’s predictions by approximately 4.935 units. A smaller RSE indicates a more precise fit.

```
checkresiduals(model3_finite)
```

```
##          1          2          3          4          5          6          7 
##  0.9315780  4.6717322 -3.2177965  1.1161037  1.6333387 -2.1983122 -0.9005238 
##          8          9         10         11         12         13         14 
##  0.3734698  2.3978682 -2.4027856 -4.0212195  3.1853056 -2.9108368 -4.9822000 
##         15         16         17         18         19         20         21 
##  4.7942895  5.7968951  1.5070977 -6.6468974 -0.7218506  0.3116510  1.2830929
```



```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 7.5762, df = 4, p-value = 0.1084
##
## Model df: 0.   Total lags used: 4
```

```
vif(model3_finite$model)>10
```

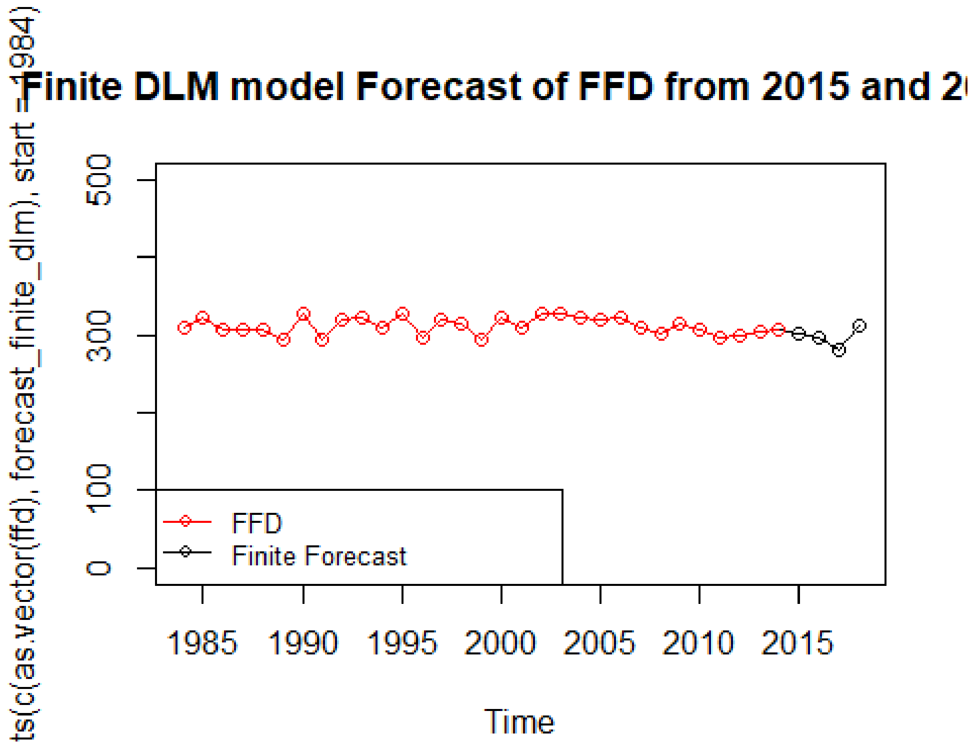
```
##   x.t   x.1   x.2   x.3   x.4   x.5   x.6   x.7   x.8   x.9  x.10
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

 * Examining the VIF values for the lag variables within model3_finite, they are all well below the threshold of 10, suggesting that multicollinearity is not a significant issue in the model.

- Taking into account the results of the Breusch-Godfrey test, the p-value does not fall below the 5% significance level, leading to the acceptance of the null hypothesis. This implies the existence of no serial correlation within the residuals. Further examination of the residuals resulting from the finite dynamic linear model (DLM) fit reveals that they does not exhibit a random distribution. Specifically, the autocorrelation function (ACF) confirming the presence of serial correlation in the residuals. Additionally, the histograms indicate a departure from normal distribution.In summary, the overall conclusion is that the model can be a good fit for the data.

Forecasting with the finite model using the radtion series.

```
forecast_finite_dlm = dLagM::forecast(model = model3_finite, x= c(14.6,14.56,14.79,14.79), h =
4)$forecasts

plot(ts(c(as.vector(ffd),forecast_finite_dlm
    ),start =1984), type="o", color = "black", ylim= c(0, 500), y_lab = "Forecasting of FFD", main="
Finite DLM model Forecast of FFD from 2015 and 2018")
lines(ts(as.vector(ffd),start = 1984),col="red",type="o")
legend("bottomleft",lty=1, pch = 1, text.width = 16, col=c("red","black"),
     c("FFD","Finite Forecast"), cex = 0.8)
```



* The above plot highlights the exceptional performance of the “model3_finite” in effectively capturing the time series data. It’s clear that this model offers highly accurate predictions, implying that we can anticipate only minimal fluctuations in the coming four years. To be more specific, the data shows a slight decline from 2015 to 2016, followed by an expected upward trend. The model’s reliability is reinforced by its close alignment with historical data, making it a robust option for making predictions and well-informed decisions regarding future trends in the time series.

```
finiteDLMauto(x = as.vector(rad), y = as.vector(ffd), 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 0.63554 163.2937 168.5163  2.21922 -13.73328  0.13005 0.3925400
## 5   5 - 2 0.67411 210.3100 216.6004  1.44172   0.45032 -0.04875 0.3464506
## 9   9 - 2 0.68221 173.3759 178.8311  2.04624   1.50345  0.01773 0.6593391
## 4   4 - 2 0.72337 217.6088 224.0880  1.99271   0.65377 -0.05659 0.4509447
## 6   6 - 2 0.72433 200.8402 206.9346  2.07337  -0.34799 -0.05557 0.4725327
## 8   8 - 2 0.72891 181.2550 186.9325  2.32222   0.70433 -0.01989 0.8999747
## 3   3 - 2 0.73654 224.6946 231.3556  4.73910   0.69672 -0.05740 0.4182264
## 2   2 - 2 0.75285 231.5493 238.3857 11.76070   0.67585 -0.05116 0.4531810
## 1   1 - 2 0.75824 238.2644 243.8691 10.05068   1.08447 -0.05138 0.4907193
## 7   7 - 2 0.76118 191.8865 197.7768  2.42301   0.53183 -0.05112 0.8319678
```

 * Based on the results discussed in the previous code, a lag length of 10 and adjust the polynomial order to 2. A second-order polynomial is chosen because it can effectively capture non-linear trends and inflection points and is known to perform well in situations with intricate and complex data patterns.

```
model_poly_temp <- polyDlm(x=as.vector(temp), y=as.vector(ffd), q=10, k = 2)

## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value P(>|t|)
## beta.0    1.1300       2.33   0.485  0.6370
## beta.1    0.8120       1.55   0.522  0.6120
## beta.2    0.4260       1.11   0.383  0.7090
## beta.3   -0.0247       1.03  -0.024  0.9810
## beta.4   -0.5400       1.12  -0.480  0.6410
## beta.5   -1.1200       1.22  -0.915  0.3800
## beta.6   -1.7600       1.27  -1.390  0.1930
## beta.7   -2.4700       1.31  -1.890  0.0859
## beta.8   -3.2500       1.46  -2.220  0.0487
## beta.9   -4.0800       1.86  -2.190  0.0508
## beta.10  -4.9900       2.55  -1.950  0.0765

model_poly_rad <- polyDlm(x=as.vector(rad), y=as.vector(ffd), q=10, k = 2)

## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value P(>|t|)
## beta.0     1.890       3.31   0.571  0.5800
## beta.1     1.010       2.12   0.479  0.6410
## beta.2     0.192       1.47   0.131  0.8990
## beta.3    -0.582       1.38  -0.423  0.6810
## beta.4    -1.310       1.50  -0.870  0.4030
## beta.5    -1.980       1.55  -1.280  0.2270
## beta.6    -2.600       1.46  -1.780  0.1020
## beta.7    -3.180       1.37  -2.310  0.0412
## beta.8    -3.700       1.64  -2.250  0.0459
## beta.9    -4.180       2.50  -1.670  0.1230
## beta.10   -4.600       3.85  -1.190  0.2580

model_poly_rain <- polyDlm(x=as.vector(rain), y=as.vector(ffd), q=10, k = 2)

## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value P(>|t|)
## beta.0    -4.870       4.60  -1.060   0.313
## beta.1    -4.350       3.06  -1.420   0.183
## beta.2    -3.700       2.39  -1.550   0.150
## beta.3    -2.910       2.41  -1.210   0.253
## beta.4    -1.970       2.57  -0.768   0.459
## beta.5    -0.895       2.54  -0.353   0.731
## beta.6     0.322       2.27   0.142   0.890
## beta.7     1.680       1.97   0.853   0.412
## beta.8     3.180       2.28   1.390   0.191
## beta.9     4.820       3.62   1.330   0.210
## beta.10    6.600       5.75   1.150   0.275
```

```
model_poly_hum <- polyDlm(x=as.vector(hum), y=as.vector(ffd), q=10, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value P(>|t|)
## beta.0   -4.8100       2.20 -2.1800  0.0515
## beta.1   -2.8800       1.59 -1.8200  0.0964
## beta.2   -1.3700       1.30 -1.0500  0.3150
## beta.3   -0.2590       1.28 -0.2030  0.8430
## beta.4    0.4350       1.35  0.3230  0.7530
## beta.5    0.7180       1.39  0.5160  0.6160
## beta.6    0.5900       1.38  0.4280  0.6770
## beta.7    0.0493       1.35  0.0365  0.9720
## beta.8   -0.9030       1.42 -0.6340  0.5390
## beta.9   -2.2700       1.75 -1.3000  0.2220
## beta.10  -4.0400       2.38 -1.7000  0.1180
```

```
sort.score(MASE(model_poly_temp$model,model_poly_rad$model,model_poly_rain$model,
model_poly_hum$model), score = "mase")
```

```
##                        n      MASE
## model_poly_rad$model  21 0.6355363
## model_poly_rain$model 21 0.6551896
## model_poly_temp$model 21 0.6796679
## model_poly_hum$model  21 0.7312748
```

```
sort.score(AIC(model_poly_temp$model,model_poly_rad$model,model_poly_rain$model,
model_poly_hum$model), score = "aic")
```

```
##                       df      AIC
## model_poly_rad$model   5 163.2937
## model_poly_hum$model   5 163.5124
## model_poly_temp$model  5 163.7328
## model_poly_rain$model  5 164.9582
```

```
sort.score(BIC(model_poly_temp$model,model_poly_rad$model,model_poly_rain$model,
model_poly_hum$model), score = "bic")
```

```
##                       df      BIC
## model_poly_rad$model   5 168.5163
## model_poly_hum$model   5 168.7350
## model_poly_temp$model  5 168.9554
## model_poly_rain$model  5 170.1808
```

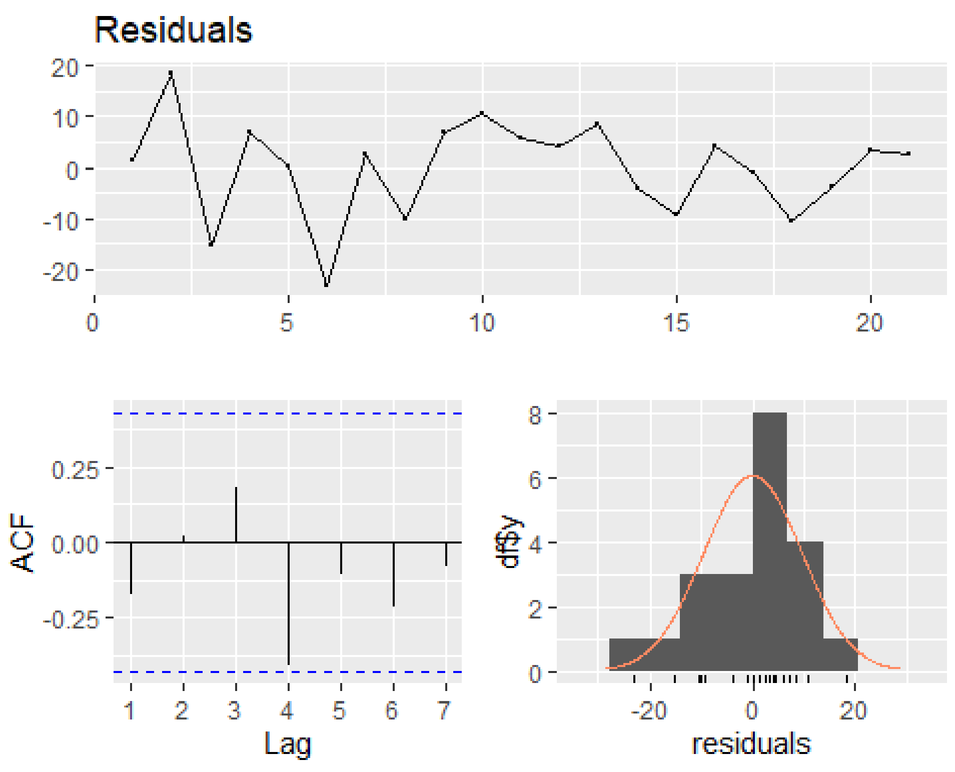
- Considering the result for AIC,BIC and MASE model_poly_rad is the best model. Therefore, this model will undergo further analysis for diagnostic checking.

```
model_poly_rad <- polyDlm(x=as.vector(rad), y=as.vector(ffd), q=10, k = 2)

## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value P(>|t|)
## beta.0     1.890       3.31   0.571  0.5800
## beta.1     1.010       2.12   0.479  0.6410
## beta.2     0.192       1.47   0.131  0.8990
## beta.3    -0.582       1.38  -0.423  0.6810
## beta.4    -1.310       1.50  -0.870  0.4030
## beta.5    -1.980       1.55  -1.280  0.2270
## beta.6    -2.600       1.46  -1.780  0.1020
## beta.7    -3.180       1.37  -2.310  0.0412
## beta.8    -3.700       1.64  -2.250  0.0459
## beta.9    -4.180       2.50  -1.670  0.1230
## beta.10   -4.600       3.85  -1.190  0.2580

summary(model_poly_rad)

##
## 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

vif(model_poly_rad$model)>10

##  z.t0  z.t1  z.t2
## FALSE  TRUE  TRUE
```

* The model “model_poly_rad” is not well fitted and almost all lag variables are statistically insignificant, except for “1st intercept.” The lower adjusted R-squared value 13% and Multiple R-squared value indicates that it can only explain 26.05% of the variation in the dependent variable.

- As seen in the preceding data, the VIF values exceed 10 for every lag variable within “model_poly_rad” This indicates a significant issue of multi collinearity in the model.

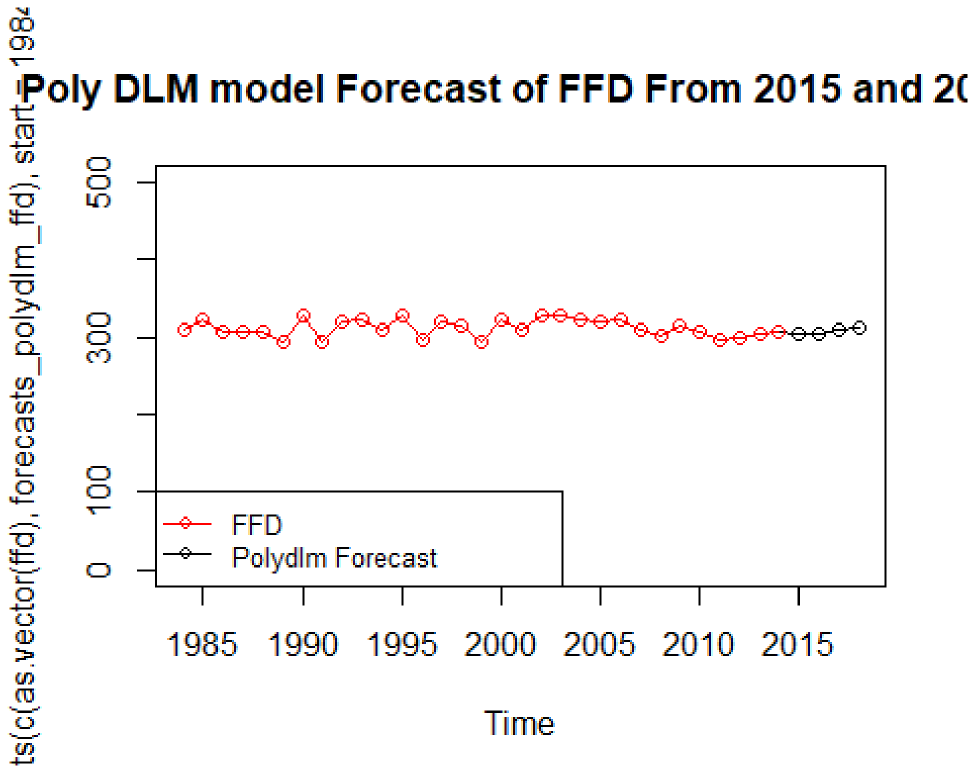
```
checkresiduals(model_poly_rad$model)
```



```
##
##  Breusch-Godfrey test for serial correlation of order up to 7
##
## data:  Residuals
## LM test = 8.8103, df = 7, p-value = 0.2666
```

- Based on the results of the Breusch-Godfrey test, the p-value is greater than the 5% significance level, leading to the acceptance of the null hypothesis, signifying the presence of no serial correlation in the residuals. Examination of residuals from the polynomial DLM reveals a lack of randomness. Notably, the autocorrelation function (ACF) displays lags well inside the 95% confidence interval, confirming the presence of no serial correlation in the residuals, due to the presence of multi collinearity, the fitted polynomial DLM is does not make it the right fit compared to the Finite dlm.

```
forecasts_polydlm_ffd<- dLagM::forecast(model = model_poly_rad, x= c(14.6,14.56,14.79,14.79), h =
4)$forecasts

plot(ts(c(as.vector(ffd),forecasts_polydlm_ffd),start =1984),type="o", color = "black", ylim= c(0, 500),
y_lab = "Forecasting_FFD", main="Poly DLM model Forecast of FFD From 2015 and 2018")
lines(ts(as.vector(ffd),start = 1984),col="red",type="o")
legend("bottomleft",lty=1, pch = 1, text.width = 16, col=c("red","black"),
      c("FFD","Polydlm Forecast"), cex = 0.8)
```



# Koyck Transformation DLM

```
model_Koyock_rad <- koyckDlm(x=as.vector(rad), y = as.vector(ffd))
model_Koyock_hum <- koyckDlm(x=as.vector(hum), y = as.vector(ffd))
model_Koyock_rain <- koyckDlm(x=as.vector(rain), y = as.vector(ffd))
model_Koyock_temp <- koyckDlm(x=as.vector(temp), y = as.vector(ffd))
```

- The plot above showcases not a bad prediction as the model's forecasts closely aligns with historical data and exhibits minimal fluctuations. Overall, the model appears to provide a reasonably accurate prediction for FFD from 2015 to 2018.

```
sort.score(MASE(model_Koyock_rad,model_Koyock_hum,model_Koyock_rain,
model_Koyock_temp), score = "mase")

##                    n      MASE
## model_Koyock_rad  30 0.7540611
## model_Koyock_temp 30 0.7968756
## model_Koyock_hum  30 0.8390184
## model_Koyock_rain 30 1.1546848

AIC(model_Koyock_rad)

## [1] 237.873

AIC(model_Koyock_hum)

## [1] 248.0706
```

```
AIC(model_Koyock_rain)

## [1] 269.7062

AIC(model_Koyock_temp)

## [1] 242.2256

BIC(model_Koyock_rad)

## [1] 243.4778

BIC(model_Koyock_hum)

## [1] 253.6754

BIC(model_Koyock_rain)

## [1] 275.3109

BIC(model_Koyock_temp)

## [1] 247.8304
```

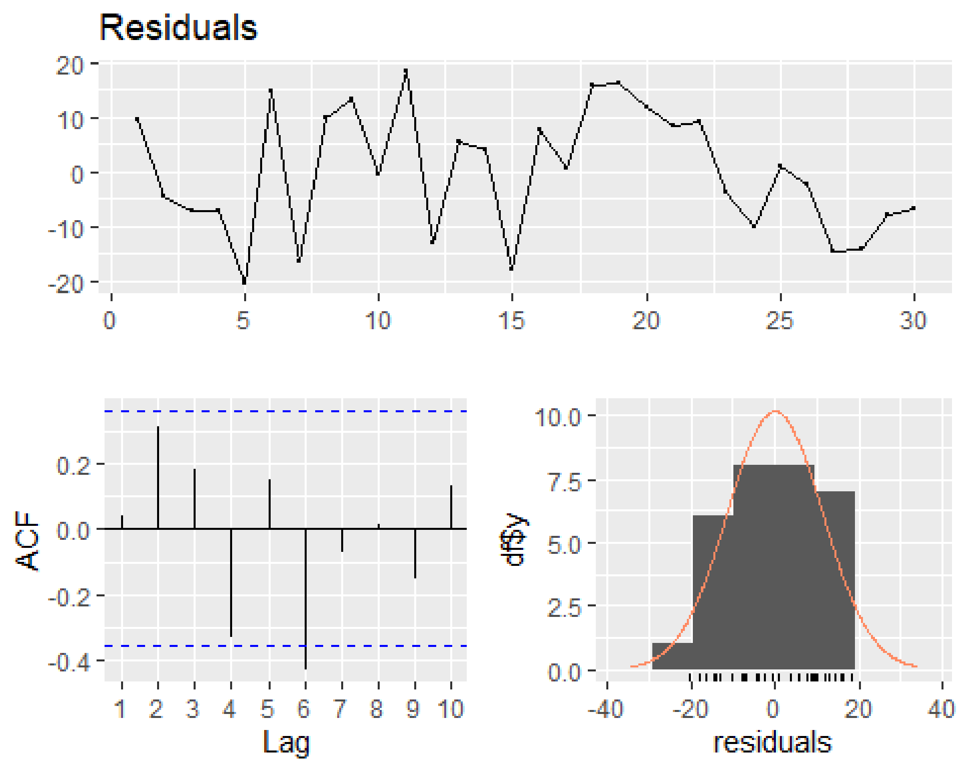
- Considering the result for AIC, BIC and MASE values model_Koyock_rad is the best model. Therefore, this model will undergo further analysis for diagnostic checking.

```
model_Koyock_rad <- koyckDlm(x=as.vector(rad), y = as.vector(ffd))
summary(model_Koyock_rad)

##
## 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:
## NULL
##
```

```
##                          alpha    beta        phi
## Geometric coefficients:  266.1391 3.50942 -0.1179891
```

 * The model_Koyock_rad is not a good fit, as all the lag variables are statistically insignificant. Additionally, the adjusted R-squared value is negative and low, indicating that the model can explain 3.4% of the variation in the dependent variable. The range of residuals spans from a minimum value of -20.3177to a maximum value of 18.2406 . This signifies that, on average, the residuals deviate from the model's predictions by approximately 39 units. A higher RSE indicates not a good fit fit

```
checkresiduals(model_Koyock_rad$model)
```



```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 16.665, df = 6, p-value = 0.01059
##
## Model df: 0.   Total lags used: 6
```

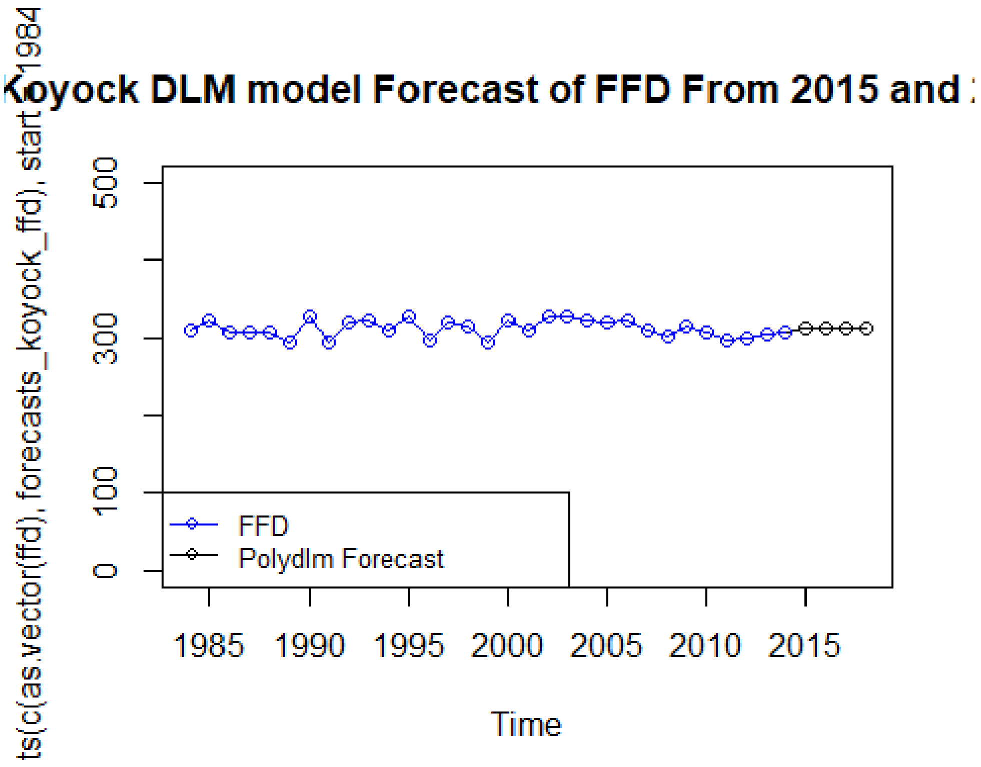
```
vif(model_Koyock_rad$model)>10
```

```
##   Y.1   X.t
## FALSE FALSE
```

 * Examining the VIF (Variance Inflation Factor) values for the lag variables within model_Koyock_rad, they are all well below the threshold of 10, suggesting that multicollinearity is not a significant issue in the model.

- Taking into account the results of the Breusch-Godfrey test, the p-value falls below the 5% significance level, leading to the rejection of the null hypothesis. This implies the existence of serial correlation within the residuals. Further examination of the residuals resulting from the model fit reveals that they do not exhibit a random distribution. Specifically, the autocorrelation function (ACF) confirming the presence of serial correlation in the residuals. Additionally, the histograms indicate a departure from normal distribution. In summary, the overall conclusion is that the model might not be the best fit for the data but the presence of no multicollinearity makes the model ready for plotting.

```
forecasts_koyock_ffd<- dLagM::forecast(model = model_Koyock_rad, x= c(14.6,14.56,14.79,14.79), h
= 4)$forecasts

plot(ts(c(as.vector(ffd),forecasts_koyock_ffd),start =1984),type="o", color = "black", ylim= c(0, 500),
y_lab = "Forecasting_FFD", main="Koyock DLM model Forecast of FFD From 2015 and 2018")
lines(ts(as.vector(ffd),start = 1984),col="blue",type="o")
legend("bottomleft",lty=1, pch = 1, text.width = 16, col=c("blue","black"),
      c("FFD","Polydlm Forecast"), cex = 0.8)
```



- The plot above showcases not a bad prediction as the model’s forecasts closely aligns with historical data and exhibits minimal fluctuations. Overall, the model appears to provide a reasonably accurate prediction for FFD from 2015 to 2018. It can be seen that there is an increase in the year 2016 and slight decrease in 2017 to 2018.

Autoregressive DLM

```
for (i in 1:5){
 for(j in 1:5){
    model_rad_ADLM <- ardlDlm(x= as.vector(rad),y=as.vector(ffd), p = i , q = j )
    cat("p =", i, "q =", j, "AIC =", AIC(model_rad_ADLM $model), "BIC =", BIC(model_rad_ADLM
$model), "MASE =", MASE(model_rad_ADLM )$MASE, "\n")
  }
}
```

```
## p = 1 q = 1 AIC = 239.7969 BIC = 246.8029 MASE = 0.7500823
## p = 1 q = 2 AIC = 231.9906 BIC = 240.1944 MASE = 0.7215629
## p = 1 q = 3 AIC = 224.9528 BIC = 234.2782 MASE = 0.6592462
## p = 1 q = 4 AIC = 212.8182 BIC = 223.1849 MASE = 0.5932357
## p = 1 q = 5 AIC = 207.439 BIC = 218.7618 MASE = 0.5750332
## p = 2 q = 1 AIC = 233.0845 BIC = 241.2883 MASE = 0.745406
## p = 2 q = 2 AIC = 232.439 BIC = 242.0101 MASE = 0.7227858
## p = 2 q = 3 AIC = 224.207 BIC = 234.8646 MASE = 0.6384684
## p = 2 q = 4 AIC = 211.6293 BIC = 223.2919 MASE = 0.5174895
## p = 2 q = 5 AIC = 206.361 BIC = 218.942 MASE = 0.5117951
## p = 3 q = 1 AIC = 228.0292 BIC = 237.3546 MASE = 0.7327656
## p = 3 q = 2 AIC = 227.6194 BIC = 238.2771 MASE = 0.7176158
## p = 3 q = 3 AIC = 226.0447 BIC = 238.0345 MASE = 0.6331971
## p = 3 q = 4 AIC = 213.5508 BIC = 226.5091 MASE = 0.5127345
## p = 3 q = 5 AIC = 208.2714 BIC = 222.1105 MASE = 0.5049498
## p = 4 q = 1 AIC = 222.9547 BIC = 233.3214 MASE = 0.7165712
## p = 4 q = 2 AIC = 221.2968 BIC = 232.9593 MASE = 0.6832271
## p = 4 q = 3 AIC = 219.7407 BIC = 232.6991 MASE = 0.6067828
## p = 4 q = 4 AIC = 214.6715 BIC = 228.9257 MASE = 0.5033145
## p = 4 q = 5 AIC = 209.1419 BIC = 224.239 MASE = 0.4911132
## p = 5 q = 1 AIC = 213.8698 BIC = 225.1927 MASE = 0.641215
## p = 5 q = 2 AIC = 211.9758 BIC = 224.5567 MASE = 0.5966462
## p = 5 q = 3 AIC = 211.9972 BIC = 225.8363 MASE = 0.5696958
## p = 5 q = 4 AIC = 204.6511 BIC = 219.7483 MASE = 0.45492
## p = 5 q = 5 AIC = 206.4543 BIC = 222.8096 MASE = 0.4516374
```

```
for (i in 1:5){
  for(j in 1:5){
    model_rain_ADLM <- ardlDlm(x= as.vector(rain),y=as.vector(ffd), p = i , q = j )
    cat("p =", i, "q =", j, "AIC =", AIC(model_rain_ADLM $model), "BIC =", BIC(model_rain_ADLM
$model), "MASE =", MASE(model_rain_ADLM )$MASE, "\n")
  }
}
```

```
## p = 1 q = 1 AIC = 237.4664 BIC = 244.4724 MASE = 0.7287328
## p = 1 q = 2 AIC = 229.8029 BIC = 238.0067 MASE = 0.6623982
## p = 1 q = 3 AIC = 223.5479 BIC = 232.8734 MASE = 0.6407011
## p = 1 q = 4 AIC = 208.0121 BIC = 218.3788 MASE = 0.499595
## p = 1 q = 5 AIC = 201.1726 BIC = 212.4955 MASE = 0.4629303
## p = 2 q = 1 AIC = 229.6602 BIC = 237.8639 MASE = 0.7028348
## p = 2 q = 2 AIC = 227.5462 BIC = 237.1173 MASE = 0.6319064
## p = 2 q = 3 AIC = 221.8552 BIC = 232.5128 MASE = 0.6033217
## p = 2 q = 4 AIC = 209.0101 BIC = 220.6726 MASE = 0.4806507
## p = 2 q = 5 AIC = 202.4954 BIC = 215.0764 MASE = 0.441155
## p = 3 q = 1 AIC = 222.7896 BIC = 232.115 MASE = 0.6186835
## p = 3 q = 2 AIC = 221.0143 BIC = 231.6719 MASE = 0.5674776
## p = 3 q = 3 AIC = 222.7141 BIC = 234.7039 MASE = 0.5737359
## p = 3 q = 4 AIC = 210.8068 BIC = 223.7651 MASE = 0.4738264
## p = 3 q = 5 AIC = 204.0461 BIC = 217.8852 MASE = 0.4288773
## p = 4 q = 1 AIC = 217.4564 BIC = 227.8231 MASE = 0.6168373
## p = 4 q = 2 AIC = 215.7074 BIC = 227.3699 MASE = 0.5456799
## p = 4 q = 3 AIC = 217.3898 BIC = 230.3482 MASE = 0.5511934
## p = 4 q = 4 AIC = 211.9798 BIC = 226.234 MASE = 0.4458928
## p = 4 q = 5 AIC = 205.6727 BIC = 220.7698 MASE = 0.4123072
## p = 5 q = 1 AIC = 212.5999 BIC = 223.9227 MASE = 0.6284589
## p = 5 q = 2 AIC = 210.8767 BIC = 223.4577 MASE = 0.5552722
## p = 5 q = 3 AIC = 212.6008 BIC = 226.4399 MASE = 0.5583099
## p = 5 q = 4 AIC = 207.1404 BIC = 222.2376 MASE = 0.4456188
## p = 5 q = 5 AIC = 207.6224 BIC = 223.9776 MASE = 0.4143799
```

```
for (i in 1:5){
  for(j in 1:5){
    model_temp_ADLM <- ardlDlm(x= as.vector(temp),y=as.vector(ffd), p = i , q = j )
    cat("p =", i, "q =", j, "AIC =", AIC(model_temp_ADLM $model), "BIC =", BIC(model_temp_ADLM
$model), "MASE =", MASE(model_temp_ADLM )$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
```

* Looking at the above output p= 5 and q =5 has the lowest MASE, therefore further exploration will be conducted using summary and diagnostic checking.

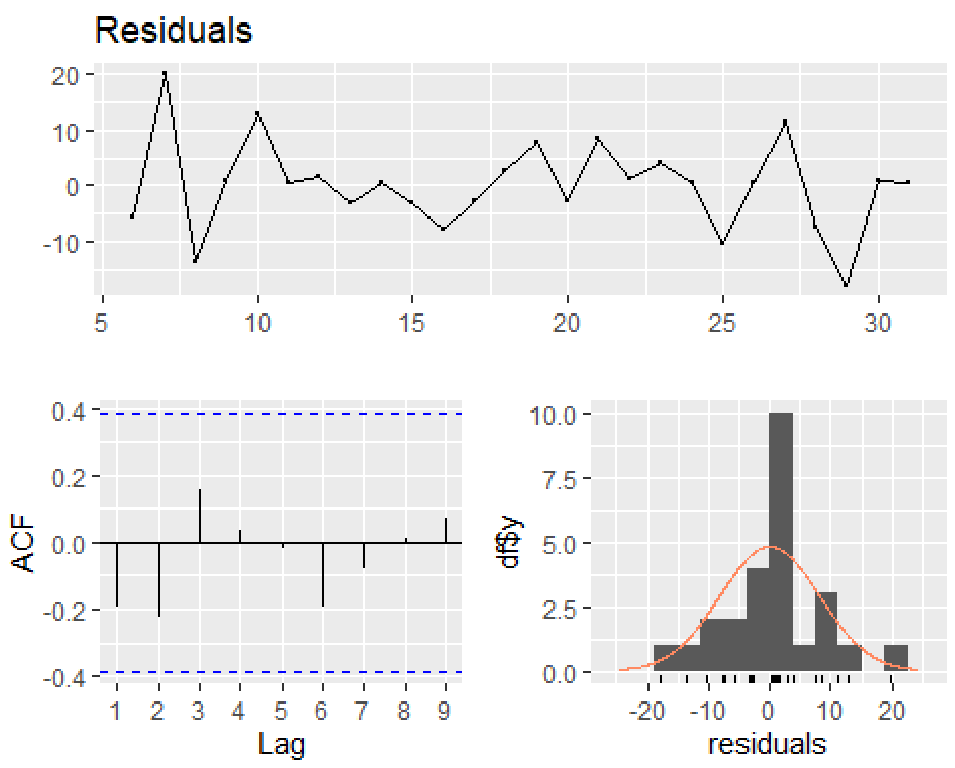
```
model_ard_rad <- ardlDlm(x=as.vector(rad), y=as.vector(ffd),p = 5, q = 5)
model_ard_temp <- ardlDlm(x=as.vector(temp), y=as.vector(ffd),p = 5, q = 5)
model_ard_rain <- ardlDlm(x=as.vector(rain), y=as.vector(ffd),p = 5, q = 5)
model_ard_hum <- ardlDlm(x=as.vector(hum), y=as.vector(ffd),p = 5, q = 5)

sort.score(AIC(model_ard_rad$model,model_ard_temp$model,model_ard_rain$model,
model_ard_hum$model), score = "aic")

##                      df      AIC
## model_ard_rad$model  13 206.4543
## model_ard_rain$model 13 207.6224
## model_ard_hum$model  13 208.0520
## model_ard_temp$model 13 216.3587

sort.score(BIC(model_ard_rad$model,model_ard_temp$model,model_ard_rain$model,
model_ard_hum$model), score = "bic")

##                      df      BIC
## model_ard_rad$model  13 222.8096
## model_ard_rain$model 13 223.9776
## model_ard_hum$model  13 224.4073
## model_ard_temp$model 13 232.7139

MASE(model_ard_rad,model_ard_temp,model_ard_rain, model_ard_hum)

##                 n      MASE
## model_ard_rad  26 0.4516374
## model_ard_temp 26 0.5540506
## model_ard_rain 26 0.4143799
## model_ard_hum  26 0.4860361
```

* Looking at the above output model_ard_rain MASE has lower mase meaning the radiation series will be the best to proceed for further exploration.

```
model_ard_rain <- ardlDlm(x=as.vector(rain), y=as.vector(ffd),p = 5, q = 5)
summary(model_ard_rain)

##
## 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
```

```
checkresiduals(model_ard_rain)
```

```
## Time Series:
## Start = 6
## End = 31
## Frequency = 1
##           6           7           8           9          10          11
##  -5.8104792  19.9177453 -13.5189977   0.9836014  12.8961182   0.6204403
##          12          13          14          15          16          17
##   1.5570187  -3.1802774   0.3599324  -2.9858150  -7.7063008  -2.8481524
##          18          19          20          21          22          23
##   2.7959743   7.6897476  -2.6154331   8.6131657   1.3568071   3.9887270
##          24          25          26          27          28          29
##   0.5687015 -10.3556200   0.6530124  11.2254034  -7.4453393 -17.8733411
##          30          31
##   0.7884984   0.3248624
```



```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 3.4505, df = 5, p-value = 0.6309
##
## Model df: 0.   Total lags used: 5
```

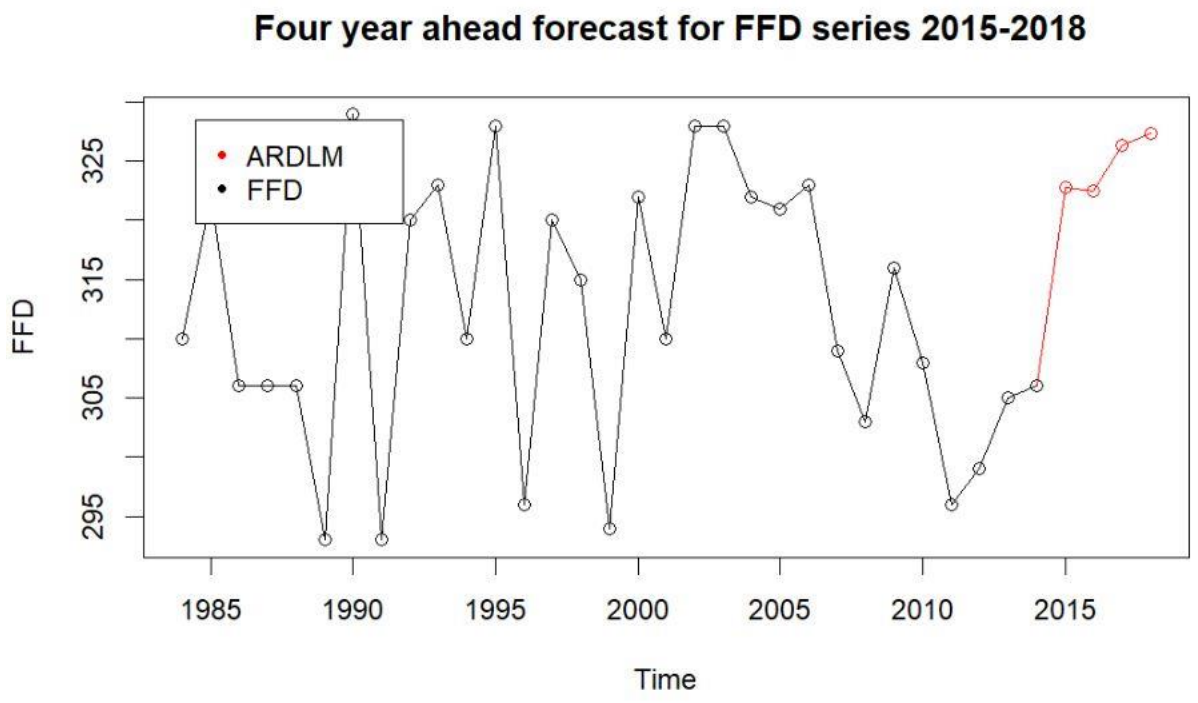
```
vif(model_ard_rain$model)
```

```
##       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
```

 * Examining the VIF (Variance Inflation Factor) values for the lag variables within `model_ard_rain`, they are all well below the threshold of 10, suggesting that multicollinearity is not a significant issue in the model.

- Taking into account the results of the Breusch-Godfrey test, the p-value exceeds the 5% significance level, leading to the acceptance of the null hypothesis. This implies the existence of no serial correlation within the residuals. Further examination of the residuals resulting from the model reveals that they do not exhibit a random distribution. Specifically, the autocorrelation function (ACF) confirming the presence of no serial correlation in the residuals. Additionally, the histograms indicate a departure from normal distribution.In summary, the overall conclusion is that the model might not be the best fit for the data but the presence of no multicollinearity makes the model ready for plotting.

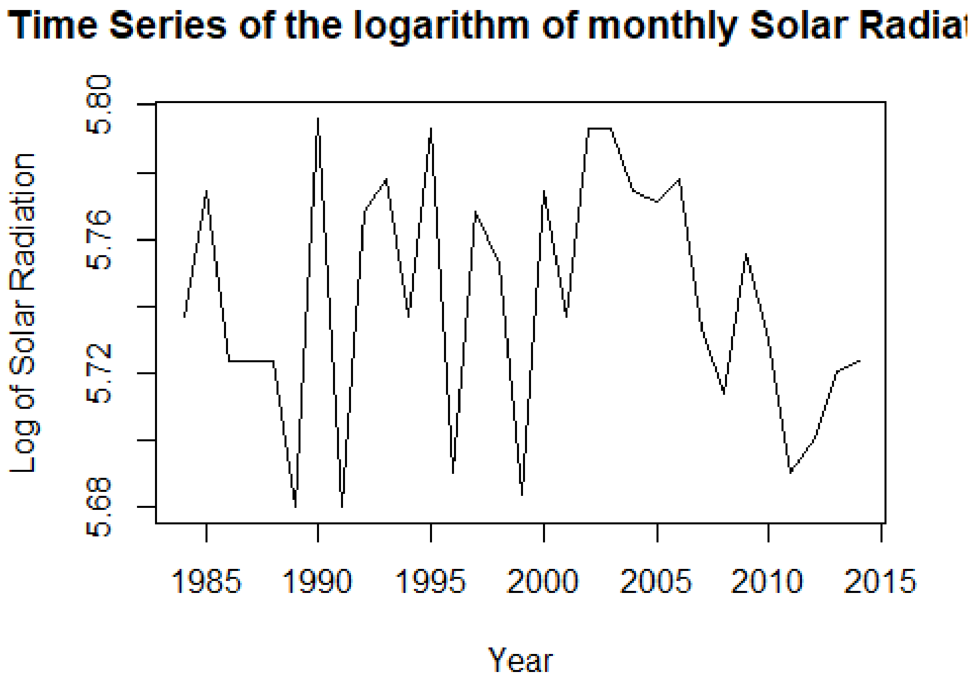
```
ffd_ardl2 <- ardlDlm(formula = ffd ~ rain, data = data.frame(ffd, rain) ,p = 5,q = 5)

cov<- matrix(c(2.27,2.38,2.26,2.27), ncol = 4)

ardlm_frc_rain = (dLagM::forecast(ffd_ardl2, x = cov, h=4))


ardlm_frc_rain = round(ardlm_frc_rain$forecasts,2)

plot(ts(c(as.vector(ffd),ardlm_frc_rain), start=1984),type="o",col="red", ylab="FFD", main="
Four year ahead forecast for FFD series 2015-2018",)

lines(ts(as.vector(ffd),start=1984),col="black",type="o")

legend("topleft",inset = 0.05,pch=20,text.width=5, col = c("red","black"), c("ARDLM ","FFD"))
```



- The plot above showcases not a bad prediction as the model's forecasts closely aligns with historical data and exhibits minimal fluctuations. Overall, the model appears to provide a reasonably accurate prediction for FFD from 2015 to 2018. It can be seen that there is an increase in the year 2016 onward.

Dynamic Linear Modelling

```
log_ffd<- log(ffd)

par(mfrow=c(1,1))
plot(log_ffd,ylab='Log of Solar Radiation',xlab='Year',
    main = "Time Series of the logarithm of monthly Solar Radiation")
```



```
Y.t <- log(ffd)
t <- 450
P.t <- 1*(seq(Y.t)==T)
P.t.1 <- lag(P.t, +1)
```

 * Considering the ADF test,ACF and PACF plots it was identified that there was no trend and seasonality, therefore Dynamic Linear Models will be adjusted with trend and no seasonal components.

```
dynlm_ffd_1<- dynlm(Y.t~ L(Y.t, k=1) + trend(Y.t))
dynlm_ffd_2 <- dynlm(Y.t~ L(Y.t, k=1) + P.t.1 + P.t + trend(Y.t))
dynlm_ffd_3<- dynlm(Y.t~ L(Y.t, k=1))
dynlm_ffd_4 <- dynlm(Y.t~ L(Y.t, k=1) + P.t)
dynlm_ffd_5 <- dynlm(Y.t~ L(Y.t, k=1) + P.t.1)
```

```
dynlm_ffd_6 <- dynlm(Y.t~ L(Y.t, k=1) + L(Y.t, k=2))
dynlm_ffd_7 <- dynlm(Y.t~ L(Y.t, k=1) + L(Y.t, k=2)+P.t)
dynlm_ffd_8 <- dynlm(Y.t~ L(Y.t, k=1) + L(Y.t, k=2)+P.t + P.t.1)
dynlm_ffd_9 <- dynlm(Y.t~ L(Y.t, k=1) + L(Y.t, k=2)+L(Y.t, k=3))
dynlm_ffd_10 <- dynlm(Y.t~ L(Y.t, k=1) + L(Y.t, k=2)+L(Y.t, k=3)+L(Y.t, k=4))
dynlm_ffd_11 <- dynlm(Y.t~ L(Y.t, k=1) + L(Y.t, k=2)+L(Y.t, k=3)+L(Y.t, k=4)+L(Y.t, k=5))

dynlm_ffd_temp_1<- dynlm(Y.t~temp + L(Y.t, k=1) + trend(Y.t))
dynlm_ffd_temp_2<- dynlm(Y.t~temp + L(Y.t, k=1))
dynlm_ffd_temp_3 <- dynlm(Y.t~temp + L(Y.t, k=1) + P.t)
dynlm_ffd_temp_4 <- dynlm(Y.t~temp +L(Y.t, k=1) + P.t.1)
dynlm_ffd_temp_5 <- dynlm(Y.t~temp + L(Y.t, k=1) + L(Y.t, k=2))
dynlm_ffd_temp_6 <- dynlm(Y.t~temp + L(Y.t, k=1) + L(Y.t, k=2)+P.t)
dynlm_ffd_temp_7 <- dynlm(Y.t~temp + L(Y.t, k=1) + L(Y.t, k=2)+P.t + P.t.1)
dynlm_ffd_temp_8 <- dynlm(Y.t~temp + L(Y.t, k=1) + L(Y.t, k=2)+L(Y.t, k=3))

dynlm_ffd_rad_1<- dynlm(Y.t~rad + L(Y.t, k=1) + trend(Y.t))
dynlm_ffd_rad_2<- dynlm(Y.t~rad + L(Y.t, k=1))
dynlm_ffd_rad_3 <- dynlm(Y.t~rad + L(Y.t, k=1) + P.t)
dynlm_ffd_rad_4 <- dynlm(Y.t~rad +L(Y.t, k=1) + P.t.1)
dynlm_ffd_rad_5 <- dynlm(Y.t~rad + L(Y.t, k=1) + L(Y.t, k=2))
dynlm_ffd_rad_6 <- dynlm(Y.t~rad + L(Y.t, k=1) + L(Y.t, k=2)+P.t)
dynlm_ffd_rad_7 <- dynlm(Y.t~rad + L(Y.t, k=1) + L(Y.t, k=2)+P.t + P.t.1)
dynlm_ffd_rad_8 <- dynlm(Y.t~rad + L(Y.t, k=1) + L(Y.t, k=2)+L(Y.t, k=3))


dynlm_ffd_hum_1<- dynlm(Y.t~hum + L(Y.t, k=1) + trend(Y.t))
dynlm_ffd_hum_2<- dynlm(Y.t~hum + L(Y.t, k=1))
dynlm_ffd_hum_3 <- dynlm(Y.t~hum + L(Y.t, k=1) + P.t)
dynlm_ffd_hum_4 <- dynlm(Y.t~hum +L(Y.t, k=1) + P.t.1)
dynlm_ffd_hum_5 <- dynlm(Y.t~hum + L(Y.t, k=1) + L(Y.t, k=2))
dynlm_ffd_hum_6 <- dynlm(Y.t~hum + L(Y.t, k=1) + L(Y.t, k=2)+P.t)
dynlm_ffd_hum_7 <- dynlm(Y.t~hum + L(Y.t, k=1) + L(Y.t, k=2)+P.t + P.t.1)
dynlm_ffd_hum_8 <- dynlm(Y.t~hum + L(Y.t, k=1) + L(Y.t, k=2)+L(Y.t, k=3))



dynlm_ffd_rain_1<- dynlm(Y.t~rain + L(Y.t, k=1) + trend(Y.t))
dynlm_ffd_rain_2<- dynlm(Y.t~rain + L(Y.t, k=1))
dynlm_ffd_rain_3 <- dynlm(Y.t~rain + L(Y.t, k=1) + P.t)
dynlm_ffd_rain_4 <- dynlm(Y.t~rain +L(Y.t, k=1) + P.t.1)
dynlm_ffd_rain_5 <- dynlm(Y.t~rain + L(Y.t, k=1) + L(Y.t, k=2))
dynlm_ffd_rain_6 <- dynlm(Y.t~rain + L(Y.t, k=1) + L(Y.t, k=2)+P.t)
dynlm_ffd_rain_7 <- dynlm(Y.t~rain + L(Y.t, k=1) + L(Y.t, k=2)+P.t + P.t.1)
dynlm_ffd_rain_8 <- dynlm(Y.t~rain + L(Y.t, k=1) + L(Y.t, k=2)+L(Y.t, k=3))


dynlm_ffd_no<- dynlm(Y.t~ L(Y.t, k=1)-1)
dynlm_ffd_4_no <- dynlm(Y.t~ L(Y.t, k=1) + P.t -1)
```

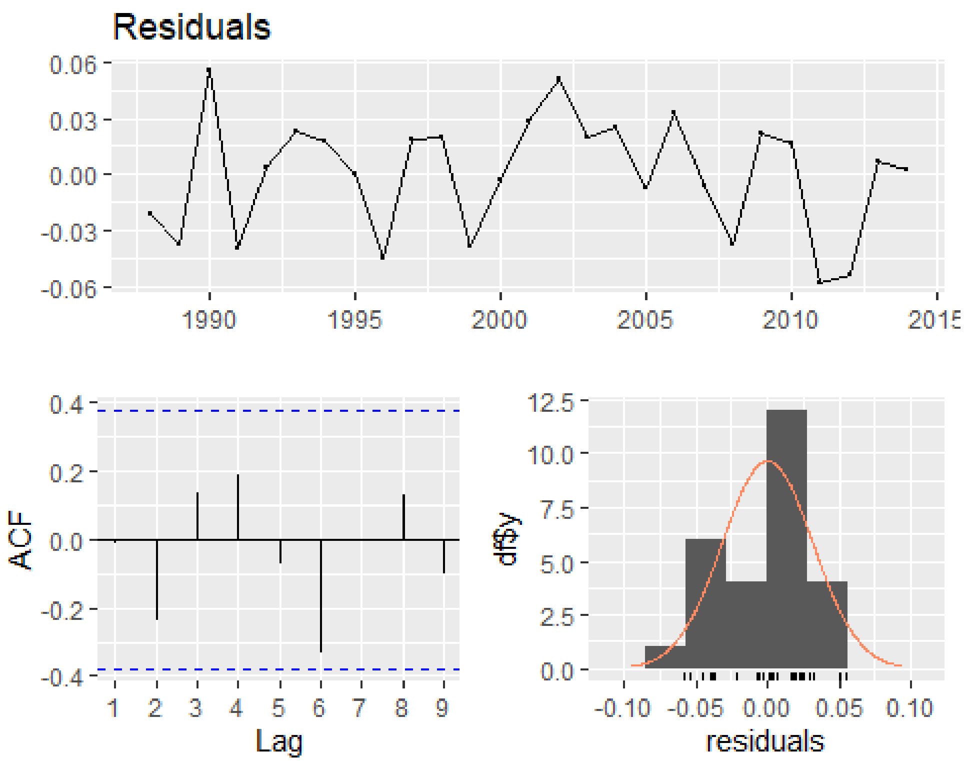
```
dynlm_ffd_temp_no<- dynlm(Y.t~temp + L(Y.t, k=1)-1)
dynlm_ffd_rad_no<- dynlm(Y.t~rad + L(Y.t, k=1)-1)

MASE(lm(dynlm_ffd_1),lm(dynlm_ffd_2),lm(dynlm_ffd_3), lm(dynlm_ffd_4),lm(dynlm_ffd_5),
lm(dynlm_ffd_6),lm(dynlm_ffd_7),lm(dynlm_ffd_8),lm(dynlm_ffd_9),lm(dynlm_ffd_10),
lm(dynlm_ffd_11),lm(dynlm_ffd_temp_1),lm(dynlm_ffd_temp_2),lm(dynlm_ffd_temp_3),
lm(dynlm_ffd_temp_4),lm(dynlm_ffd_temp_5),lm(dynlm_ffd_temp_6),lm(dynlm_ffd_temp_7),
lm(dynlm_ffd_temp_8),lm(dynlm_ffd_rad_1),
lm(dynlm_ffd_rad_2),lm(dynlm_ffd_rad_3),lm(dynlm_ffd_rad_4),lm(dynlm_ffd_rad_5),
lm(dynlm_ffd_rad_6),lm(dynlm_ffd_rad_7),lm(dynlm_ffd_rad_8),lm(dynlm_ffd_hum_1),
lm(dynlm_ffd_hum_2),lm(dynlm_ffd_hum_3),lm(dynlm_ffd_hum_4),lm(dynlm_ffd_hum_5),
lm(dynlm_ffd_hum_6),lm(dynlm_ffd_hum_7),
lm(dynlm_ffd_rain_1),
lm(dynlm_ffd_rain_2),lm(dynlm_ffd_rain_3),lm(dynlm_ffd_rain_4),lm(dynlm_ffd_rain_5),
lm(dynlm_ffd_rain_6),lm(dynlm_ffd_rain_7),lm(dynlm_ffd_rain_8), lm(dynlm_ffd_temp_no),
lm(dynlm_ffd_rad_no))

##                        n      MASE
## lm(dynlm_ffd_1)       30 0.7556991
## lm(dynlm_ffd_2)       30 0.7344319
## lm(dynlm_ffd_3)       30 0.7638151
## lm(dynlm_ffd_4)       30 0.7638151
## lm(dynlm_ffd_5)       30 0.7360008
## lm(dynlm_ffd_6)       29 0.7211532
## lm(dynlm_ffd_7)       29 0.7211532
## lm(dynlm_ffd_8)       29 0.7211532
## lm(dynlm_ffd_9)       28 0.6655454
## lm(dynlm_ffd_10)      27 0.5786860
## lm(dynlm_ffd_11)      26 0.5794746
## lm(dynlm_ffd_temp_1)  30 0.7523692
## lm(dynlm_ffd_temp_2)  30 0.7544082
## lm(dynlm_ffd_temp_3)  30 0.7544082
## lm(dynlm_ffd_temp_4)  30 0.7295233
## lm(dynlm_ffd_temp_5)  29 0.7207923
## lm(dynlm_ffd_temp_6)  29 0.7207923
## lm(dynlm_ffd_temp_7)  29 0.7207923
## lm(dynlm_ffd_temp_8)  28 0.6599284
## lm(dynlm_ffd_rad_1)   30 0.7366381
## lm(dynlm_ffd_rad_2)   30 0.7481176
## lm(dynlm_ffd_rad_3)   30 0.7481176
## lm(dynlm_ffd_rad_4)   30 0.7250853
## lm(dynlm_ffd_rad_5)   29 0.7184964
## lm(dynlm_ffd_rad_6)   29 0.7184964
## lm(dynlm_ffd_rad_7)   29 0.7184964
## lm(dynlm_ffd_rad_8)   28 0.6653619
## lm(dynlm_ffd_hum_1)   30 0.7550707
## lm(dynlm_ffd_hum_2)   30 0.7565113
## lm(dynlm_ffd_hum_3)   30 0.7565113
## lm(dynlm_ffd_hum_4)   30 0.7285708
## lm(dynlm_ffd_hum_5)   29 0.7293316
## lm(dynlm_ffd_hum_6)   29 0.7293316
## lm(dynlm_ffd_hum_7)   29 0.7293316
## lm(dynlm_ffd_rain_1)  30 0.7319067
## lm(dynlm_ffd_rain_2)  30 0.7358060
## lm(dynlm_ffd_rain_3)  30 0.7358060
## lm(dynlm_ffd_rain_4)  30 0.7045486
## lm(dynlm_ffd_rain_5)  29 0.7111800
## lm(dynlm_ffd_rain_6)  29 0.7111800
## lm(dynlm_ffd_rain_7)  29 0.7111800
## lm(dynlm_ffd_rain_8)  28 0.6523088
## lm(dynlm_ffd_temp_no) 30 0.9773088
## lm(dynlm_ffd_rad_no)  30 0.9972821
```

* Considering all the models and fitting the model with intercept and non intercept, dynlm_ffd_rain_10 has the lowest MASE.

```
summary(dynlm_ffd_10)

## 
## Time series regression with "ts" data:
## Start = 1988, End = 2014
## 
## Call:
## dynlm(formula = Y.t ~ L(Y.t, k = 1) + L(Y.t, k = 2) + L(Y.t, 
##     k = 3) + L(Y.t, k = 4))
## 
## Residuals:
##       Min        1Q    Median        3Q       Max 
## -0.057722 -0.028937  0.004072  0.020885  0.055190 
## 
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)  
## (Intercept)    4.72159    1.93886   2.435   0.0234 *
## L(Y.t, k = 1) -0.01323    0.19122  -0.069   0.9455  
## L(Y.t, k = 2)  0.44051    0.18336   2.403   0.0252 *
## L(Y.t, k = 3)  0.19680    0.19247   1.023   0.3177  
## L(Y.t, k = 4) -0.44610    0.20455  -2.181   0.0402 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.03419 on 22 degrees of freedom
## Multiple R-squared:  0.3282, Adjusted R-squared:  0.2061 
## F-statistic: 2.687 on 4 and 22 DF,  p-value: 0.05797
```

* The model labeled “dynlm_ffd_rain_10” does not appear to be a better fit. This is evident from its slightly lower Adjusted R-squared value of 20.61%.

```
vif(dynlm_ffd_10)

## L(Y.t, k = 1) L(Y.t, k = 2) L(Y.t, k = 3) L(Y.t, k = 4) 
##      1.197405      1.097170      1.179962      1.225427

checkresiduals(dynlm_ffd_10)
```



```
##
##  Breusch-Godfrey test for serial correlation of order up to 8
##
## data:  Residuals
## LM test = 10.22, df = 8, p-value = 0.2499
```

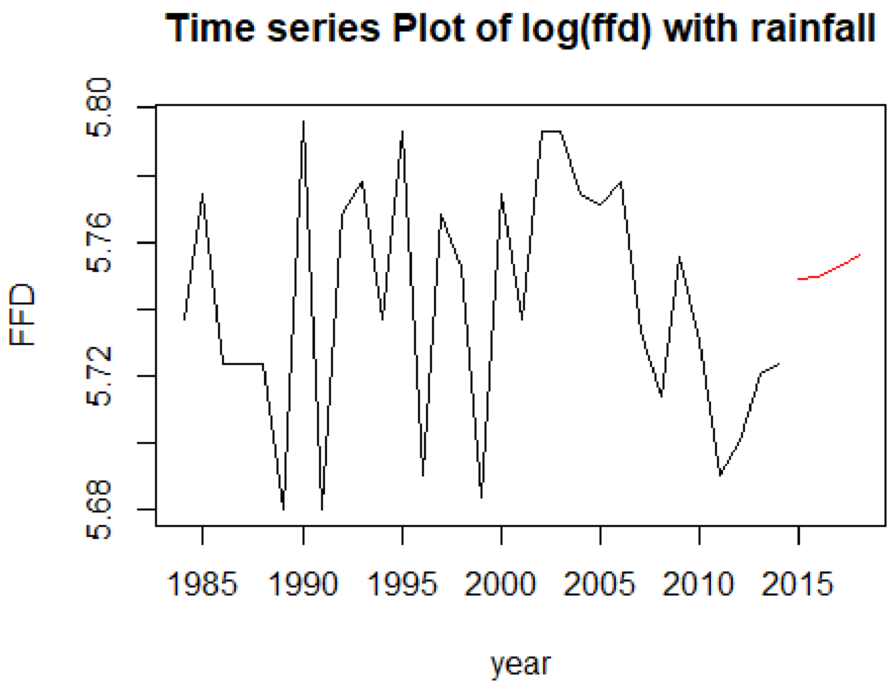
 * Examining the VIF (Variance Inflation Factor) values for the lag variables within dynlm_ffd_rain_10, they are all well below the threshold of 10, suggesting that multicollinearity is not a significant issue in the model.

- the p-value does not fall below the 5% significance level, leading to the acceptance of the null hypothesis. This implies the existence of no serial correlation within the residuals.

 # Forecasting with Dynamic Linear Models

```
q <- 4
n <- nrow(dynlm_ffd_10$model)

ffd.frc <- rep(NA, n + q)

ffd.frc[1:n] <- Y.t[5:(4 + n)]

for (i in 1:q) {
  data_new <- c(1, ffd.frc[(n - 1 + i):(n - 4 + i)])
  ffd.frc[n + i] <- sum(dynlm_ffd_10$coefficients * data_new)
}
```

```
par(mfrow = c(1, 1))
plot(Y.t, xlim = c(1984, 2018), ylab = "FFD", xlab = "year", main = "Time series Plot of log(ffd) with
rainfall")
lines(ts(ffd.frc[(n + 1):(n + q)], start = c(2015)), col = "red")
```



* As mentioned before the model is not a good fit and the forecasting does not provide a reliable values.

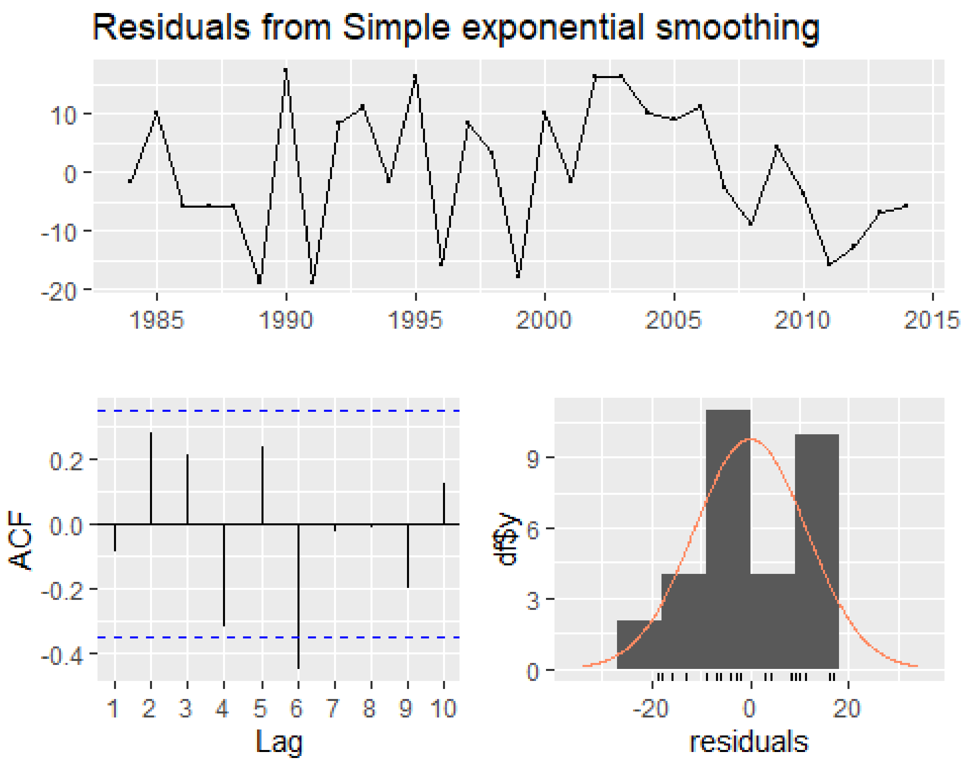
Exponential Smoothing

The FFD series lacks any discernible seasonal or trend components, rendering methods such as “Holt’s Linear Method,” the “Damped Trend Method,” and the “Exponential Trend Method,” among others, unsuitable for fitting.

```
fit1 <- ses(ffd, h=1)
fit2<- ses(ffd, h=2)
fit3 <- ses(ffd, h=3)
fit4 <- ses(ffd, h=4)

summary(fit1)

##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
```

```
## Call:
##  ses(y = ffd, h = 1)
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 311.8397
##
##   sigma:  11.5528
##
##      AIC     AICc      BIC
## 262.0960 262.9848 266.3979
##
## 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
##
## Forecasts:
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2015       311.8397 297.0341 326.6452 289.1965 334.4828
```

```
checkresiduals(fit1)
```

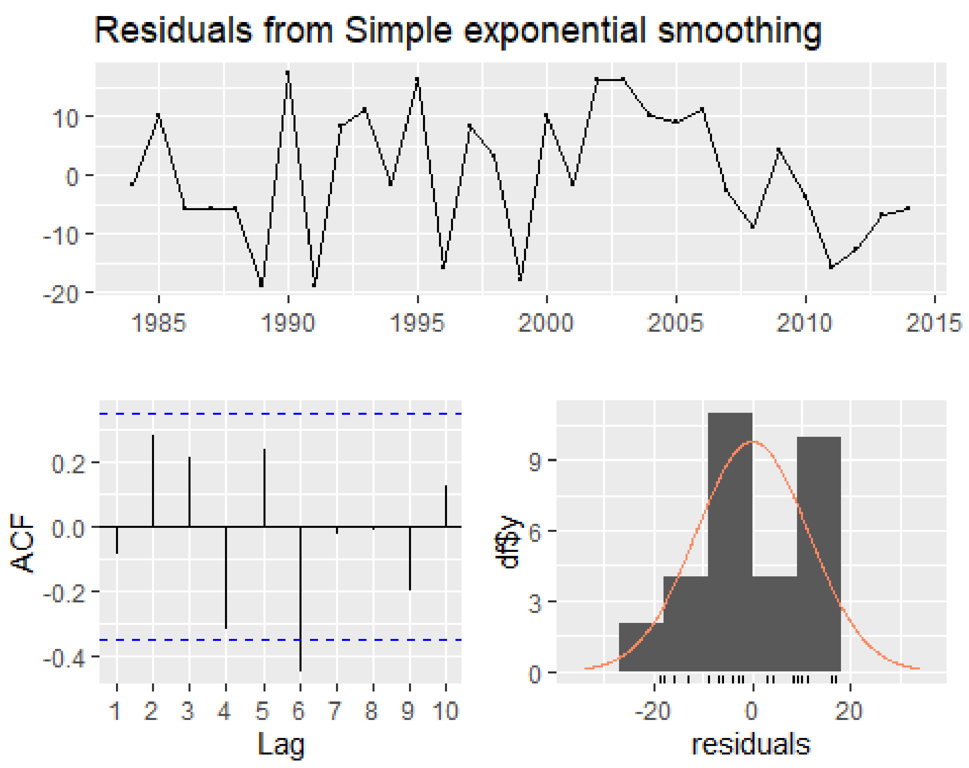


```
##
##  Ljung-Box test
##
## data:  Residuals from Simple exponential smoothing
## Q* = 19.27, df = 6, p-value = 0.003731
##
## Model df: 0.   Total lags used: 6
```

```
summary(fit2)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = ffd, h = 2)
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 311.8397
##
##   sigma:  11.5528
##
##      AIC     AICc      BIC
## 262.0960 262.9848 266.3979
##
## 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
##
## Forecasts:
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2015       311.8397 297.0341 326.6452 289.1965 334.4828
## 2016       311.8397 297.0341 326.6452 289.1965 334.4828
```

```
checkresiduals(fit2)
```



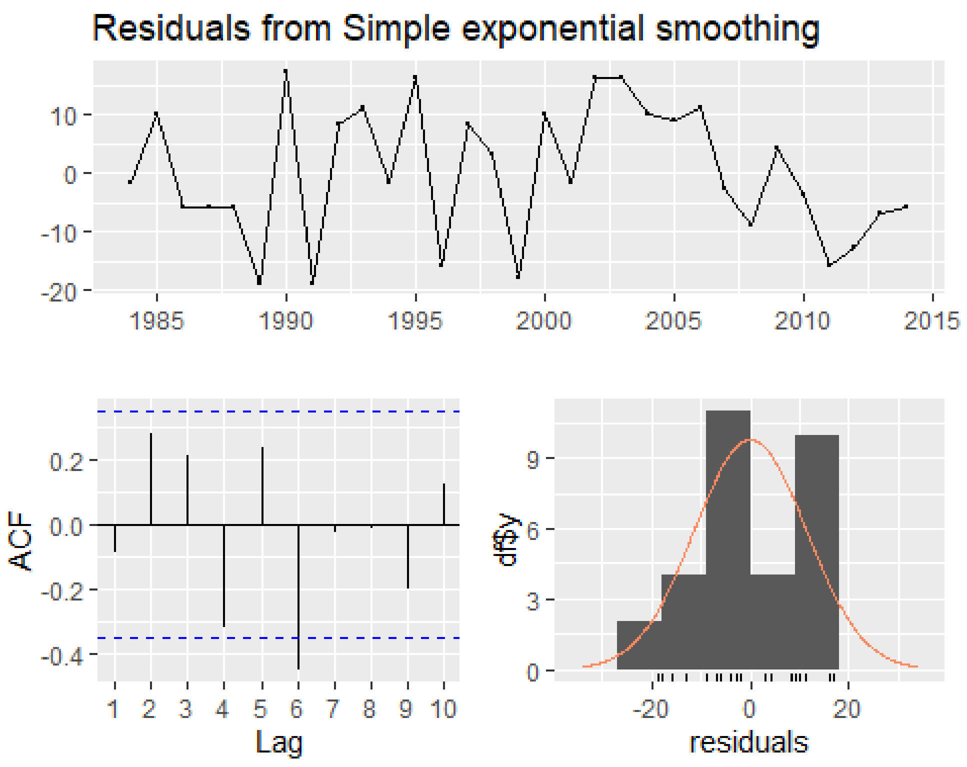
```
##
##  Ljung-Box test
##
## data:  Residuals from Simple exponential smoothing
## Q* = 19.27, df = 6, p-value = 0.003731
##
## Model df: 0.   Total lags used: 6
```

```
summary(fit3)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = ffd, h = 3)
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 311.8397
##
##   sigma:  11.5528
##
```

```
##       AIC      AICc       BIC
## 262.0960 262.9848 266.3979
##
## 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
##
## Forecasts:
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2015       311.8397 297.0341 326.6452 289.1965 334.4828
## 2016       311.8397 297.0341 326.6452 289.1965 334.4828
## 2017       311.8397 297.0341 326.6452 289.1965 334.4828
```

```
checkresiduals(fit3)
```

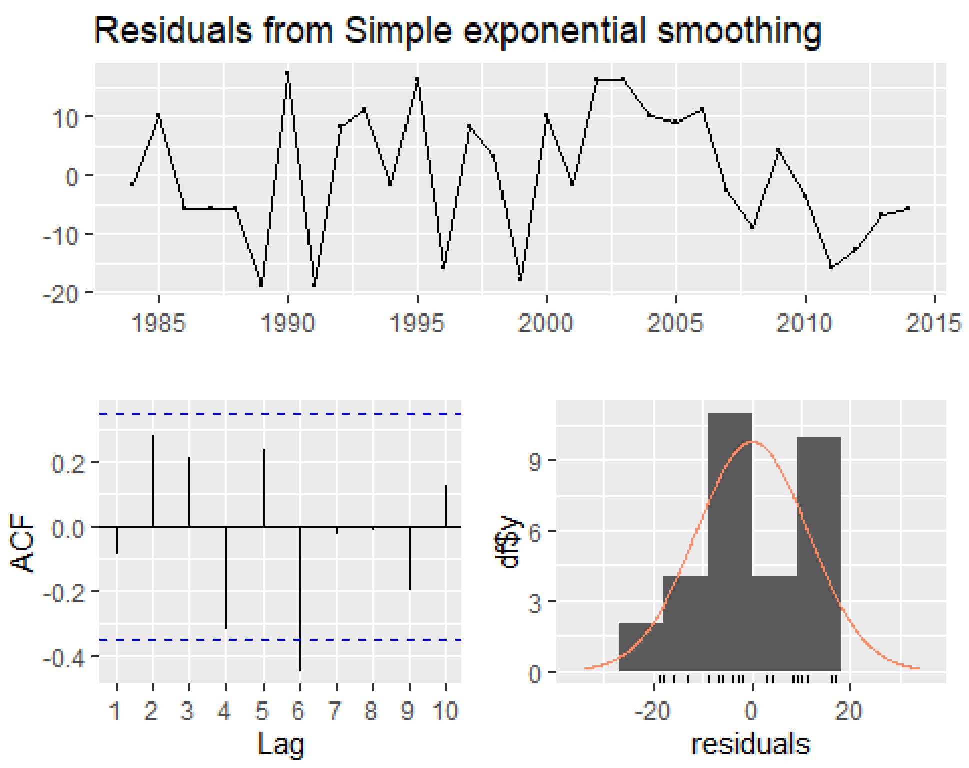


```
##
##  Ljung-Box test
##
## data:  Residuals from Simple exponential smoothing
## Q* = 19.27, df = 6, p-value = 0.003731
##
## Model df: 0.   Total lags used: 6
```

```
summary(fit4)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = ffd, h = 4)
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 311.8397
##
##   sigma:  11.5528
##
##      AIC     AICc      BIC
## 262.0960 262.9848 266.3979
##
## 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
##
## Forecasts:
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2015       311.8397 297.0341 326.6452 289.1965 334.4828
## 2016       311.8397 297.0341 326.6452 289.1965 334.4828
## 2017       311.8397 297.0341 326.6452 289.1965 334.4828
## 2018       311.8397 297.0341 326.6452 289.1965 334.4828
```

```
checkresiduals(fit4)
```

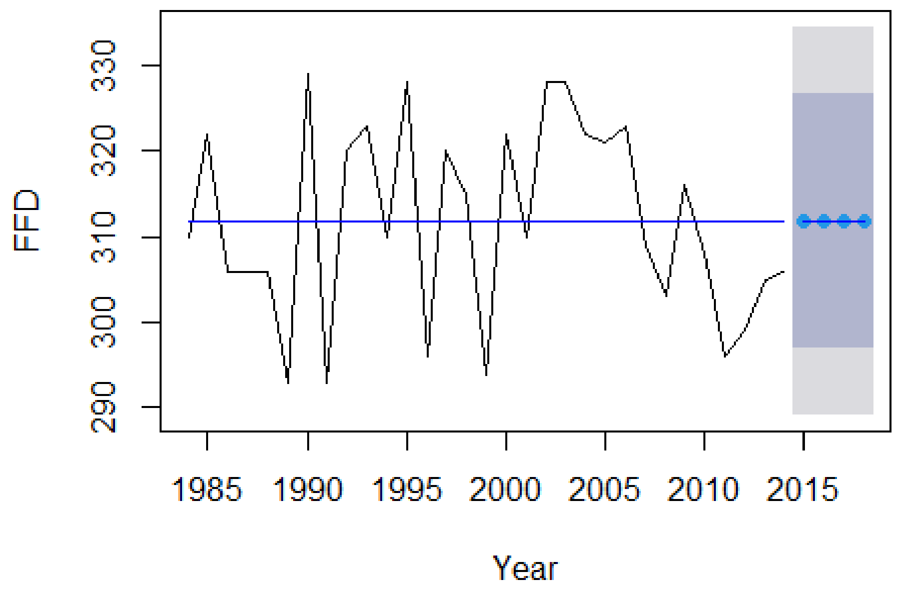


```
##
##  Ljung-Box test
##
## data:  Residuals from Simple exponential smoothing
## Q* = 19.27, df = 6, p-value = 0.003731
##
## Model df: 0.   Total lags used: 6
```

* None Of the models are a good fit because the MASE value is significantly higher than the MASE values of the previous models. All models show signs of presence of serial correlation as all the p-values are less than the 5% level of significance.

- Forecasting with the fit4 model

```
plot(fit4, plot.conf=FALSE, ylab="FFD", xlab="Year", main="", type="l")
lines(fitted(fit4), col="blue", type="l")
lines(fit4$mean, col="blue", type="l")
```



* As previously noted, the exponential smoothing model is not a suitable choice for forecasting. An examination of the fitted values provides further evidence of its inadequacy, as it consistently produces identical prediction values for the years 2015, 2016, 2017, and 2018.

State Space Models

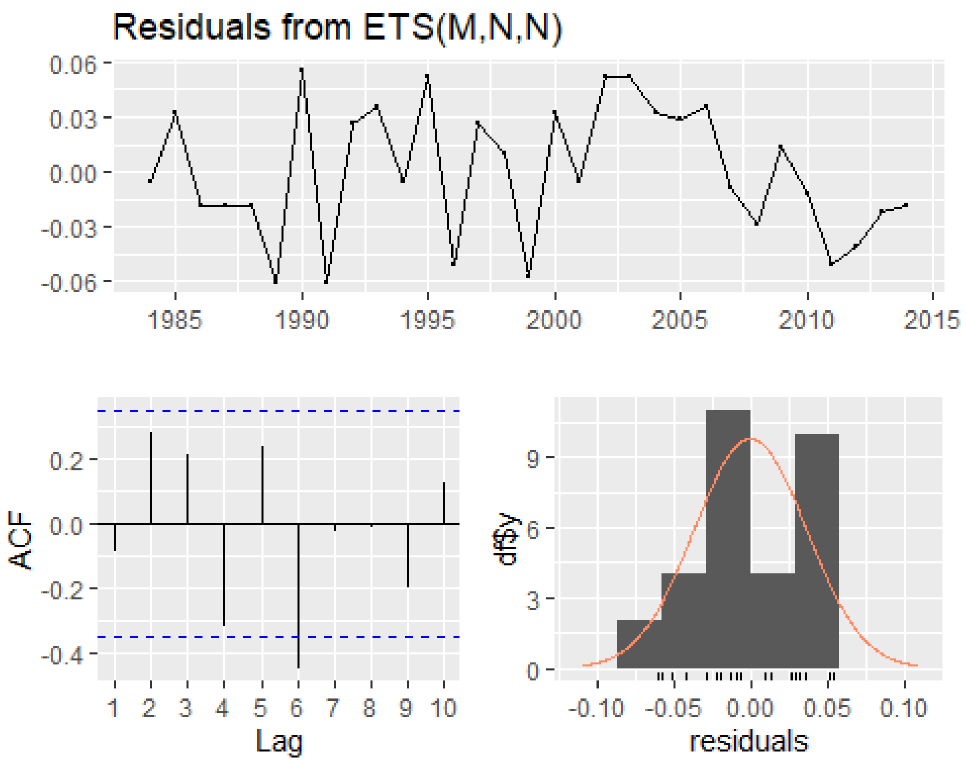
- Given that there is no trend or seasonality in the time series data, we are limited to fitting models with either multiplicative or additive errors. This limitation is primarily due to the absence of clear patterns or systematic variations in the data.

```
ffd_1 <- ets(ffd, model= "MNN")
ffd_2 <- ets(ffd, model= "MAN")
ffd_3 <- ets(ffd, model= "ANN")

auto_ets <- ets(ffd)
auto_ets

## ETS(A,N,N)
##
## Call:
##  ets(y = ffd)
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 311.8397
```

```
##
##   sigma:  11.5528
##
##      AIC     AICc      BIC
## 262.0960 262.9848 266.3979
```

- The automated model fitting process is recommending the use of ETS(A,N,N), which signifies a model comprising an Additive Error, No Trend, No Seasonality.

```
summary(ffd_1)

## 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(ffd_1)
```



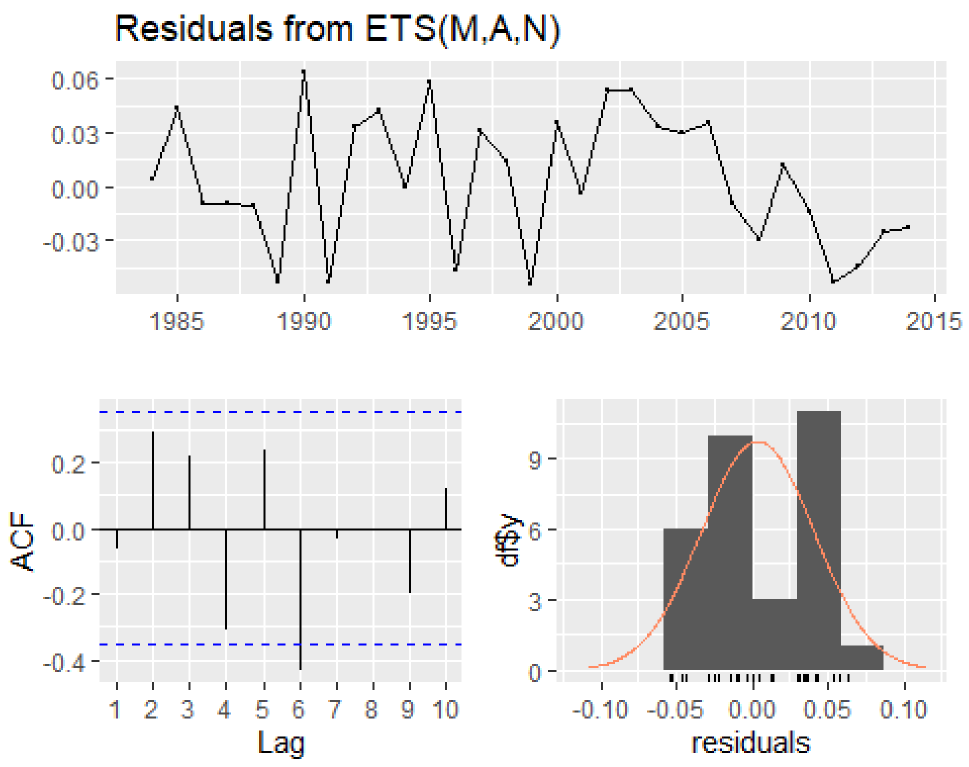
```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,N)
## Q* = 19.27, df = 6, p-value = 0.003731
##
## Model df: 0.   Total lags used: 6
```

```
summary(ffd_2)
```

```
## ETS(M,A,N)
##
## Call:
##  ets(y = ffd, model = "MAN")
##
##   Smoothing parameters:
##     alpha = 1e-04
##     beta  = 1e-04
##
##   Initial states:
##     l = 308.4005
##     b = 0.1449
##
##   sigma:  0.0393
##
##      AIC     AICc      BIC
## 267.3581 269.7581 274.5280
```

```
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 1.082257 11.40017 9.822519 0.2171807 3.148306 0.7634082
##                     ACF1
## Training set -0.06136522
```

```
checkresiduals(ffd_2)
```



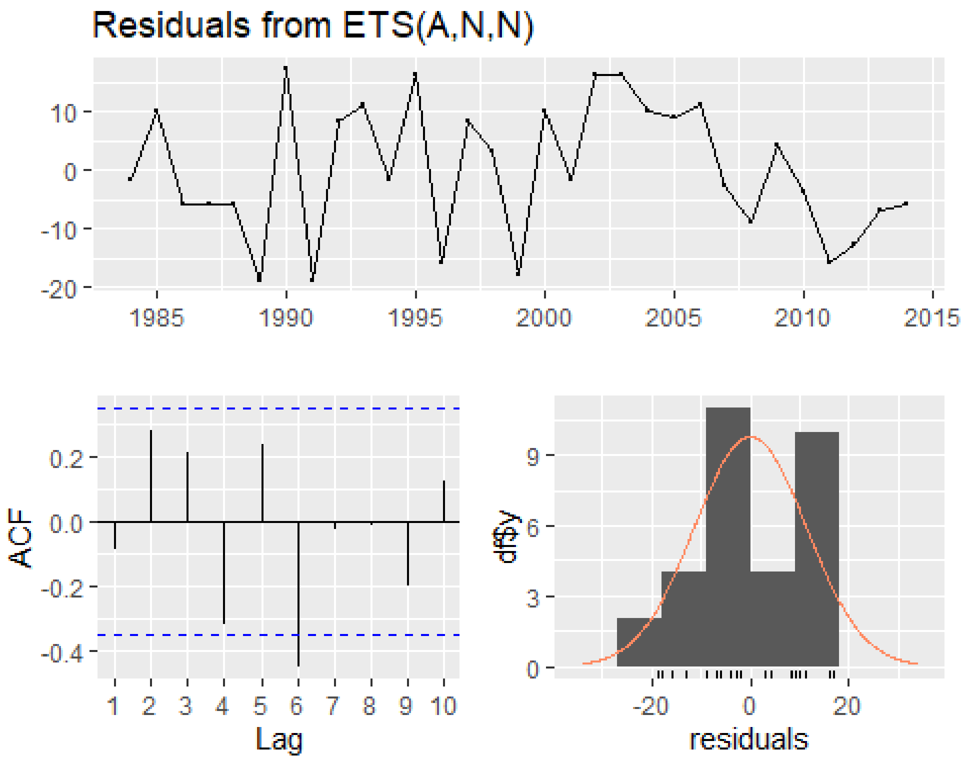
```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,A,N)
## Q* = 18.177, df = 6, p-value = 0.005806
##
## Model df: 0.   Total lags used: 6
```

```
summary(ffd_3)
```

```
## 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(ffd_3)
```

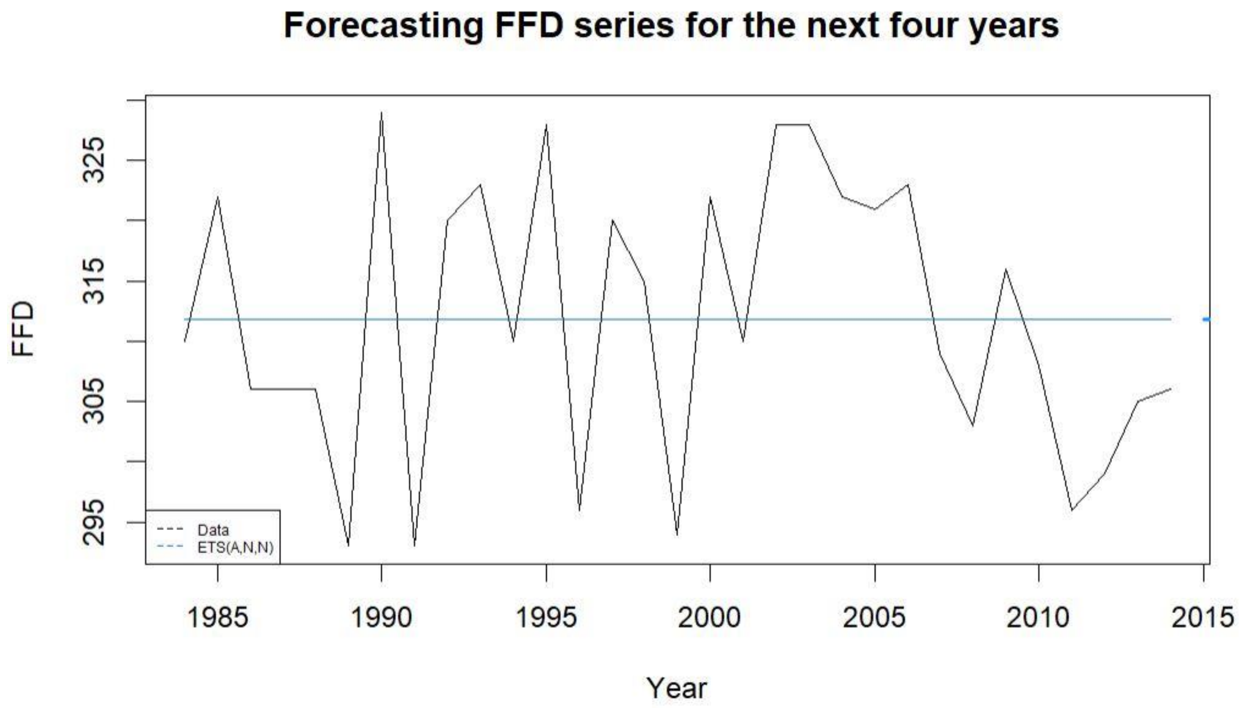


```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,N)
## Q* = 19.27, df = 6, p-value = 0.003731
##
## Model df: 0.   Total lags used: 6
```

When evaluating all the models, it's apparent that none of them provide a satisfactory fit. This conclusion is drawn from several factors:

- The p-values suggest the presence of serial correlation, as they are below the significance level of 0.05, indicating that the models do not adequately account for autocorrelation in the data.
- The Mean Absolute Scaled Error (MASE) values are not sufficiently low when compared to previous models, indicating that the forecasting accuracy is not significantly better with these models.
- The values of AICc, AIC, and BIC are similar to those of the previous exponential models, suggesting that the state space models do not offer a substantial improvement in terms of model fit or complexity.
- Given these observations, it is not possible to assert that the state space models represent a better fit for the data compared to the previous exponential models.
- Forecasting with ANN model

```
frc_ffd_3<- forecast(ffd_3)

plot(ffd, fcol = "white", main = "Forecasting FFD series for the next four years ", ylab =
"Radiation", xlab="Year")

lines(fitted(ffd_3), col = "dodgerblue3")

lines(frc_ffd_3$mean, col = "dodgerblue", lwd = 2)

legend("bottomleft", lty = 2, col = c("black", "dodgerblue3"), c("Data", "ETS(A,N,N)"), , cex =
0.5)
```



* As mentioned earlier, using State Space Models for forecasting is not a suitable option. A closer look at the predicted values confirms its shortcomings, as it consistently generates the same forecasted values for the years 2015, 2016, 2017, and 2018.

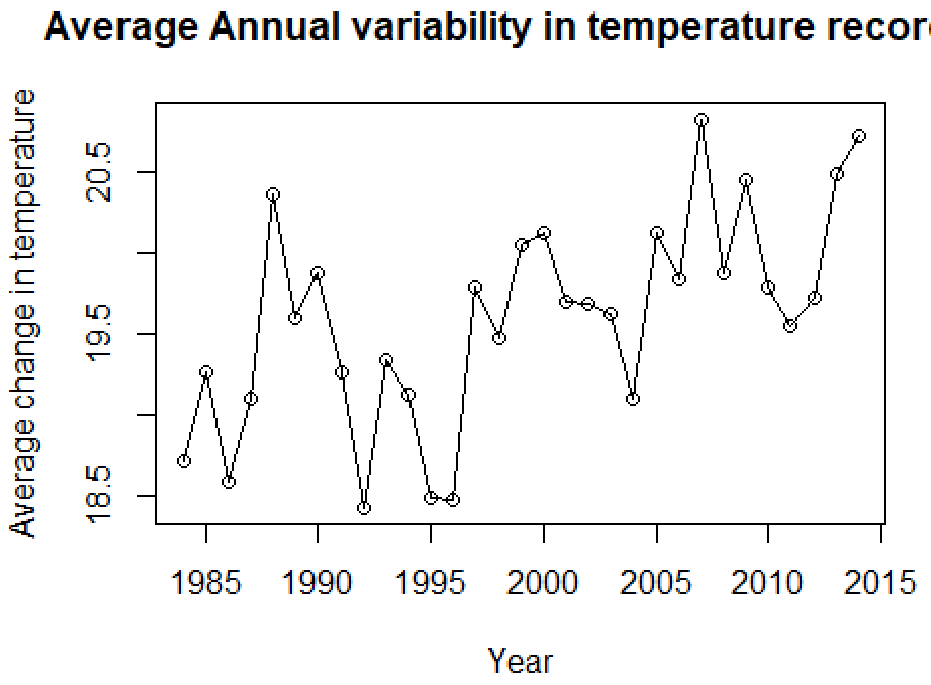
Conclusion:

Through the evaluation of different accuracy measures such as R2, MASE, AIC, and BIC, we determined the best models from each method. Notably, radiation and rainfall consistently emerged as influential climate indicators across these best models. Among the methods employed, finite, poly, koyck, and ARDL & dynlm methods demonstrated their ability to generate accurate and reliable predictions for the four-year ahead values. However, models derived from exponential smoothing (ETS) were found to be unsuitable for predicting the FFD series. The recommended and best-performing model, based on the MASE value, R2 value, F-test.

Task 3

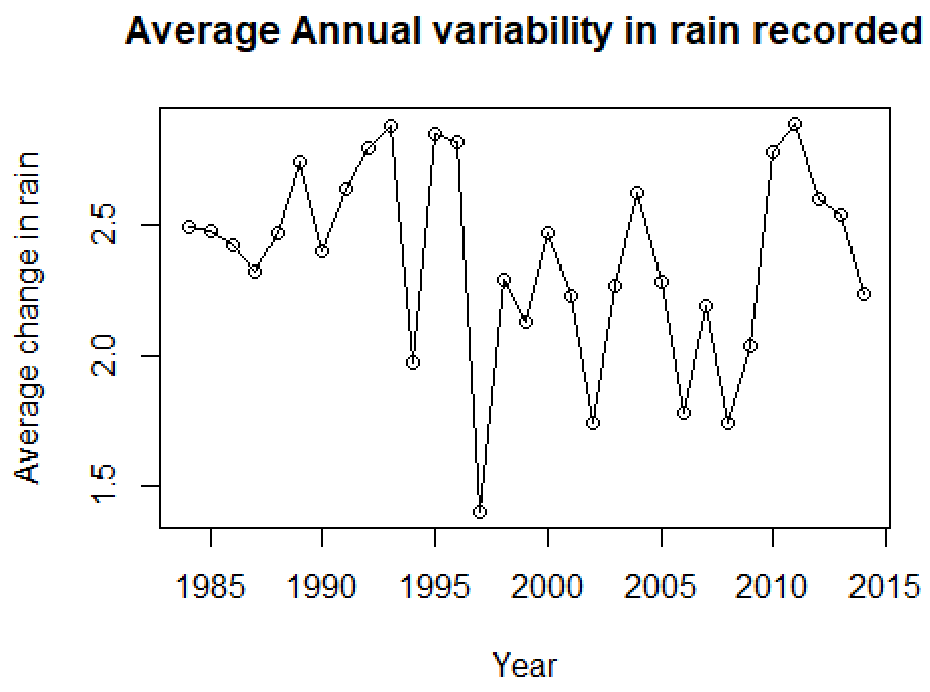
```
RBO<- read.csv("rbo_data.csv")
Covariate_task_3 <- read.csv("Covariate_task_3.csv")

head(RBO)

##   Year       RBO Temperature Rainfall Radiation RelHumidity  X X.1 X.2 X.3 X.4
## 1 1984 0.7550088    18.71038 2.489344  14.87158    54.64891 NA  NA  NA  NA  NA
## 2 1985 0.7407520    19.26301 2.475890  14.68493    54.95781 NA  NA  NA  NA  NA
## 3 1986 0.8423860    18.58356 2.421370  14.51507    54.96301 NA  NA  NA  NA  NA
## 4 1987 0.7484425    19.10137 2.319726  14.67397    53.87205 NA  NA  NA  NA  NA
## 5 1988 0.7984084    20.36066 2.465301  14.74863    53.11885 NA  NA  NA  NA  NA
## 6 1989 0.7938803    19.59589 2.735890  14.78356    55.37671 NA  NA  NA  NA  NA
##   X.5 X.6 X.7
## 1  NA  NA  NA
## 2  NA  NA  NA
## 3  NA  NA  NA
## 4  NA  NA  NA
## 5  NA  NA  NA
## 6  NA  NA  NA

head(Covariate_task_3)

##   Year Temperature Rainfall Radiation RelHumidity
## 1 2015       20.74     2.27     14.60       52.16
## 2 2016       20.49     2.38     14.56       52.87
## 3 2017       20.52     2.26     14.79       52.58
## 4 2018       20.56     2.27     14.79       52.50
## 5   NA          NA       NA        NA          NA
## 6   NA          NA       NA        NA          NA
```

```
rbo_ts <- ts(RBO[,2:6], start = c(1984,1), frequency = 1)
rbo_temp <- ts(RBO$Temperature, start = c(1984,1) , frequency = 1)
rbo_rain <- ts(RBO$Rainfall, start = c(1984,1) , frequency = 1)
rbo_rad <- ts(RBO$Radiation, start = c(1984,1), frequency = 1)
rbo_hum <- ts(RBO$RelHumidity, start = c(1984,1), frequency = 1)
rbo <- ts(RBO$RBO, start = c(1984,1), frequency = 1)

rbo<- na.omit(rbo)
Covariate_task_3<- na.omit(Covariate_task_3)
rbo_temp<-na.omit(rbo_temp)
rbo_rad <- na.omit(rbo_rad)
rbo_hum<- na.omit(rbo_hum)
rbo_rain <- na.omit(rbo_rain)

par(mfrow=c(1,1))
plot(rbo_temp, ylab="Average change in temperature", xlab = "Year", main =
"Average Annual variability in temperature recorded", type ="o")
```



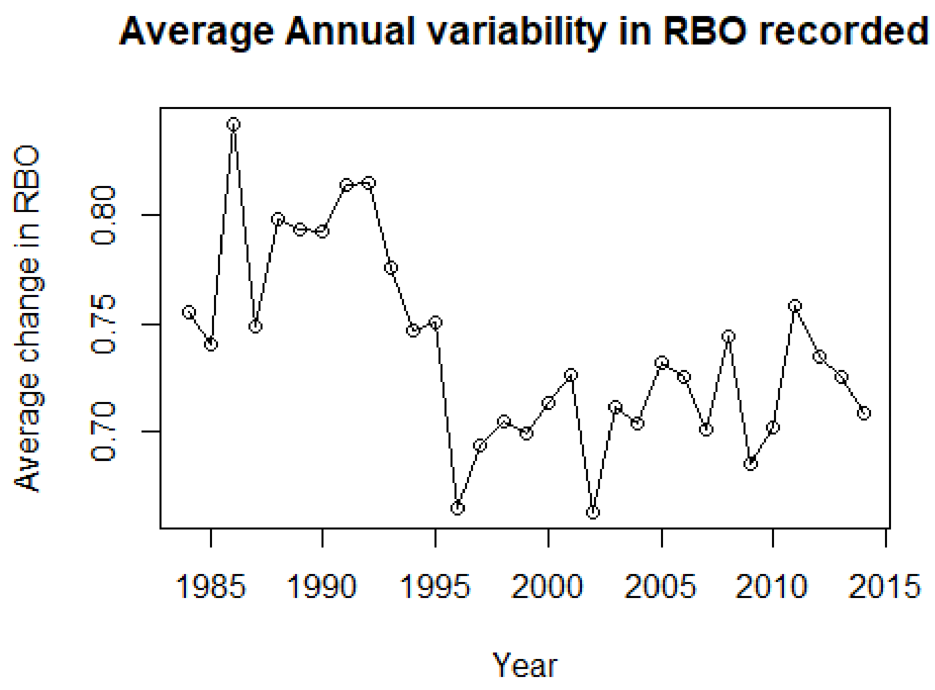
* The time series data shows no clear seasonal pattern with evidence of upward trend or change of variance . However, there is a notable intervention starting from the year 1992, suggesting a significant external effect influenced the data.

```
par(mfrow=c(1,1))
plot(rbo_rain, ylab="Average change in rain", xlab = "Year", main =
"Average Annual variability in rain recorded", type ="o")
```



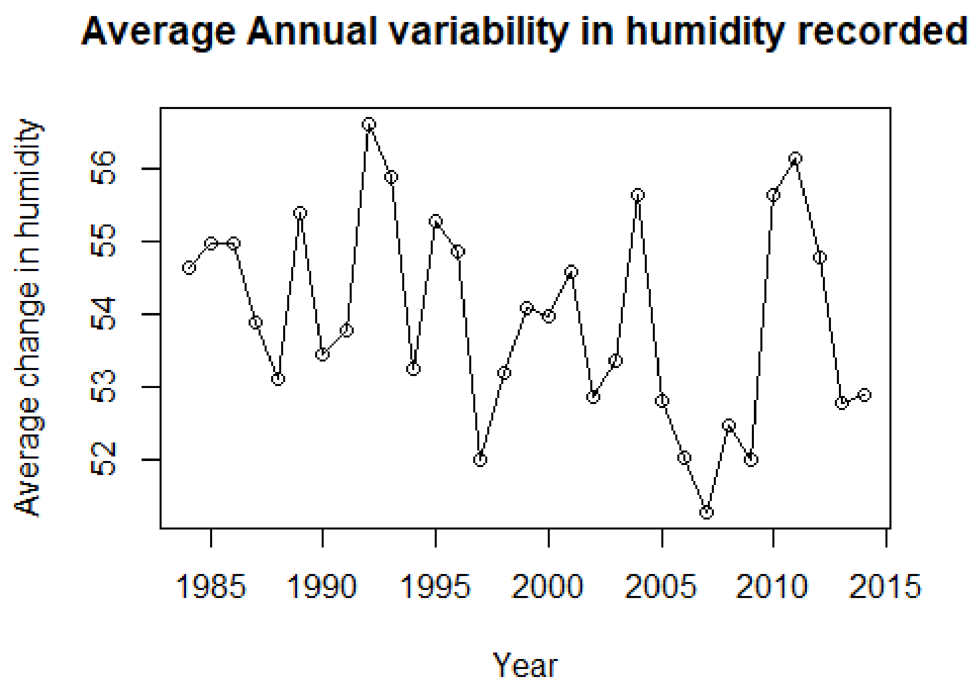
* The time series data shows no clear seasonal pattern with no evidence of trend or change of variance . However, there is a notable intervention in the year 1997, suggesting a significant external effect influenced the data.

```
par(mfrow=c(1,1))
plot(rbo, ylab="Average change in RBO", xlab = "Year", main =
"Average Annual variability in RBO recorded", type ="o")
```



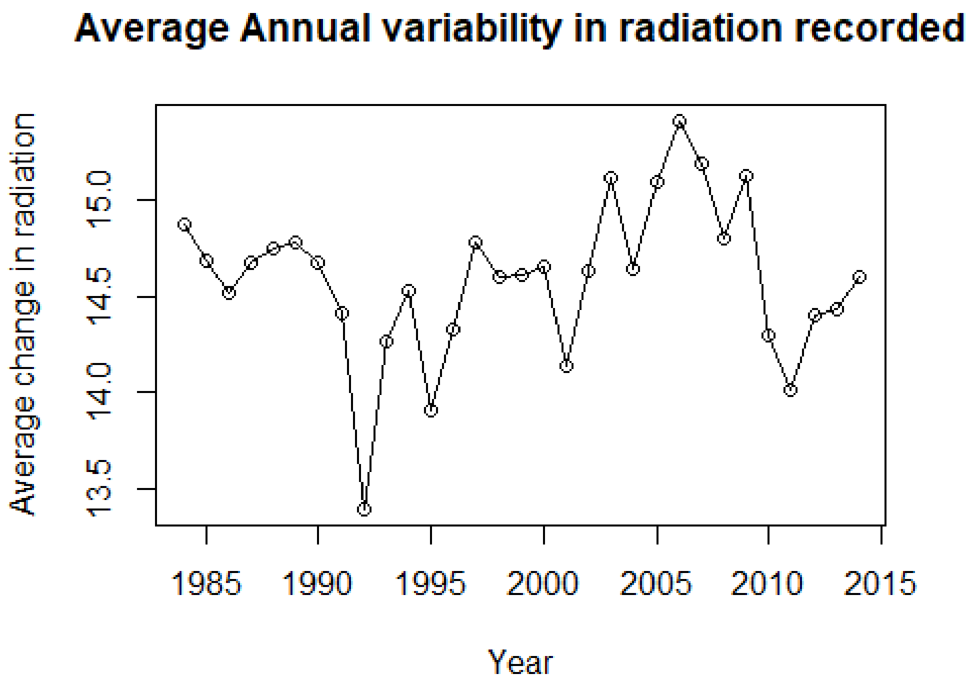
* The time series data shows no clear seasonal pattern but there is evidence of downward trend and change of variance . However, there is a notable intervention starting from the year 1996, suggesting a significant external effect influenced the data.

```
par(mfrow=c(1,1))
plot(rbo_hum, ylab="Average change in humidity", xlab = "Year", main =
"Average Annual variability in humidity recorded", type ="o")
```



* The time series data shows no clear seasonal pattern, there is no evidence of downward trend but evidence of change of variance . However, there are few notable interventions in the year 1997, 2007 and 2103, suggesting a significant external effect influenced the data.

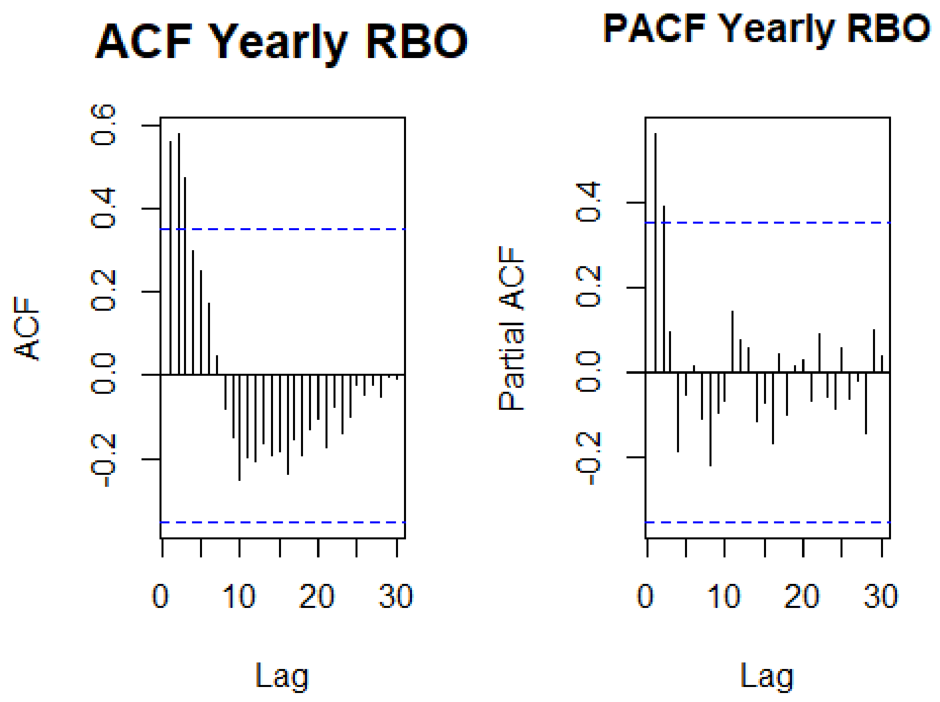
```
par(mfrow=c(1,1))
plot(rbo_rad, ylab="Average change in radiation", xlab = "Year", main =
"Average Annual variability in radiation recorded", type ="o")
```



* The time series data shows no clear seasonal pattern with no evident trend but little change of variance 2000 onward. However, there is a notable intervention between the year 1992-193, suggesting a significant external effect influenced the data.

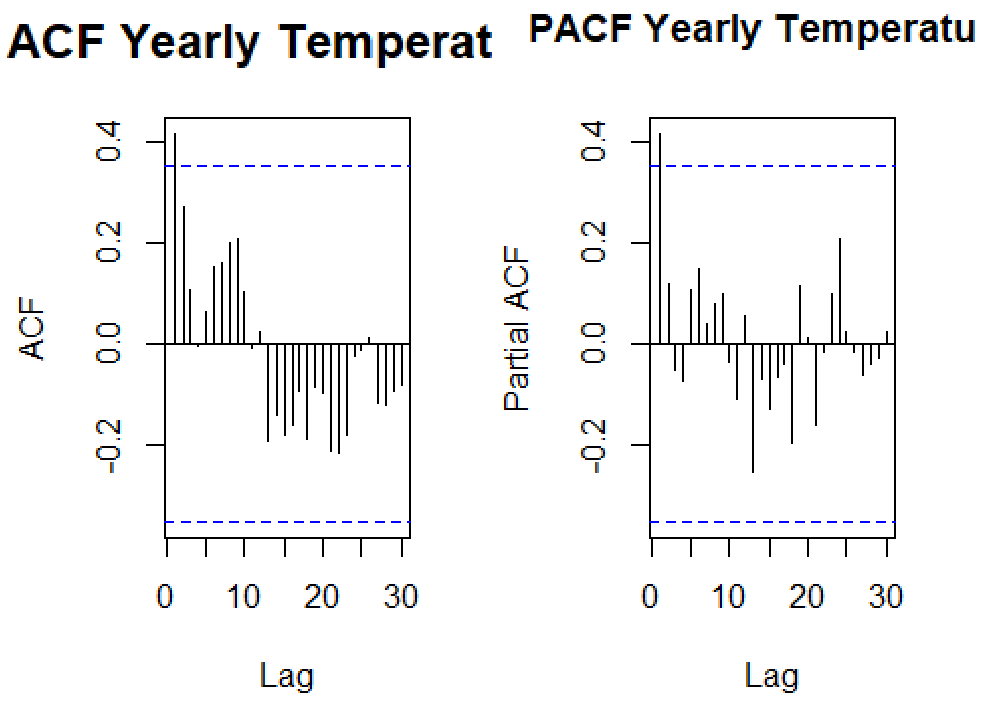
Analysing Non-Stationarity in dataset

```
par(mfrow=c(1,2))
acf(rbo, lag.max = 48, main = "ACF Yearly RBO", cex.main = 1.5)
pacf(rbo, lag.max = 48, main = "PACF Yearly RBO", cex.main = 1.5)
```



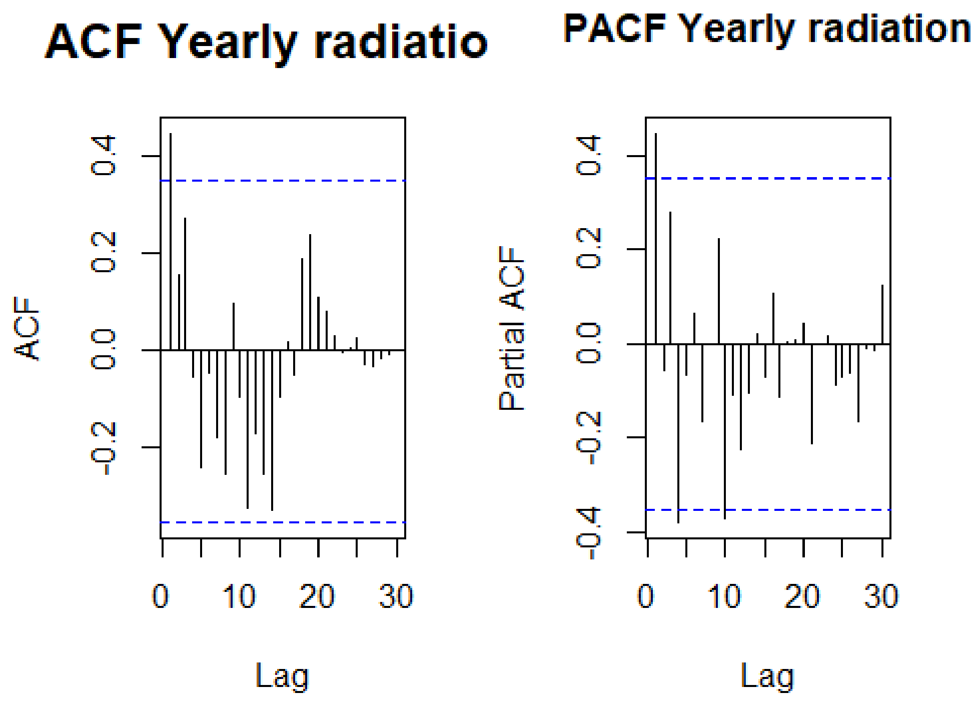
* The ACF plot shows a gradual decline, indicating a noticeable decreasing trend in the time series data. There is no sign of seasonal pattern in this data. On the other hand, when you look at the PACF plot, it's clear that the first few lags are highly significant, suggesting that the series is not stationary and does exhibit autocorrelation.

```
par(mfrow=c(1,2))
acf(rbo_temp , lag.max = 48, main = "ACF Yearly Temperature", cex.main = 1.5)
pacf(rbo_temp , lag.max = 48, main = "PACF Yearly Temperature", cex.main = 1.5)
```



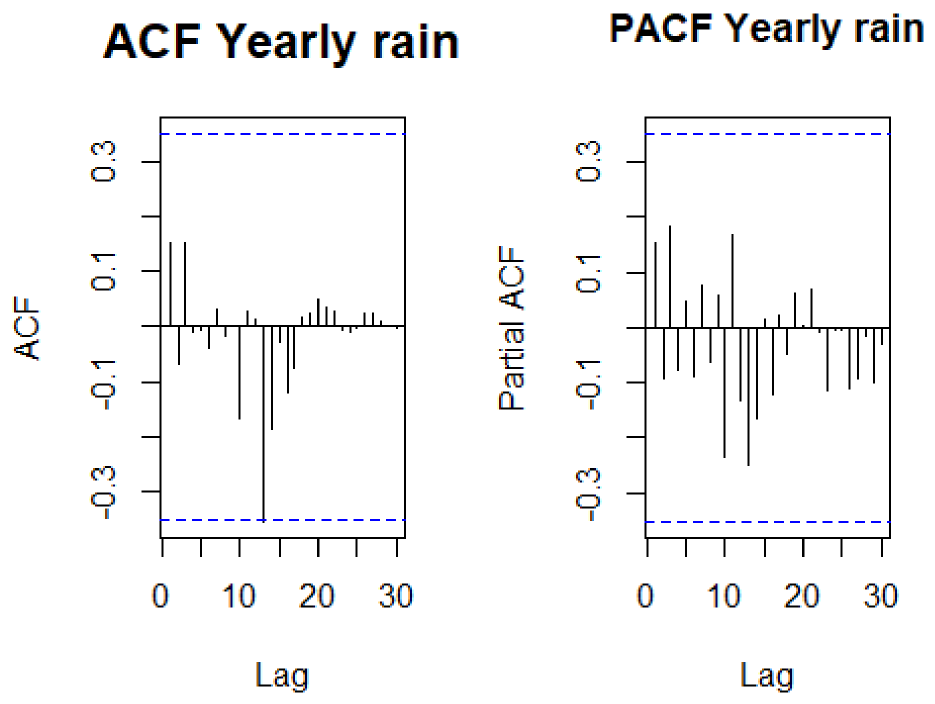
* The ACF plot shows signs of declining pattern, which infer that there is a trend in the series trend in the time series data that will dominate the series correlation. Al the lag values are well below the 95% confidence interval except the first lag.On the other hand, when looking at the PACF plot, there is only one lag that appears highly significant, suggesting that the series is not stationary and does exhibit autocorrelation.

```
par(mfrow=c(1,2))
acf(rbo_rad, lag.max = 48, main = "ACF Yearly radiation", cex.main = 1.5)
pacf(rbo_rad, lag.max = 48, main = "PACF Yearly radiation", cex.main = 1.5)
```



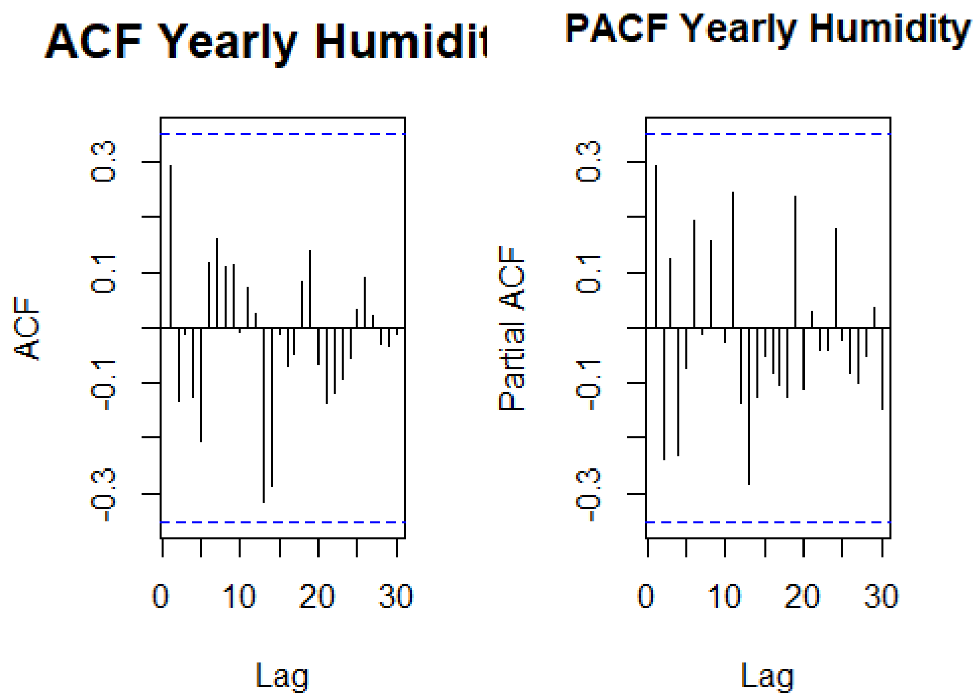
* The ACF plot shows signs of declining and inclining pattern, which infer that there is trend in the series trend in the time series data that will dominate the series correlation. Al the lag values are well below the 95% confidence interval except the first lag.On the other hand, when looking at the PACF plot, there is are lags that appears highly significant, suggesting that the series is not stationary and does exhibit autocorrelation.

```
par(mfrow=c(1,2))
acf(rbo_rain, lag.max = 48, main = "ACF Yearly rain", cex.main = 1.5)
pacf(rbo_rain, lag.max = 48, main = "PACF Yearly rain", cex.main = 1.5)
```



* The ACF plot shows no signs of declining pattern, which infer that there is no trend in the series that will dominate the series correlation. Al the lag values are well below the 95% confidence interval.On the other hand, when looking at the PACF plot, there is no lag that appears highly significant, suggesting that the series is stationary and does not exhibit autocorrelation.

```
par(mfrow=c(1,2))
acf(rbo_hum, lag.max = 48, main = "ACF Yearly Humidity", cex.main = 1.5)
pacf(rbo_hum, lag.max = 48, main = "PACF Yearly Humidity", cex.main = 1.5)
```



* The ACF plot shows no signs of declining pattern, which infer that there is no trend in the series that will dominate the series correlation. Looking at the PACF plot, there is no lag that appears highly significant, all the lag values are well below the 95% confidence interval.On the other hand, suggesting that the series is stationary and does not exhibit autocorrelation.

```
k <- ar(rbo)$order
adf.test(rbo, k = k)

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

* In this situation, with a lag order of 2 and a p-value of 0.7829, we cannot reject the null hypothesis. This essentially means that the RBO series is not stationary. In other words,it has time dependent structure and does have constant variance over time. This demonstrates the time-dependent patterns and its variance is not uniform over time.

```
k <- ar(rbo_temp)$order
adf.test(rbo_temp, k = k)

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

 * In this situation, with a lag order of 1 and a p-value of 0.1902, we cannot reject the null hypothesis. This essentially means that the temp series is not stationary. In other words,it has time dependent structure and does have constant variance over time. This demonstrates the time-dependent patterns and its variance is not uniform over time.

```
k <- ar(rbo_hum)$order
adf.test(rbo_hum, k = k)
```

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

 * In this situation, with a lag order of 1 and a p-value of 0.01789, we can reject the null hypothesis. This essentially means that the temp series is stationary. In other words,it has time no dependent structure and does not have constant variance over time. This demonstrates no time-dependent patterns and its variance is uniform over time.

```
k <- ar(rbo_rad)$order
adf.test(rbo_rad, k = k)
```

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

```
k <- ar(rbo_rain)$order
adf.test(rbo_rain, k = k)
```

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

 * In this situation, with a lag order of 0 and a p-value of 0.01, we can reject the null hypothesis. This essentially means that all the time series are stationary. In other words,it has no time dependent structure and does have constant variance over time. This demonstrates the time-dependent patterns and its variance is uniform over time.

- Given that the series displays non-stationarity without any indications of seasonality, the most appropriate course of action is to employ the ordinary differencing operation.

```
rbo_temp_diff <- diff(rbo_temp)
```

```r
rbo_diff <- diff(rbo)

rbo_rad_diff <- diff(rbo_rad)

rbo_hum_diff <- diff(rbo_hum)

adf.test(rbo_diff)

##
##  Augmented Dickey-Fuller Test
##
## data:  rbo_diff
## Dickey-Fuller = -2.7645, Lag order = 3, p-value = 0.279
## alternative hypothesis: stationary

adf.test(rbo_temp_diff)

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

adf.test(rbo_rad_diff)

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

adf.test(rbo_hum_diff)

##
##  Augmented Dickey-Fuller Test
##
## data:  rbo_hum_diff
## Dickey-Fuller = -3.1232, Lag order = 3, p-value = 0.1412
## alternative hypothesis: stationary
```

 * 1st order difference did not seem to have much effect on stationarity, now lets try the second order difference.

```r
rbo_temp_diff_3 <- diff(rbo_temp, differences = 3)
rbo_hum_diff_3 <- diff(rbo_hum, differences = 3)
rbo_diff_3 <- diff(rbo, differences = 3)
rbo_rad_diff_3 <- diff(rbo_rad, differences = 3)

adf.test(rbo_temp_diff_3)

##
##  Augmented Dickey-Fuller Test
```

```
##
## data:  rbo_temp_diff_3
## Dickey-Fuller = -4.0851, Lag order = 3, p-value = 0.01979
## alternative hypothesis: stationary
```

```
adf.test(rbo_hum_diff_3)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  rbo_hum_diff_3
## Dickey-Fuller = -4.1215, Lag order = 3, p-value = 0.0185
## alternative hypothesis: stationary
```

```
adf.test(rbo_diff_3)
```

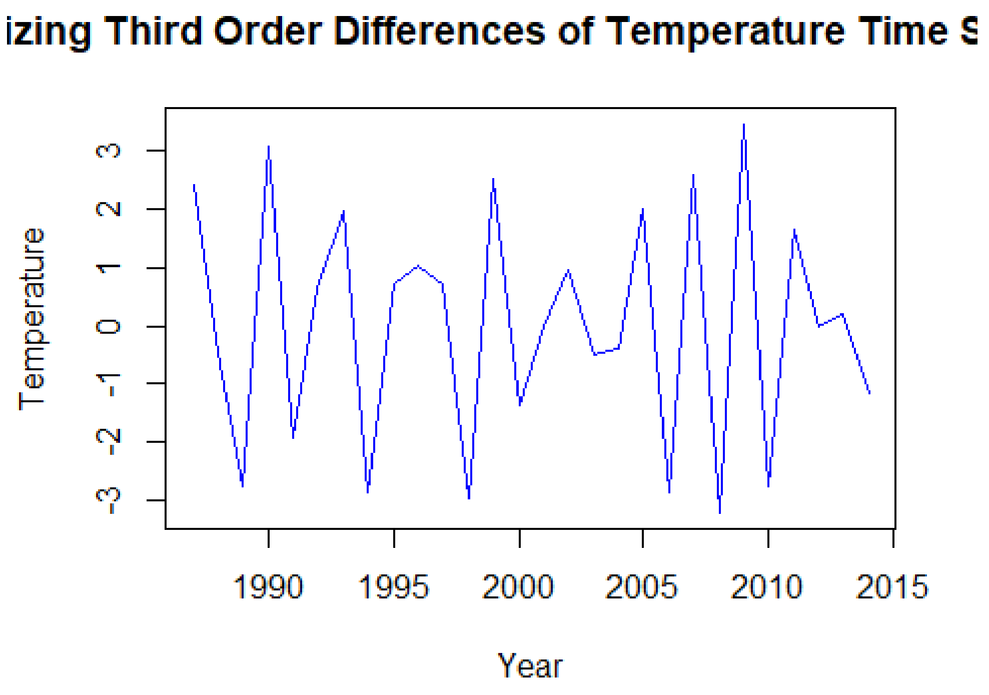
```
##
##  Augmented Dickey-Fuller Test
##
## data:  rbo_diff_3
## Dickey-Fuller = -4.0249, Lag order = 3, p-value = 0.02192
## alternative hypothesis: stationary
```

```
adf.test(rbo_rad_diff_3)
```

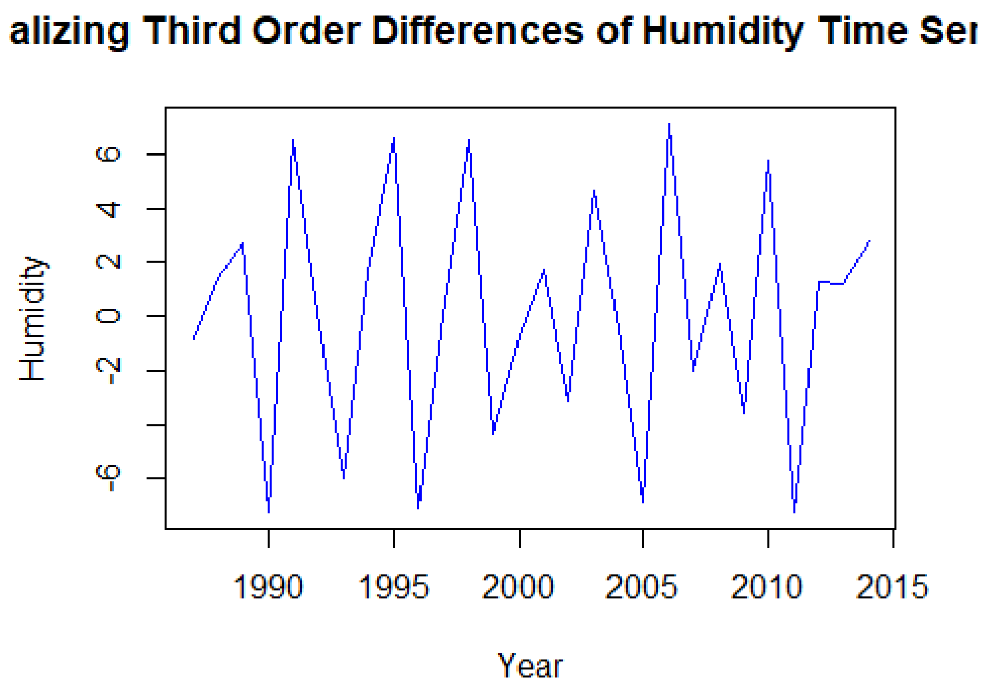
```
##
##  Augmented Dickey-Fuller Test
##
## data:  rbo_rad_diff_3
## Dickey-Fuller = -4.1613, Lag order = 3, p-value = 0.01709
## alternative hypothesis: stationary
```

 * 2nd order difference seem to have much effect on stationarity, all the p-values are less than 0.05. This indicates, series are stationary.

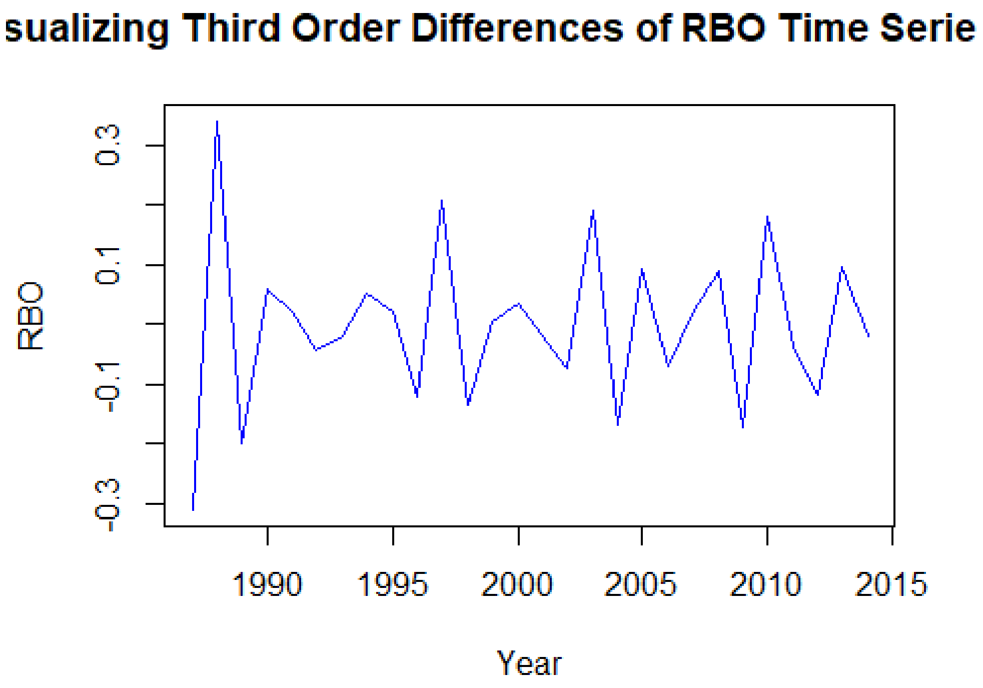
```
plot(rbo_temp_diff_3, xlab = "Year", main =
"Visualizing Third Order Differences of Temperature Time Series Data", ylab = "Temperature", col =
"blue")
```



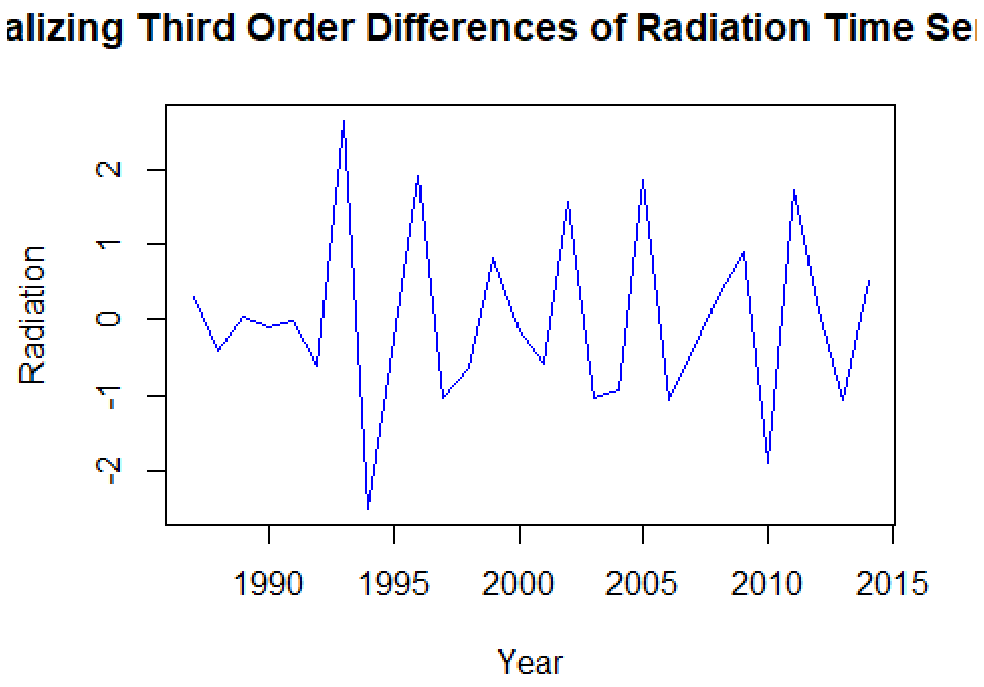
```
plot(rbo_hum_diff_3, xlab = "Year", main =
"Visualizing Third Order Differences of Humidity Time Series Data", ylab = "Humidity", col = "blue")
```



```
plot(rbo_diff_3, xlab = "Year", main =
"Visualizing Third Order Differences of RBO Time Series Data", ylab = "RBO", col = "blue")
```



```
plot(rbo_rad_diff_3, xlab = "Year", main =
"Visualizing Third Order Differences of Radiation Time Series Data", ylab = "Radiation", col = "blue")
```



* Considering the outcome of the ADF test and upon visualizing the data,after 1st and 2nd order differencing did not seems to alter all the series transformation in terms of its non-stationary in nature. therefore third order differencing was taken into consideration the humidity, radiation and ffd series. In addition the third order differencing seems to be more appropriate for radiation series to make the series stationary.

Modelling utlizing distributed Lag Models

Finite DLM

```
finiteDLMauto(x=as.vector(rbo_temp), 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
```

```
finiteDLMauto(x=as.vector(rbo_rain), 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 0.94180 -100.89798 -95.29319 0.94415  0.46484  0.15753 0.04687413
## 3      3 0.97969  -97.19966 -89.20643 0.86453  0.97573  0.18688 0.01249933
## 2      2 0.99937  -96.70956 -89.87308 0.83164  0.44840  0.19180 0.03708415
## 4      4 1.03883  -90.46187 -81.39101 0.86444  0.78209  0.14281 0.02073547
## 5      5 0.92568  -87.24242 -77.17765 0.76943  0.47689  0.12600 0.02871866
## 6      6 0.85440  -82.31788 -71.34800 0.51455 27.53976  0.04227 0.04784411
## 9      9 0.62059  -80.79432 -67.70181 0.54274 -0.14207 -0.22123 0.84870967
## 7      7 0.82934  -77.98405 -66.20351 0.77764  0.94056 -0.04849 0.17026081
## 10    10 0.58562  -76.93255 -63.35376 0.53594  0.09998 -0.49754 0.52875140
## 8      8 0.72338  -76.81922 -64.32879 0.62316  0.02713 -0.17396 0.59232039
```

```
finiteDLMauto(x=as.vector(rbo_rain), 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 0.94180 -100.89798 -95.29319 0.94415  0.46484  0.15753 0.04687413
## 3      3 0.97969  -97.19966 -89.20643 0.86453  0.97573  0.18688 0.01249933
## 2      2 0.99937  -96.70956 -89.87308 0.83164  0.44840  0.19180 0.03708415
## 4      4 1.03883  -90.46187 -81.39101 0.86444  0.78209  0.14281 0.02073547
## 5      5 0.92568  -87.24242 -77.17765 0.76943  0.47689  0.12600 0.02871866
## 6      6 0.85440  -82.31788 -71.34800 0.51455 27.53976  0.04227 0.04784411
## 9      9 0.62059  -80.79432 -67.70181 0.54274 -0.14207 -0.22123 0.84870967
## 7      7 0.82934  -77.98405 -66.20351 0.77764  0.94056 -0.04849 0.17026081
## 10    10 0.58562  -76.93255 -63.35376 0.53594  0.09998 -0.49754 0.52875140
## 8      8 0.72338  -76.81922 -64.32879 0.62316  0.02713 -0.17396 0.59232039
```

```
finiteDLMauto(x=as.vector(rbo_hum), 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 0.98587 -98.90083 -93.29604 0.95516   0.52535  0.09953 0.0082122276
## 2      2 1.10265 -94.63327 -87.79679 1.08715  -2.50595  0.13181 0.0035155519
## 3      3 1.11836 -93.20827 -85.21504 1.15099  -0.01100  0.06230 0.0005534027
## 4      4 1.18147 -86.75196 -77.68110 1.17221 -10.23049  0.01655 0.0013029391
## 5      5 1.07137 -84.46252 -74.39775 1.05294   0.08338  0.02737 0.0014758201
## 6      6 0.98763 -79.22764 -68.25775 0.84701   0.58433 -0.08375 0.0051118546
## 9      9 0.65074 -78.44362 -65.35111 0.50720  -0.03750 -0.35895 0.8294536723
## 7      7 0.89202 -78.33745 -66.55691 0.75348  -2.66227 -0.03316 0.0318380443
## 10    10 0.59476 -77.84443 -64.26564 0.61868   0.82194 -0.43390 0.3196091222
## 8      8 0.74046 -77.57454 -65.08410 0.68069  -1.27782 -0.13604 0.1752809505
```

- After conducting a comprehensive analysis that took into account factors like MASE, AIC, and BIC, we have determined that a lag length of 10 is the optimal choice. It is evident that as the lag length increases, there is consistent improvement in these metrics. Based on this insight, we can conclude that implementing a finite Dynamic Linear Model (DLM) is a favorable decision

Modelling with intercept

```
model_rbo_temp_finite = dlm(x = as.vector(rbo_temp), y = as.vector(rbo),
q =1)

model_rbo_hum_finite = dlm(x = as.vector(rbo_hum), y = as.vector(rbo),
q =1)

model_rbo_rain_finite = dlm(x = as.vector(rbo_rain), y = as.vector(rbo),
q =1)

model_rbo_rad_finite = dlm(x = as.vector(rbo_rad), y = as.vector(rbo),
q =1)
```

Modelling without intercept

```
rbo_ts %>% head

## Time Series:
## Start = 1984
## End = 1989
## Frequency = 1
##            RBO Temperature Rainfall Radiation RelHumidity
## 1984 0.7550088    18.71038 2.489344  14.87158    54.64891
## 1985 0.7407520    19.26301 2.475890  14.68493    54.95781
## 1986 0.8423860    18.58356 2.421370  14.51507    54.96301
## 1987 0.7484425    19.10137 2.319726  14.67397    53.87205
## 1988 0.7984084    20.36066 2.465301  14.74863    53.11885
## 1989 0.7938803    19.59589 2.735890  14.78356    55.37671

model_rbo_rain_finite_noin = dlm(formula= RBO~Rainfall-1, data = data.frame(rbo_ts), q = 1)

model_rbo_temp_finite_noin = dlm(formula= RBO~Temperature -1, data = data.frame(rbo_ts), q = 1)

model_rbo_rad_finite_noin = dlm(formula= RBO~Radiation -1, data = data.frame(rbo_ts), q = 1)

model_rbo_hum_finite_noin = dlm(formula= RBO~RelHumidity -1, data = data.frame(rbo_ts), q = 1)

sort.score <- function(x, score = c("bic", "aic", "mase")) {
  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")')
  }
}

sort.score(AIC(model_rbo_temp_finite$model,model_rbo_hum_finite$model,
model_rbo_rain_finite$model,model_rbo_rad_finite$model,
model_rbo_rain_finite_noin$model,
model_rbo_temp_finite_noin$model,model_rbo_rad_finite_noin$model,
model_rbo_hum_finite_noin$model), score = "aic")

##                                  df        AIC
## model_rbo_rain_finite$model       4 -100.89798
## model_rbo_hum_finite_noin$model   3 -100.84952
## model_rbo_hum_finite$model        4  -98.90083
## model_rbo_rad_finite$model        4  -97.71113
## model_rbo_temp_finite$model       4  -97.52442
## model_rbo_rad_finite_noin$model   3  -89.33439
## model_rbo_temp_finite_noin$model  3  -86.53783
## model_rbo_rain_finite_noin$model  3  -62.07890

sort.score(BIC(model_rbo_temp_finite$model,model_rbo_hum_finite$model,
model_rbo_rain_finite$model,model_rbo_rad_finite$model,
model_rbo_rain_finite_noin$model,
model_rbo_temp_finite_noin$model,model_rbo_rad_finite_noin$model,
model_rbo_hum_finite_noin$model), score = "bic")

##                                  df       BIC
## model_rbo_hum_finite_noin$model   3 -96.64593
## model_rbo_rain_finite$model       4 -95.29319
## model_rbo_hum_finite$model        4 -93.29604
## model_rbo_rad_finite$model        4 -92.10634
## model_rbo_temp_finite$model       4 -91.91963
## model_rbo_rad_finite_noin$model   3 -85.13079
## model_rbo_temp_finite_noin$model  3 -82.33424
## model_rbo_rain_finite_noin$model  3 -57.87530

sort.score(MASE(model_rbo_temp_finite$model,model_rbo_hum_finite$model,
model_rbo_rain_finite$model,model_rbo_rad_finite$model,
model_rbo_rain_finite_noin$model,
model_rbo_temp_finite_noin$model,model_rbo_rad_finite_noin$model,
model_rbo_hum_finite_noin$model), score = "mase")

##                                   n      MASE
## model_rbo_rain_finite$model      30 0.9417954
## model_rbo_hum_finite$model       30 0.9858746
## model_rbo_hum_finite_noin$model  30 0.9918515
## model_rbo_temp_finite$model      30 1.0056219
## model_rbo_rad_finite$model       30 1.0630293
## model_rbo_rad_finite_noin$model  30 1.3296554
## model_rbo_temp_finite_noin$model 30 1.4121567
## model_rbo_rain_finite_noin$model 30 1.9110310
```

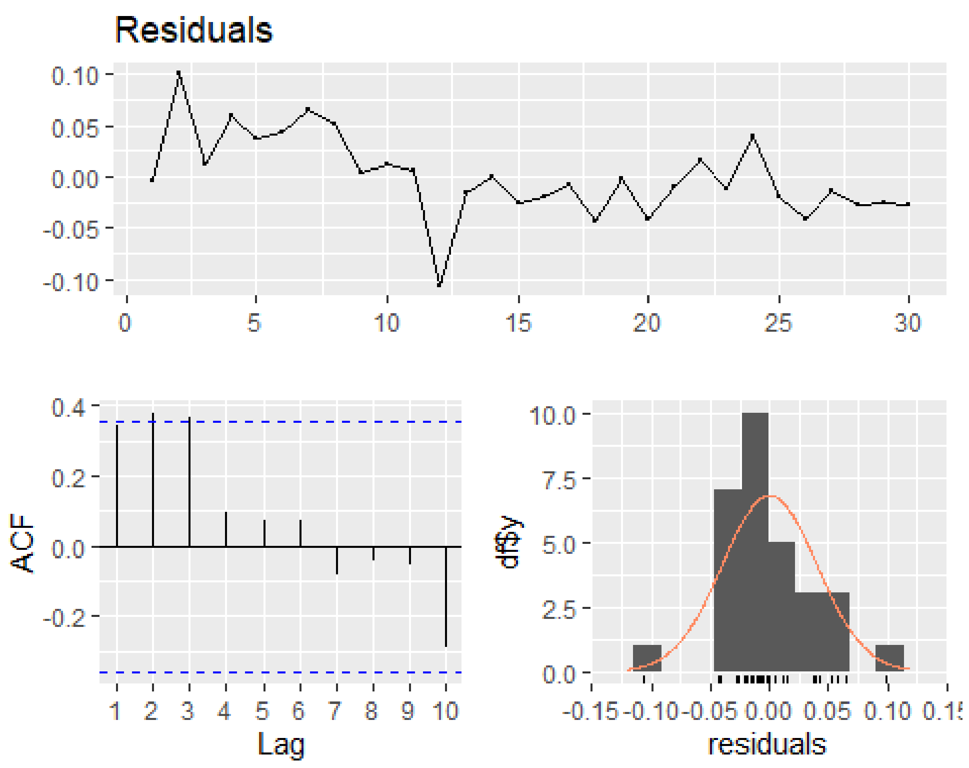
- Considering the result of AIC and MASE model_rbo_rain_finite model is the best model. The model with model_rbo_rain_finite as the predictor stands out as the most suitable choice according to MASE value. Therefore, this model will undergo further analysis for diagnostic checking. Based on BIC model_rbo_hum_finite_noin is also a better model.

```
summary(model_rbo_rain_finite)
```

```
##
## 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
```

 * The model_rbo_rain_finite is not a good fit, as most of the lag variables are statistically insignificant. Additionally, the adjusted R-squared value is very low indicating that the model can explain 15.75% and considering the Multiple R-squared it can explain 21.56% of the variation in the dependent variable. The AIC and BIC values for the model are -100.898 and -95.29319 , respectively. The Residual Standard Error (RSE) is 4.1%, and the range of residuals spans from a minimum value of -0.105903 to a maximum value of 0.099699. This signifies that, on average, the residuals deviate from the model's predictions by approximately 0.19 units. A lower RSE indicates a more precise fit with p value less than 0.05 indicating the model is significant.

```r
checkresiduals(model_rbo_rain_finite$model)
```



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

```
vif(model_rbo_rain_finite$model)>10
```

```
##   x.t   x.1
## FALSE FALSE
```

 * Examining the VIF (Variance Inflation Factor) values for the lag variables within model7_rbo__hum_finite_in, they are all well below the threshold of 10, suggesting that multicollinearity is not a significant issue in the model.

- Taking into account the results of the Breusch-Godfrey test, the p-value is above the 5% significance level, leading to the acceptance of the null hypothesis. This implies there is no existence of serial correlation within the residuals. Further examination of the residuals resulting from the finite dynamic linear model (DLM) fit reveals that they do exhibit a random distribution. Specifically, the autocorrelation function (ACF) confirming the presence of serial correlation in the residuals. Additionally, the histograms indicate a departure from normal distribution.

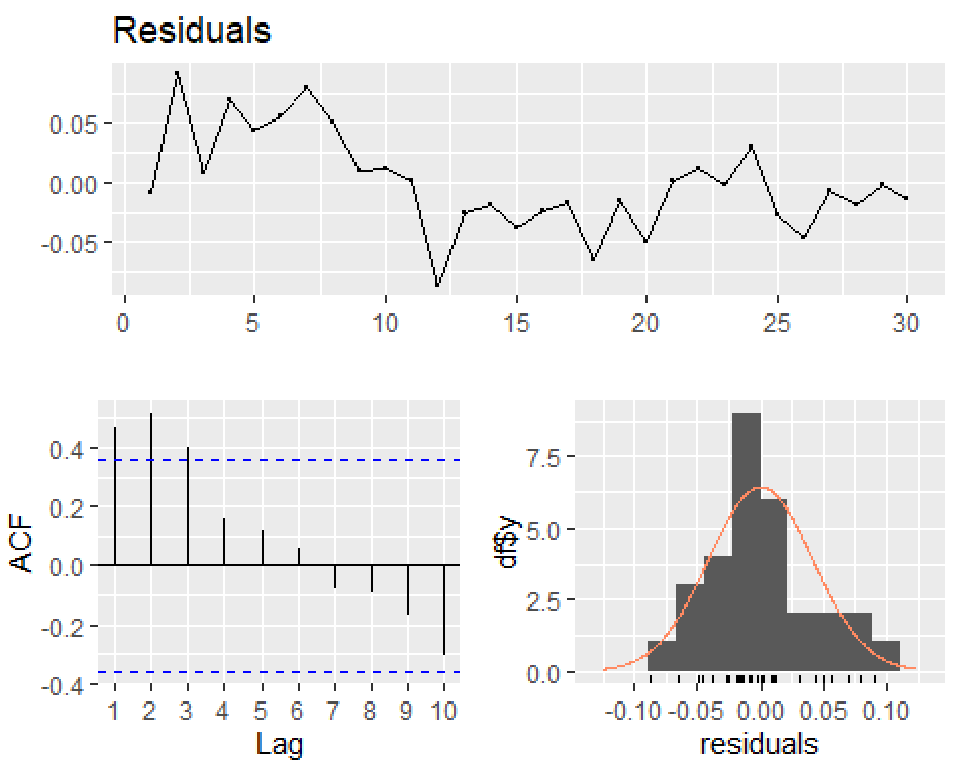
```
summary(model_rbo_hum_finite_noin)
```

```
##
## 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|)
## RelHumidity.t 0.010456   0.004723   2.214   0.0351 *
## RelHumidity.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
##   (5 observations deleted due to missingness)
## 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
```

* The model7_rbo__hum_finite_in is a good fit, the adjusted R-squared value is very high indicating that the model can explain 99.7% and considering the Multiple R-squared it can explain 99.7% of the variation in the dependent variable. The AIC and BIC values for the model are -100.8495 and -96.64593 respectively. The Residual Standard Error (RSE) is 4.2%, and the range of residuals spans from a minimum value of -0.086693 to a maximum value of 0.091105. This signifies that, on average, the residuals deviate from the model's predictions by approximately 0.17 units. A lower RSE indicates a more precise fit with p value less than 0.05 indicating the model is significant.

```
checkresiduals(model_rbo_hum_finite_noin$model)
```



```
##
## 	Breusch-Godfrey test for serial correlation of order up to 6
##
## data:  Residuals
## LM test = 12.005, df = 6, p-value = 0.06186
```

```
vif(model_rbo_hum_finite_noin$model)
```

```
## RelHumidity.t RelHumidity.1 
##      1093.203      1093.203
```

 * Examining the VIF (Variance Inflation Factor) values for the lag variables within model7_rbo__hum_finite_in, they are all well above the threshold of 10, suggesting that multicollinearity is a significant issue in the model.

- Taking into account the results of the Breusch-Godfrey test, the p-value is more than the 5% significance level, leading to the acceptance of the null hypothesis. This implies there is no existence of serial correlation within the residuals. Further examination of the residuals resulting from the model fit reveals that they do not exhibit a random distribution. Specifically, the autocorrelation function (ACF) confirming the presence of serial correlation in the residuals. Additionally, the histograms indicate a departure from normal distribution.

#Polynomial DLM

```
finiteDLMauto(x = as.vector(rbo_rain), y = as.vector(rbo), 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 0.64518  -88.26525 -83.04264 0.54703  1.21414  0.00987 0.24089089
## 9   9 - 2 0.66465  -90.82864 -85.37342 0.54579  0.71453  0.10628 0.64416480
## 8   8 - 2 0.71953  -87.81723 -82.13976 0.55123 -3.16107  0.16100 0.73744124
## 7   7 - 2 0.83196  -86.84712 -80.95686 0.75983  0.73645  0.17548 0.21710348
## 6   6 - 2 0.87732  -89.17565 -83.08127 0.78988  0.42199  0.18845 0.07236669
## 5   5 - 2 0.92642  -92.46231 -86.17182 0.87299  7.18996  0.22219 0.04046159
## 1   1 - 2 0.94180 -100.89798 -95.29319 0.94415  0.46484  0.15753 0.04687413
## 3   3 - 2 0.94916  -98.84343 -92.18241 0.75174  0.19596  0.21078 0.01159322
## 2   2 - 2 0.99937  -96.70956 -89.87308 0.83164  0.44840  0.19180 0.03708415
## 4   4 - 2 1.00619  -94.04271 -87.56352 0.81974 -9.48452  0.20510 0.01870122
```

* Considering the findings presented in the previous code, a lag length of 9 will be employed, and the polynomial order will be modified to 2.The second order polynomial can capture non-linear trend and inflection data and tends perform well in complex patterns.

```
model_poly_rbo_temp <- polyDlm(x=as.vector(rbo_temp), y=as.vector(rbo), q=10, k = 2)

## Estimates and t-tests for beta coefficients:
##          Estimate Std. Error t value P(>|t|)
## beta.0  -0.003340    0.00592 -0.5640   0.584
## beta.1  -0.000509    0.00394 -0.1290   0.900
## beta.2   0.001700    0.00282  0.6030   0.559
## beta.3   0.003290    0.00260  1.2600   0.232
## beta.4   0.004270    0.00285  1.5000   0.163
## beta.5   0.004630    0.00310  1.4900   0.164
## beta.6   0.004370    0.00322  1.3500   0.203
## beta.7   0.003490    0.00332  1.0500   0.316
## beta.8   0.001990    0.00372  0.5360   0.602
## beta.9  -0.000120    0.00473 -0.0253   0.980
## beta.10 -0.002850    0.00647 -0.4400   0.668

model_poly_rbo_rad <- polyDlm(x=as.vector(rbo_rad), y=as.vector(rbo), q=10, k = 2)

## Estimates and t-tests for beta coefficients:
##          Estimate Std. Error t value P(>|t|)
## beta.0  -0.005740    0.00831  -0.691   0.504
## beta.1  -0.002240    0.00532  -0.422   0.681
## beta.2   0.000648    0.00369   0.176   0.864
## beta.3   0.002930    0.00346   0.847   0.415
## beta.4   0.004600    0.00377   1.220   0.248
## beta.5   0.005650    0.00389   1.450   0.174
## beta.6   0.006100    0.00367   1.660   0.125
## beta.7   0.005940    0.00345   1.720   0.113
## beta.8   0.005160    0.00413   1.250   0.237
## beta.9   0.003780    0.00628   0.601   0.560
## beta.10  0.001780    0.00968   0.184   0.857

model_poly_rbo_rain <- polyDlm(x=as.vector(rbo_rain ), y=as.vector(rbo), q=10, k = 2)

## Estimates and t-tests for beta coefficients:
##          Estimate Std. Error t value P(>|t|)
## beta.0   0.015300    0.01110   1.380   0.196
## beta.1   0.009620    0.00737   1.310   0.218
## beta.2   0.004830    0.00576   0.839   0.419
## beta.3   0.000907    0.00580   0.156   0.879
## beta.4  -0.002160    0.00618  -0.349   0.734
## beta.5  -0.004360    0.00611  -0.712   0.491
## beta.6  -0.005690    0.00546  -1.040   0.320
## beta.7  -0.006170    0.00474  -1.300   0.220
## beta.8  -0.005780    0.00549  -1.050   0.315
## beta.9  -0.004520    0.00872  -0.519   0.614
## beta.10 -0.002410    0.01380  -0.174   0.865
```

```
model_poly_rbo_hum <- polyDlm(x=as.vector(rbo_hum), y=as.vector(rbo), q=10, k = 2)
```

```
## Estimates and t-tests for beta coefficients:
##          Estimate Std. Error  t value P(>|t|)
## beta.0   3.57e-03    0.00263  1.36000   0.201
## beta.1   2.22e-03    0.00200  1.11000   0.290
## beta.2   1.03e-03    0.00173  0.59900   0.562
## beta.3   1.63e-05    0.00168  0.00969   0.992
## beta.4  -8.36e-04    0.00166 -0.50400   0.625
## beta.5  -1.52e-03    0.00159 -0.96000   0.358
## beta.6  -2.04e-03    0.00148 -1.38000   0.195
## beta.7  -2.39e-03    0.00150 -1.59000   0.139
## beta.8  -2.58e-03    0.00190 -1.36000   0.202
## beta.9  -2.60e-03    0.00275 -0.94500   0.365
## beta.10 -2.46e-03    0.00397 -0.61800   0.549
```

```
sort.score(MASE(model_poly_rbo_temp$model,model_poly_rbo_rad$model,
           model_poly_rbo_rain$model, model_poly_rbo_hum$model), score = "mase")
```

```
##                             n      MASE
## model_poly_rbo_rain$model 21 0.6451804
## model_poly_rbo_hum$model  21 0.6523137
## model_poly_rbo_rad$model  21 0.6711964
## model_poly_rbo_temp$model 21 0.6714874
```

```
sort.score(AIC(model_poly_rbo_temp$model,model_poly_rbo_rad$model,
           model_poly_rbo_rain$model, model_poly_rbo_hum$model), score = "aic")
```

```
##                           df       AIC
## model_poly_rbo_hum$model   5 -88.49321
## model_poly_rbo_rain$model  5 -88.26525
## model_poly_rbo_rad$model   5 -88.15084
## model_poly_rbo_temp$model  5 -87.29374
```

```
sort.score(BIC(model_poly_rbo_temp$model,model_poly_rbo_rad$model,
           model_poly_rbo_rain$model, model_poly_rbo_hum$model), score = "bic")
```

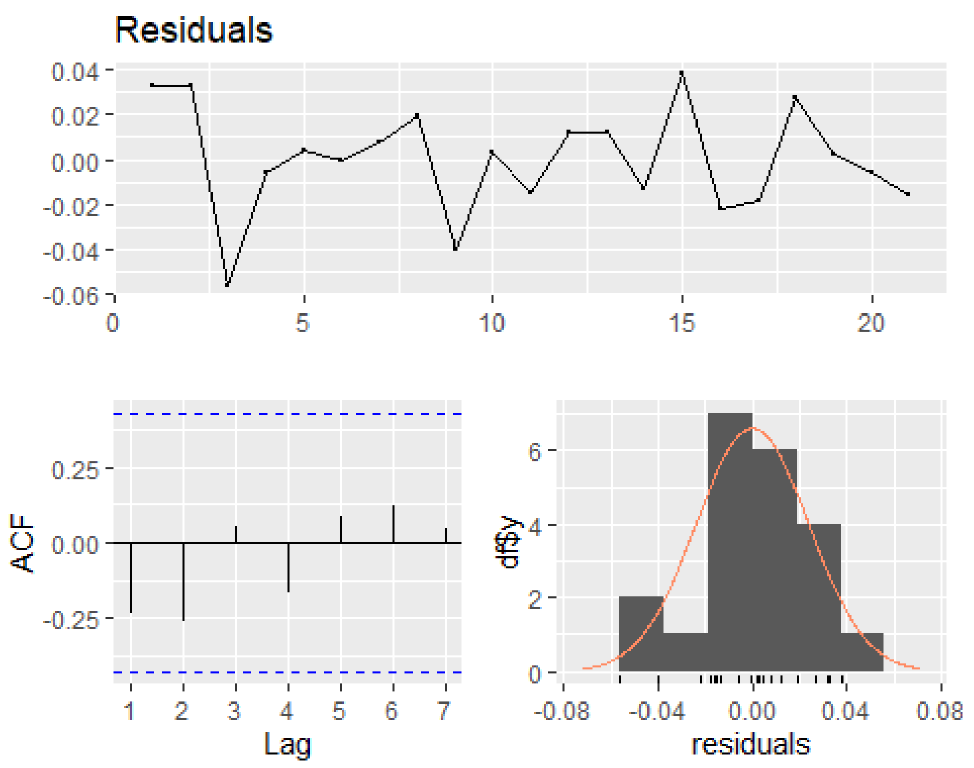
```
##                           df       BIC
## model_poly_rbo_hum$model   5 -83.27060
## model_poly_rbo_rain$model  5 -83.04264
## model_poly_rbo_rad$model   5 -82.92823
## model_poly_rbo_temp$model  5 -82.07113
```

- Considering the result for AIC BIC and MASE model_poly_rbo_rain is the best mode. Therefore, these two models will undergo further analysis for diagnostic checking.

```
summary(model_poly_rbo_rain)

##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.056127 -0.014819  0.002352  0.012277  0.038053
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.7166742  0.1029308   6.963 2.29e-06 ***
## z.t0         0.0152725  0.0110859   1.378    0.186
## z.t1        -0.0060829  0.0055819  -1.090    0.291
## z.t2         0.0004315  0.0005823   0.741    0.469
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02592 on 17 degrees of freedom
## Multiple R-squared:  0.1584, Adjusted R-squared:  0.009865
## F-statistic: 1.066 on 3 and 17 DF,  p-value: 0.3894
```

* The model “model_poly_rbo_rain” is not well fitted and almost all lag variables are statistically insignificant, except for “1st intercept.” The lower adjusted R-squared value 10% and Multiple R-squared value indicates that it can only explain 23.4% of the variation in the dependent variable. The Residual Standard Error (RSE) is 2.3%, and the range of residuals spans from a minimum value of -0.056127 to a maximum value of 0.038053. A lower RSE indicates a more precise fit..

```
checkresiduals(model_poly_rbo_rain$model)
```



```
##
##  Breusch-Godfrey test for serial correlation of order up to 7
##
## data:  Residuals
## LM test = 9.9973, df = 7, p-value = 0.1887
```

```
vif(model_poly_rbo_rain$model)
```

```
##      z.t0      z.t1      z.t2
##  8.346588 77.158308 47.720427
```

 * Examining the VIF (Variance Inflation Factor) values for the lag variables within model_poly_rbo_rain, they are all well above the threshold of 10, suggesting that multicollinearity is a significant issue in the model.

- Taking into account the results of the Breusch-Godfrey test, the p-value is above the 5% significance level, leading to the acceptance of the null hypothesis. This implies there is existence of no serial correlation within the residuals. Further examination of the residuals resulting from the model fit reveals that they exhibit a random distribution. Specifically, the autocorrelation function (ACF) confirming the presence of no serial correlation in the residuals. Additionally, the histograms indicate a departure from normal distribution.

Koyck Transformation DLM

```
model_Koyock_rbo_rad <- koyckDlm(x=as.vector(rbo_rad), y = as.vector(rbo))
model_Koyock_rbo_hum <- koyckDlm(x=as.vector(rbo_hum), y = as.vector(rbo))
```

```
model_Koyock_rbo_rain <- koyckDlm(x=as.vector(rbo_rain), y = as.vector(rbo))
model_Koyock_rbo_temp <- koyckDlm(x=as.vector(rbo_temp), y = as.vector(rbo))

sort.score(MASE(model_Koyock_rbo_rad,model_Koyock_rbo_hum,model_Koyock_rbo_rain,
model_Koyock_rbo_temp), score = "mase")

##                        n       MASE
## model_Koyock_rbo_hum  30  0.8400135
## model_Koyock_rbo_temp 30  0.9535116
## model_Koyock_rbo_rad  30  1.0314227
## model_Koyock_rbo_rain 30 19.1057647

AIC(model_Koyock_rbo_rad)

## [1] -93.8669

AIC(model_Koyock_rbo_hum)

## [1] -106.3136

AIC(model_Koyock_rbo_rain)

## [1] 76.22068

AIC(model_Koyock_rbo_temp)

## [1] -98.54907

BIC(model_Koyock_rbo_rad)

## [1] -88.26211

BIC(model_Koyock_rbo_hum)

## [1] -100.7088

BIC(model_Koyock_rbo_rain)

## [1] 81.82547

BIC(model_Koyock_rbo_temp)

## [1] -92.94428
```

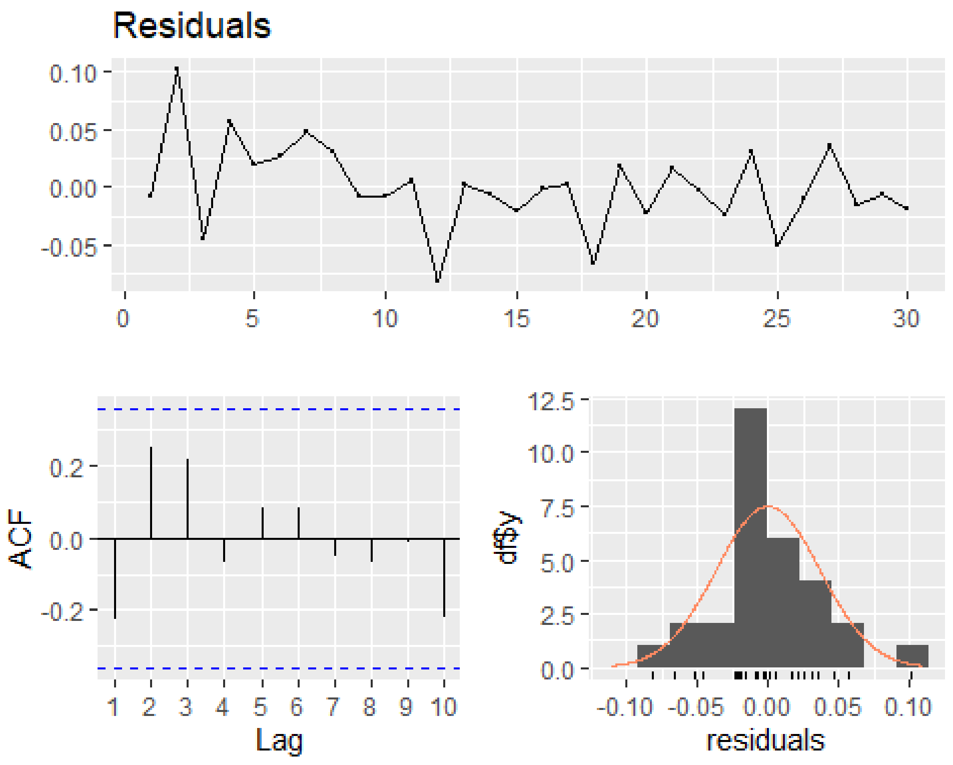
- Considering the result for AIC, BIC and MASE values model_Koyock_rbo_hum is the best model. Therefore, this model will undergo further analysis for diagnostic checking.

```
summary(model_Koyock_rbo_hum)

##
## 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:
## NULL
##
##                              alpha        beta       phi
## Geometric coefficients:  0.4769667 0.002164239 0.5484653
```

 * The model_Koyock_rbo_hum is not a good fit, as all the lag variables are statistically insignificant. Additionally, the multiple and Adjusted R-squared value is low, indicating that the model can explain 34.5% of the variation in the dependent variable. The range of residuals spans from a minimum value of -0.081436 to a maximum value of 0.101789. This signifies that, on average, the residuals deviate from the model's predictions by approximately 0.18 units. A lower RSE indicates a precise fit.

```
checkresiduals(model_Koyock_rbo_hum$model)
```



```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 6.2549, df = 6, p-value = 0.3952
##
## Model df: 0.   Total lags used: 6
```

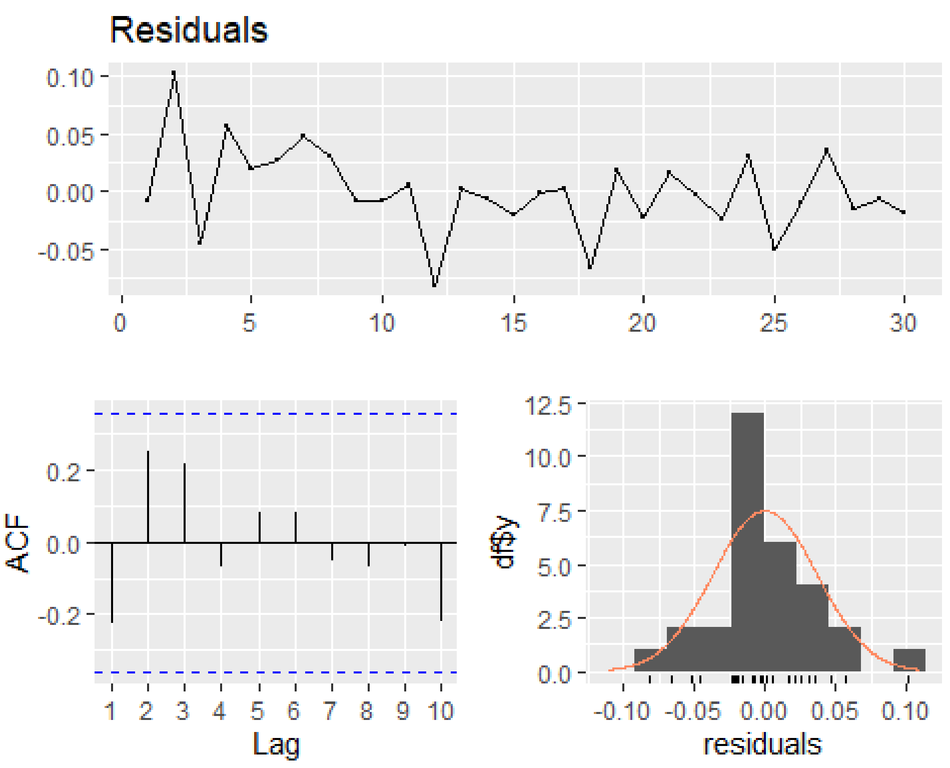
```
vif(model_Koyock_rbo_hum$model)>10
```

```
##   Y.1   X.t
## FALSE FALSE
```

 * Examining the VIF (Variance Inflation Factor) values for the lag variables within model_Koyock_rbo_hum, they are all well below the threshold of 10, suggesting that multicollinearity is not a significant issue in the model.

- Taking into account the results of the Breusch-Godfrey test, the p-value falls above the 5% significance level, leading to the acceptance of the null hypothesis. This implies the existence of no serial correlation within the residuals. Further examination of the residuals resulting from the model fit reveals that they exhibit a random distribution. Specifically, the autocorrelation function (ACF) confirming the presence of no serial correlation in the residuals, which aligns with the findings of the Breusch-Godfrey test having no significant lag. Additionally, the histograms indicate a departure from normal distribution.In summary, the overall conclusion is that the model might be the best fit when compared to all other models as far as R-squared, multicollinearity and serial correlation is concerned.

```
checkresiduals(model_Koyock_rbo_hum$model)
```



```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 6.2549, df = 6, p-value = 0.3952
##
## Model df: 0.   Total lags used: 6
```

```
vif(model_Koyock_rbo_hum$model)>10
```

```
##   Y.1   X.t
## FALSE FALSE
```

Autoregressive DLM

```
for (i in 1:5){
  for(j in 1:5){
    model_rbo_rad_ADLM <- ardlDlm(x= as.vector(rbo_rad),y=as.vector(rbo), p = i , q = j )
    cat("p =", i, "q =", j, "AIC =", AIC(model_rbo_rad_ADLM$model), "BIC =",
BIC(model_rbo_rad_ADLM$model), "MASE =", MASE(model_rbo_rad_ADLM)$MASE, "\n")
  }
}
```

```
## p = 1 q = 1 AIC = -107.5622 BIC = -100.5562 MASE = 0.8390648
## p = 1 q = 2 AIC = -106.904 BIC = -98.70018 MASE = 0.8249931
## p = 1 q = 3 AIC = -110.6338 BIC = -101.3084 MASE = 0.8202653
## p = 1 q = 4 AIC = -105.7923 BIC = -95.42561 MASE = 0.8192118
## p = 1 q = 5 AIC = -100.2033 BIC = -88.88038 MASE = 0.7627672
## p = 2 q = 1 AIC = -100.8085 BIC = -92.60473 MASE = 0.9205412
## p = 2 q = 2 AIC = -106.3664 BIC = -96.79529 MASE = 0.765347
## p = 2 q = 3 AIC = -109.7302 BIC = -99.07252 MASE = 0.7689908
## p = 2 q = 4 AIC = -104.7047 BIC = -93.04219 MASE = 0.7822872
## p = 2 q = 5 AIC = -98.98255 BIC = -86.40158 MASE = 0.7268185
## p = 3 q = 1 AIC = -104.0823 BIC = -94.75685 MASE = 0.8867923
## p = 3 q = 2 AIC = -109.5259 BIC = -98.8683 MASE = 0.756036
## p = 3 q = 3 AIC = -108.3733 BIC = -96.38349 MASE = 0.7495842
## p = 3 q = 4 AIC = -102.9405 BIC = -89.98212 MASE = 0.7620013
## p = 3 q = 5 AIC = -97.36077 BIC = -83.5217 MASE = 0.7009101
## p = 4 q = 1 AIC = -99.86384 BIC = -89.49714 MASE = 0.8800549
## p = 4 q = 2 AIC = -103.3431 BIC = -91.6806 MASE = 0.7906658
## p = 4 q = 3 AIC = -101.9438 BIC = -88.98546 MASE = 0.7820422
## p = 4 q = 4 AIC = -100.979 BIC = -86.72481 MASE = 0.7706287
## p = 4 q = 5 AIC = -95.51106 BIC = -80.4139 MASE = 0.7052516
## p = 5 q = 1 AIC = -96.72745 BIC = -85.40458 MASE = 0.7913984
## p = 5 q = 2 AIC = -96.91278 BIC = -84.33182 MASE = 0.7549241
## p = 5 q = 3 AIC = -95.91334 BIC = -82.07428 MASE = 0.7375755
## p = 5 q = 4 AIC = -94.72831 BIC = -79.63116 MASE = 0.7302406
## p = 5 q = 5 AIC = -93.6481 BIC = -77.29285 MASE = 0.7145698
```

```
for (i in 1:5){
  for(j in 1:5){
    model_rbo_rain_ADLM <- ardlDlm(x= as.vector(rbo_rain),y=as.vector(rbo), p = i , q = j )
    cat("p =", i, "q =", j, "AIC =", AIC(model_rbo_rain_ADLM$model), "BIC =", 
BIC(model_rbo_rain_ADLM$model), "MASE =", MASE(model_rbo_rain_ADLM)$MASE, "\n")
  }
}
```

```
## p = 1 q = 1 AIC = -105.2619 BIC = -98.25588 MASE = 0.828275
## p = 1 q = 2 AIC = -103.3681 BIC = -95.16429 MASE = 0.8543791
## p = 1 q = 3 AIC = -106.9248 BIC = -97.59935 MASE = 0.8322089
## p = 1 q = 4 AIC = -102.0678 BIC = -91.70114 MASE = 0.8714349
## p = 1 q = 5 AIC = -95.80256 BIC = -84.47969 MASE = 0.8152025
## p = 2 q = 1 AIC = -99.21505 BIC = -91.01127 MASE = 0.9189202
## p = 2 q = 2 AIC = -101.4552 BIC = -91.88416 MASE = 0.8562257
## p = 2 q = 3 AIC = -104.9284 BIC = -94.27076 MASE = 0.8328582
## p = 2 q = 4 AIC = -100.0892 BIC = -88.4267 MASE = 0.8706236
## p = 2 q = 5 AIC = -93.80342 BIC = -81.22246 MASE = 0.81573
## p = 3 q = 1 AIC = -102.0287 BIC = -92.70325 MASE = 0.8852654
## p = 3 q = 2 AIC = -106.4754 BIC = -95.8178 MASE = 0.8316226
## p = 3 q = 3 AIC = -105.1996 BIC = -93.2098 MASE = 0.8307901
## p = 3 q = 4 AIC = -99.66585 BIC = -86.70748 MASE = 0.860244
## p = 3 q = 5 AIC = -93.30294 BIC = -79.46387 MASE = 0.8095313
## p = 4 q = 1 AIC = -96.40802 BIC = -86.04133 MASE = 0.8956382
## p = 4 q = 2 AIC = -100.4881 BIC = -88.82555 MASE = 0.8337316
## p = 4 q = 3 AIC = -100.0049 BIC = -87.04657 MASE = 0.7754451
## p = 4 q = 4 AIC = -98.96532 BIC = -84.71111 MASE = 0.7942758
## p = 4 q = 5 AIC = -92.62017 BIC = -77.52301 MASE = 0.7390848
## p = 5 q = 1 AIC = -93.55318 BIC = -82.23031 MASE = 0.7936346
## p = 5 q = 2 AIC = -94.0473 BIC = -81.46633 MASE = 0.7842158
## p = 5 q = 3 AIC = -93.91526 BIC = -80.0762 MASE = 0.7237105
## p = 5 q = 4 AIC = -92.68035 BIC = -77.58319 MASE = 0.7490465
## p = 5 q = 5 AIC = -90.68282 BIC = -74.32757 MASE = 0.7469974
```

```
for (i in 1:5){
  for(j in 1:5){
    model_rbo_temp_ADLM <- ardlDlm(x= as.vector(rbo_temp),y=as.vector(rbo), p = i , q = j )
    cat("p =", i, "q =", j, "AIC =", AIC(model_rbo_temp_ADLM$model), "BIC =",
BIC(model_rbo_temp_ADLM$model), "MASE =", MASE(model_rbo_temp_ADLM)$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
```

```
for (i in 1:5){
  for(j in 1:5){
    model_rbo_hum_ADLM <- ardlDlm(x= as.vector(rbo_hum),y=as.vector(rbo), p = i , q = j )
    cat("p =", i, "q =", j, "AIC =", AIC(model_rbo_hum_ADLM$model), "BIC =",
BIC(model_rbo_hum_ADLM$model), "MASE =", MASE(model_rbo_hum_ADLM)$MASE, "\n")
  }
}
```

```
## p = 1 q = 1 AIC = -105.5243 BIC = -98.51828 MASE = 0.8268308
## p = 1 q = 2 AIC = -105.4814 BIC = -97.27767 MASE = 0.831358
## p = 1 q = 3 AIC = -108.3405 BIC = -99.01504 MASE = 0.8196903
## p = 1 q = 4 AIC = -103.6137 BIC = -93.24697 MASE = 0.8299345
## p = 1 q = 5 AIC = -97.27466 BIC = -85.95179 MASE = 0.7813433
## p = 2 q = 1 AIC = -100.142 BIC = -91.93822 MASE = 0.9289166
## p = 2 q = 2 AIC = -103.7809 BIC = -94.20985 MASE = 0.823535
## p = 2 q = 3 AIC = -106.5199 BIC = -95.86225 MASE = 0.8124903
## p = 2 q = 4 AIC = -101.6346 BIC = -89.97207 MASE = 0.826765
## p = 2 q = 5 AIC = -95.28529 BIC = -82.70433 MASE = 0.779284
## p = 3 q = 1 AIC = -103.6086 BIC = -94.28315 MASE = 0.8727659
## p = 3 q = 2 AIC = -109.1887 BIC = -98.53104 MASE = 0.7361386
## p = 3 q = 3 AIC = -107.2029 BIC = -95.21302 MASE = 0.7390577
## p = 3 q = 4 AIC = -101.3096 BIC = -88.35125 MASE = 0.7691159
## p = 3 q = 5 AIC = -95.58362 BIC = -81.74456 MASE = 0.734551
## p = 4 q = 1 AIC = -99.05124 BIC = -88.68455 MASE = 0.8693994
## p = 4 q = 2 AIC = -102.3928 BIC = -90.73025 MASE = 0.7616985
## p = 4 q = 3 AIC = -100.4041 BIC = -87.44573 MASE = 0.7626157
## p = 4 q = 4 AIC = -99.33532 BIC = -85.08111 MASE = 0.7712952
## p = 4 q = 5 AIC = -93.59832 BIC = -78.50116 MASE = 0.73115
## p = 5 q = 1 AIC = -97.08049 BIC = -85.75762 MASE = 0.792207
## p = 5 q = 2 AIC = -97.18983 BIC = -84.60887 MASE = 0.7060096
## p = 5 q = 3 AIC = -95.23858 BIC = -81.39952 MASE = 0.7051169
## p = 5 q = 4 AIC = -94.25778 BIC = -79.16063 MASE = 0.7262065
## p = 5 q = 5 AIC = -92.45536 BIC = -76.1001 MASE = 0.7370787
```

* Looking at the above output p= 5 and q =5 has the lowest MASE, AIC and BIC, therefore further exploration will be conducted using summary and diagnostic checking.

```
model_ard_rbo_rad <- ardlDlm(x=as.vector(rbo_rad), y=as.vector(rbo),p = 5, q = 5)
model_ard_rbo_temp <- ardlDlm(x=as.vector(rbo_temp), y=as.vector(rbo),p = 5, q = 5)
model_ard_rbo_rain <- ardlDlm(x=as.vector(rbo_rain), y=as.vector(rbo),p = 5, q = 5)
model_ard_rbo_hum <- ardlDlm(x=as.vector(rbo_hum), y=as.vector(rbo),p = 5, q = 5)

sort.score(AIC(model_ard_rbo_rad$model,model_ard_rbo_temp$model,
           model_ard_rbo_rain$model, model_ard_rbo_hum$model), score = "aic")

##                          df       AIC
## model_ard_rbo_temp$model 13 -93.93583
## model_ard_rbo_rad$model  13 -93.64810
## model_ard_rbo_hum$model  13 -92.45536
## model_ard_rbo_rain$model 13 -90.68282

sort.score(BIC(model_ard_rbo_rad$model,model_ard_rbo_temp$model,
           model_ard_rbo_rain$model, model_ard_rbo_hum$model), score = "bic")

##                          df       BIC
## model_ard_rbo_temp$model 13 -77.58057
## model_ard_rbo_rad$model  13 -77.29285
## model_ard_rbo_hum$model  13 -76.10010
## model_ard_rbo_rain$model 13 -74.32757

MASE(model_ard_rbo_temp,model_ard_rbo_rad,model_ard_rbo_rain, model_ard_rbo_hum)

##                     n      MASE
## model_ard_rbo_temp 26 0.7335979
## model_ard_rbo_rad  26 0.7145698
## model_ard_rbo_rain 26 0.7469974
## model_ard_rbo_hum  26 0.7370787
```

* Looking at the above output model_ard_rbo_rad has lower MASE meaning the model_ard_rbo_rad model will be the best to proceed for further exploration.But based AIC and BIC on model_ard_rbo_temp is also better model

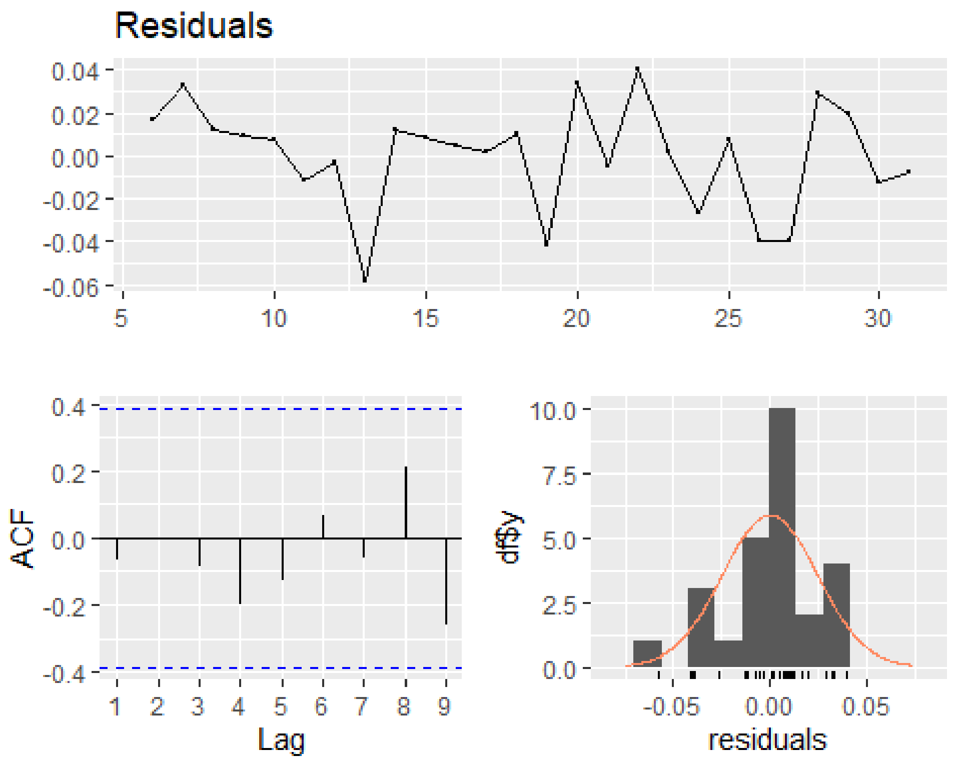
```
summary(model_ard_rbo_rad)
```

```
##
## 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
```

* The model_ard_rbo_rad is slightly a better fit as the multiple R-squared value is high compared to the previous models, indicating that the model can explain 64.7% of the variation in the dependent variable. The range of residuals spans from a minimum value of -0.058181 to a maximum value of 0.039893. This signifies that, on average, the residuals deviate from the model’s predictions by approximately 0.09 units. A much lower RSE indicates a precise fit.

```
checkresiduals(model_ard_rbo_rad)
```

```
## Time Series:
## Start = 6
## End = 31
## Frequency = 1
##            6            7            8            9           10           11
##  0.017139855  0.032528043  0.011978201  0.009529145  0.007149992 -0.011484462
##           12           13           14           15           16           17
## -0.002936589 -0.058181129  0.011619359  0.008094096  0.004856336  0.001416247
##           18           19           20           21           22           23
##  0.010294444 -0.041510270  0.033257139 -0.005333645  0.039892523  0.001881551
##           24           25           26           27           28           29
## -0.026160046  0.007017437 -0.039978587 -0.039287089  0.029186017  0.019457487
##           30           31
## -0.012637191 -0.007788862
```



```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 2.236, df = 5, p-value = 0.8156
##
## Model df: 0.   Total lags used: 5
```

```
vif(model_ard_rbo_rad$model)>10
```

```
##       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)
##     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE
## L(y.t, 3) L(y.t, 4) L(y.t, 5)
##     FALSE     FALSE     FALSE
```

* Examining the VIF values for the lag variables within model_ard_rbo_rad, they are all well below the threshold of 10, suggesting that multicollinearity is not a significant issue in the model.

- Taking into account the results of the Breusch-Godfrey test, the p-value exceeds the 5% significance level, leading to the acceptance of the null hypothesis. This implies the existence of no serial correlation within the residuals. Further examination of the residuals resulting from the finite dynamic linear model (DLM) fit reveals that they do not exhibit a random distribution. Specifically, the autocorrelation function (ACF) confirming the presence of no serial correlation in the residuals, which aligns with the findings of the Breusch-Godfrey test having all lags below the 95% confidence interval. Additionally, the histograms indicate a departure from normal distribution.In summary, the overall conclusion is that the model might be the better fit for the data and the presence of no serial correlation makes the model ready for plotting.

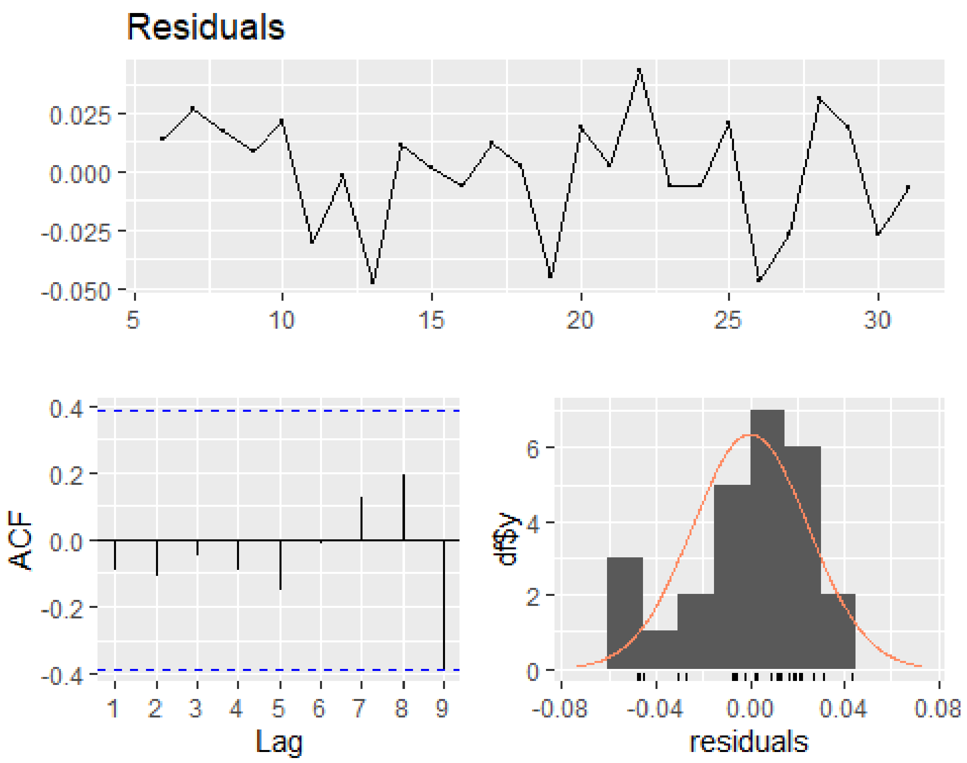
```
summary(model_ard_rbo_temp)

##
## 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
```

* The model_ard_rbo_temp is slightly a better fit as the multiple R-squared value is high compared to the previous models, indicating that the model can explain 65% of the variation in

the dependent variable. The range of residuals spans from a minimum value of -0.047576 to a maximum value of 0.042903 . This signifies that, on average, the residuals deviate from the model's predictions by approximately 1 units. A much lower RSE indicates a precise fit.

```
checkresiduals(model_ard_rbo_temp)
```

```
## Time Series:
## Start = 6
## End = 31
## Frequency = 1
##            6            7            8            9           10           11
##  0.013624442  0.026926565  0.016904945  0.008702514  0.021623472 -0.030201793
##           12           13           14           15           16           17
## -0.002033309 -0.047575774  0.011499212  0.001973955 -0.005908240  0.012278603
##           18           19           20           21           22           23
##  0.002761799 -0.045280430  0.018852291  0.002889156  0.042903061 -0.005913975
##           24           25           26           27           28           29
## -0.006206665  0.020934084 -0.047308567 -0.027378127  0.031260563  0.018695179
##           30           31
## -0.026934165 -0.007088798
```



```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 1.7239, df = 5, p-value = 0.8859
```

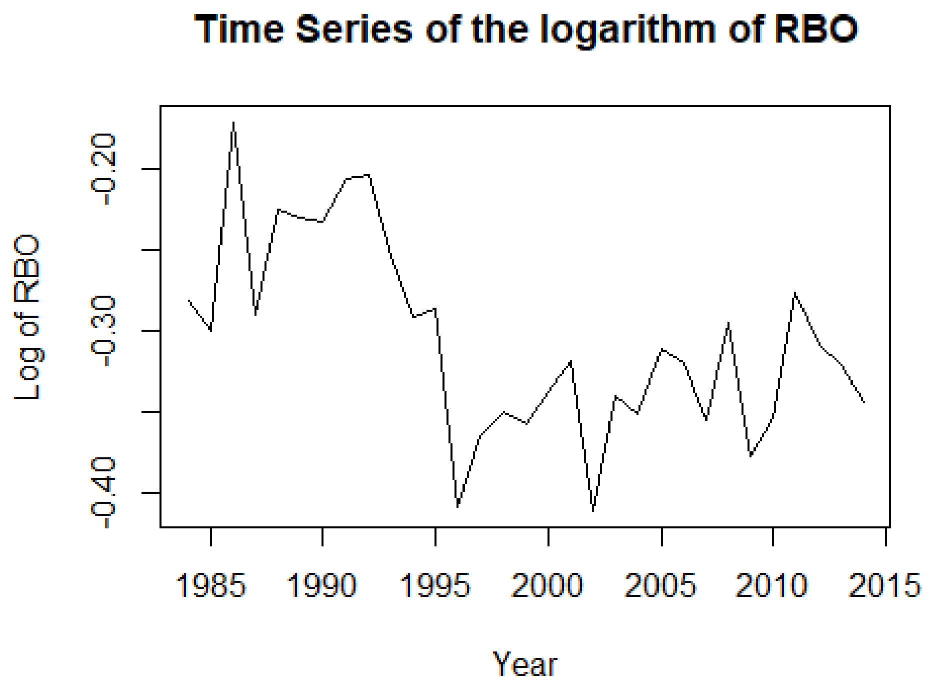
```
##
## Model df: 0.   Total lags used: 5
```

```
vif(model_ard_rbo_temp$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
```

Dynamic Linear Modelling

```
log_rbo <- log(rbo)

par(mfrow=c(1,1))
plot(log_rbo,ylab='Log of RBO',xlab='Year',
    main = "Time Series of the logarithm of RBO")
```



```
X.t <- log(rbo)
t <- 96
P.t <- 1*(seq(log(rbo))==T)
P.t.1 <- lag(P.t, +1)

dynlm_rbo_1<- dynlm(X.t~ L(X.t, k=1) + trend(X.t))
dynlm_rbo_2<- dynlm(X.t~ L(X.t, k=1))
dynlm_rbo_3<- dynlm(X.t~ L(X.t, k=1) + P.t)
dynlm_rbo_4<- dynlm(X.t~ L(X.t, k=1) + P.t.1)
```

```
dynlm_rbo_5 <- dynlm(X.t~ L(X.t, k=1) + L(X.t, k=2))
dynlm_rbo_6 <- dynlm(X.t~ L(X.t, k=1) + L(X.t, k=2)+P.t)
dynlm_rbo_7 <- dynlm(X.t~ L(X.t, k=1) + L(X.t, k=2)+P.t + P.t.1)
dynlm_rbo_8 <- dynlm(X.t~ L(X.t, k=1) + L(X.t, k=2)+L(X.t, k=3))

dynlm_rbo_temp_1<- dynlm(X.t~temp + L(X.t, k=1) + trend(X.t))
dynlm_rbo_temp_2<- dynlm(X.t~temp + L(X.t, k=1))
dynlm_rbo_temp_3 <- dynlm(X.t~temp + L(X.t, k=1) + P.t)
dynlm_rbo_temp_4 <- dynlm(X.t~temp +L(X.t, k=1) + P.t.1)
dynlm_rbo_temp_5 <- dynlm(X.t~temp + L(X.t, k=1) + L(X.t, k=2))
dynlm_rbo_temp_6 <- dynlm(X.t~temp + L(X.t, k=1) + L(X.t, k=2)+P.t)
dynlm_rbo_temp_7<- dynlm(X.t~temp-1 + L(X.t, k=1))


dynlm_rbo_rad_1<- dynlm(X.t~rad + L(X.t, k=1) + trend(X.t))
dynlm_rbo_rad_2<- dynlm(X.t~rad + L(X.t, k=1))
dynlm_rbo_rad_3 <- dynlm(X.t~rad + L(X.t, k=1) + P.t)
dynlm_rbo_rad_4 <- dynlm(X.t~rad +L(X.t, k=1) + P.t.1)
dynlm_rbo_rad_5 <- dynlm(X.t~rad + L(X.t, k=1) + L(X.t, k=2))
dynlm_rbo_rad_6 <- dynlm(X.t~rad + L(X.t, k=1) + L(X.t, k=2)+P.t)
dynlm_rbo_rad_7 <- dynlm(X.t~rad + L(X.t, k=1) + L(X.t, k=2)+P.t + P.t.1)
dynlm_rbo_rad_8 <- dynlm(X.t~rad + L(X.t, k=1) + L(X.t, k=2)+L(X.t, k=3))
dynlm_rbo_rad_9<- dynlm(X.t~rad-1 + L(X.t, k=1))

dynlm_rbo_hum_1<- dynlm(X.t~hum + L(X.t, k=1) + trend(X.t))
dynlm_rbo_hum_2<- dynlm(X.t~hum + L(X.t, k=1))
dynlm_rbo_hum_3 <- dynlm(X.t~hum + L(X.t, k=1) + P.t)
dynlm_rbo_hum_4 <- dynlm(X.t~hum +L(X.t, k=1) + P.t.1)
dynlm_rbo_hum_5 <- dynlm(X.t~hum + L(X.t, k=1) + L(X.t, k=2))
dynlm_rbo_hum_6 <- dynlm(X.t~hum + L(X.t, k=1) + L(X.t, k=2)+P.t)
dynlm_rbo_hum_7 <- dynlm(X.t~hum + L(X.t, k=1) + L(X.t, k=2)+P.t + P.t.1)
dynlm_rbo_hum_8 <- dynlm(X.t~hum + L(X.t, k=1) + L(X.t, k=2)+L(X.t, k=3))
dynlm_rbo_hum_9<- dynlm(X.t~hum-1 + L(X.t, k=1))


dynlm_rbo_rain_1<- dynlm(X.t~rain + L(X.t, k=1) + trend(X.t))
dynlm_rbo_rain_2<- dynlm(X.t~rain + L(X.t, k=1))
dynlm_rbo_rain_3 <- dynlm(X.t~rain + L(X.t, k=1) + P.t)
dynlm_rbo_rain_4 <- dynlm(X.t~rain +L(X.t, k=1) + P.t.1)
dynlm_rbo_rain_5 <- dynlm(X.t~rain + L(X.t, k=1) + L(X.t, k=2))
dynlm_rbo_rain_6 <- dynlm(X.t~rain + L(X.t, k=1) + L(X.t, k=2)+P.t)
dynlm_rbo_rain_7 <- dynlm(X.t~rain + L(X.t, k=1) + L(X.t, k=2)+P.t + P.t.1)
dynlm_rbo_rain_8 <- dynlm(X.t~rain + L(X.t, k=1) + L(X.t, k=2)+L(X.t, k=3))
dynlm_rbo_rain_9<- dynlm(X.t~rain -1+ L(X.t, k=1))

MASE(lm(dynlm_rbo_1),lm(dynlm_rbo_2),lm(dynlm_rbo_3),lm(dynlm_rbo_4),lm(dynlm_rbo_5),lm(d
ynlm_rbo_6),lm(dynlm_rbo_7),lm(dynlm_rbo_8),lm(dynlm_rbo_temp_1),
lm(dynlm_rbo_temp_2),lm(dynlm_rbo_temp_3),lm(dynlm_rbo_temp_4),lm(dynlm_rbo_temp_5),lm(dy
nlm_rbo_temp_6),lm(dynlm_rbo_temp_7),lm(dynlm_rbo_rad_1),lm(dynlm_rbo_rad_2),lm(dynlm_rbo_r
ad_3),lm(dynlm_rbo_rad_4),lm(dynlm_rbo_rad_5),lm(dynlm_rbo_rad_6),lm(dynlm_rbo_rad_7),lm(dynl
m_rbo_rad_8),lm(dynlm_rbo_rad_9),lm(dynlm_rbo_hum_1),lm(dynlm_rbo_hum_2),lm(dynlm_rbo_hu
m_3),lm(dynlm_rbo_hum_4),lm(dynlm_rbo_hum_5),lm(dynlm_rbo_hum_6),lm(dynlm_rbo_hum_7),lm
(dynlm_rbo_hum_8),lm(dynlm_rbo_hum_9),lm(dynlm_rbo_rain_2),lm(dynlm_rbo_rain_3),lm(dynlm_r
bo_rain_4),lm(dynlm_rbo_rain_5),lm(dynlm_rbo_rain_6),lm(dynlm_rbo_rain_7),lm(dynlm_rbo_rain_8),
lm(dynlm_rbo_rain_9))
```

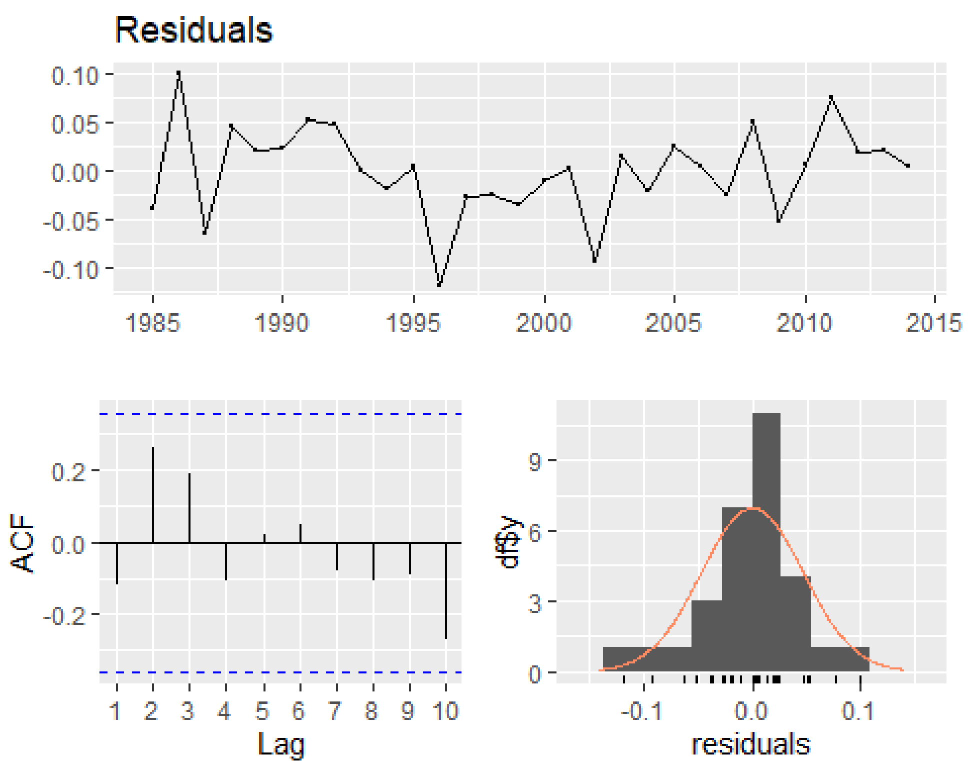
```
##                        n      MASE
## lm(dynlm_rbo_1)      30 0.8176689
## lm(dynlm_rbo_2)      30 0.8459895
## lm(dynlm_rbo_3)      30 0.8459895
## lm(dynlm_rbo_4)      30 0.8412781
## lm(dynlm_rbo_5)      29 0.8908321
## lm(dynlm_rbo_6)      29 0.8908321
## lm(dynlm_rbo_7)      29 0.8908321
## lm(dynlm_rbo_8)      28 0.8670128
## lm(dynlm_rbo_temp_1) 30 0.8182221
## lm(dynlm_rbo_temp_2) 30 0.8158516
## lm(dynlm_rbo_temp_3) 30 0.8158516
## lm(dynlm_rbo_temp_4) 30 0.8087164
## lm(dynlm_rbo_temp_5) 29 0.8608099
## lm(dynlm_rbo_temp_6) 29 0.8608099
## lm(dynlm_rbo_temp_7) 30 0.8267773
## lm(dynlm_rbo_rad_1)  30 0.8139310
## lm(dynlm_rbo_rad_2)  30 0.8304974
## lm(dynlm_rbo_rad_3)  30 0.8304974
## lm(dynlm_rbo_rad_4)  30 0.8285124
## lm(dynlm_rbo_rad_5)  29 0.8877896
## lm(dynlm_rbo_rad_6)  29 0.8877896
## lm(dynlm_rbo_rad_7)  29 0.8877896
## lm(dynlm_rbo_rad_8)  28 0.8666714
## lm(dynlm_rbo_rad_9)  30 0.8265941
## lm(dynlm_rbo_hum_1)  30 0.8175301
## lm(dynlm_rbo_hum_2)  30 0.8428049
## lm(dynlm_rbo_hum_3)  30 0.8428049
## lm(dynlm_rbo_hum_4)  30 0.8390357
## lm(dynlm_rbo_hum_5)  29 0.8932058
## lm(dynlm_rbo_hum_6)  29 0.8932058
## lm(dynlm_rbo_hum_7)  29 0.8932058
## lm(dynlm_rbo_hum_8)  28 0.8618015
## lm(dynlm_rbo_hum_9)  30 0.8436956
## lm(dynlm_rbo_rain_2) 30 0.8460971
## lm(dynlm_rbo_rain_3) 30 0.8460971
## lm(dynlm_rbo_rain_4) 30 0.8398416
## lm(dynlm_rbo_rain_5) 29 0.8413297
## lm(dynlm_rbo_rain_6) 29 0.8413297
## lm(dynlm_rbo_rain_7) 29 0.8413297
## lm(dynlm_rbo_rain_8) 28 0.8194140
## lm(dynlm_rbo_rain_9) 30 0.9343187
```

* Considering all the models and fitting the model with intercept and non intercept,dynlm_rbo_1 has the lowest MASE. Lets look into summary statistics.

```
summary(dynlm_rbo_1)

##
## Time series regression with "ts" data:
## Start = 1985, End = 2014
##
## Call:
## dynlm(formula = X.t ~ L(X.t, k = 1) + trend(X.t))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.117502 -0.025013  0.004407  0.022792  0.099885
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.154099   0.047123  -3.270  0.00293 **
## L(X.t, k = 1)  0.366055   0.178084   2.056  0.04962 *
## trend(X.t)    -0.002485   0.001221  -2.035  0.05173 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04831 on 27 degrees of freedom
## Multiple R-squared:  0.4081, Adjusted R-squared:  0.3643
## F-statistic: 9.309 on 2 and 27 DF,  p-value: 0.0008414
```

- The model labeled “dynlm_rbo_1” does not appear to be a better fit. This is evident from its lower Adjusted R-squared value of 36.43% when compared to previous model.

```
checkresiduals(dynlm_rbo_1)
```



```
##
##  Breusch-Godfrey test for serial correlation of order up to 6
##
## data:  Residuals
## LM test = 5.5694, df = 6, p-value = 0.4731
```

```
vif(dynlm_rbo_1)
```

```
## L(X.t, k = 1)    trend(X.t)
##      1.434959      1.434959
```

 * Examining the VIF (Variance Inflation Factor) values for the lag variables within dynlm_rbo_temp_40, they are all well below the threshold of 10, suggesting that multicollinearity is not a significant issue in the model.

- However, the results of the Breusch-Godfrey test suggest the presence of no serial correlation in the residuals of this model.The histogram does not appear to be normally distributed.

```
MASE(model_ard_rbo_rad)
```

```
##                        MASE
## model_ard_rbo_rad 0.7145698
```

```
MASE(model_ard_rbo_temp)
```

```
##                         MASE
## model_ard_rbo_temp 0.7335979
```

```
MASE(model_rbo_hum_finite_noin$model)

##                                   MASE
## model_rbo_hum_finite_noin$model 0.9918515

MASE(model_poly_rbo_rain$model)

##                              MASE
## model_poly_rbo_rain$model 0.6451804

MASE(model_Koyock_rbo_hum)

##                         MASE
## model_Koyock_rbo_hum 0.8400135

MASE(model_poly_rbo_hum$model)

##                             MASE
## model_poly_rbo_hum$model 0.6523137

MASE(model_rbo_rain_finite$model)

##                               MASE
## model_rbo_rain_finite$model 0.9417954
```

- Considering all the models. model_ard_rbo_rad has the lowest MASE and no presence of multicollinearity makes the model compatible for forecasting with higher Multiple R-squared of 0.6466. The next lowest MASE is model_ard_rbo_temp. The model_ard_rbo_temp has higher adjusted R square of 65% and lower RSE that makes the model more suitable for modelling with no issue of multicollinearity.

Even model_poly_rbo_rain has the lowest mase but the presence of multicollinearity and lower Adjust R squared makes it incompatible for forecasting

Forecasting With Auto Regressive Model

Task 3 Part_B

```
Y.t <- log(rbo)
t <- 13
K.t <- 1*(seq(Y.t)==T)
K.t.1 <- lag(K.t, +1)

dynlm_rbo_1<- dynlm(Y.t~ L(Y.t, k=1) + trend(Y.t))
dynlm_rbo_2 <- dynlm(Y.t~ L(Y.t, k=1) + K.t.1 + K.t + trend(Y.t))
dynlm_rbo_3<- dynlm(Y.t~ L(Y.t, k=1))
dynlm_rbo_4 <- dynlm(Y.t~ L(Y.t, k=1) + K.t+ trend(Y.t))
dynlm_rbo_5 <- dynlm(Y.t~ L(Y.t, k=1) + K.t.1)
dynlm_rbo_6 <- dynlm(Y.t~ L(Y.t, k=1) + L(Y.t, k=2))
dynlm_rbo_7 <- dynlm(Y.t~ L(Y.t, k=1) + L(Y.t, k=2)+Y.t)
dynlm_rbo_8 <- dynlm(Y.t~ L(Y.t, k=1) + L(Y.t, k=2)+K.t + K.t.1)
```

```
dynlm_rbo_9 <- dynlm(Y.t~ L(Y.t, k=1) + L(Y.t, k=2)+L(Y.t, k=3))
dynlm_rbo_10 <- dynlm(Y.t~ L(Y.t, k=1) + L(Y.t, k=2)+L(Y.t, k=3)+L(Y.t, k=4))
dynlm_rbo_11 <- dynlm(Y.t~ L(Y.t, k=1) + L(Y.t, k=2)+L(Y.t, k=3)+L(Y.t, k=4)+L(Y.t, k=5))

dynlm_rbo_12 = dynlm(Y.t~rbo_rad + L(Y.t , k = 1 ) + L(Y.t , k = 2 )+ K.t+ + K.t.1 + trend(Y.t))
dynlm_rbo_13 = dynlm(Y.t~rbo_rain + L(Y.t , k = 1 ) + L(Y.t , k = 2 )+ K.t+ + K.t.1 + trend(Y.t))
dynlm_rbo_14 = dynlm(Y.t~rbo_temp + L(Y.t , k = 1 ) + L(Y.t , k = 2 )+ K.t+ + K.t.1 + trend(Y.t))
dynlm_rbo_15 = dynlm(Y.t~rbo_hum + L(Y.t , k = 1 ) + L(Y.t , k = 2 )+ K.t+ + K.t.1 + trend(Y.t))

dynlm_rbo_16 = dynlm(Y.t~rbo_hum + L(Y.t , k = 1 ) + K.t+ + K.t.1 + trend(Y.t))
dynlm_rbo_17 = dynlm(Y.t~rbo_rad + L(Y.t , k = 1 ) + K.t+ + K.t.1 + trend(Y.t))
dynlm_rbo_17 = dynlm(Y.t~rbo_rain + L(Y.t , k = 1 ) + K.t+ + K.t.1 + trend(Y.t))
dynlm_rbo_17 = dynlm(Y.t~rbo_temp + L(Y.t , k = 1 ) + K.t+ + K.t.1 + trend(Y.t))

MASE(lm(dynlm_rbo_1),lm(dynlm_rbo_2),lm(dynlm_rbo_3),lm(dynlm_rbo_4),lm(dynlm_rbo_5),lm(d
ynlm_rbo_6),lm(dynlm_rbo_7),lm(dynlm_rbo_8),lm(dynlm_rbo_9),lm(dynlm_rbo_10),lm(dynlm_rbo_1
1),lm(dynlm_rbo_12),lm(dynlm_rbo_13),lm(dynlm_rbo_14),lm(dynlm_rbo_15),lm(dynlm_rbo_16),lm(
dynlm_rbo_17))

##                     n      MASE
## lm(dynlm_rbo_1)  30 0.8176689
## lm(dynlm_rbo_2)  30 0.7907736
## lm(dynlm_rbo_3)  30 0.8459895
## lm(dynlm_rbo_4)  30 0.8176689
## lm(dynlm_rbo_5)  30 0.8412781
## lm(dynlm_rbo_6)  29 0.8908321
## lm(dynlm_rbo_7)  29 0.8908321
## lm(dynlm_rbo_8)  29 0.8908321
## lm(dynlm_rbo_9)  28 0.8670128
## lm(dynlm_rbo_10) 27 0.8838536
## lm(dynlm_rbo_11) 26 0.8125114
## lm(dynlm_rbo_12) 29 0.8170220
## lm(dynlm_rbo_13) 29 0.7903930
## lm(dynlm_rbo_14) 29 0.8144944
## lm(dynlm_rbo_15) 29 0.7964913
## lm(dynlm_rbo_16) 30 0.7797798
## lm(dynlm_rbo_17) 30 0.7919299

summary(dynlm_rbo_16)

##
## Time series regression with "ts" data:
## Start = 1985, End = 2014
##
## Call:
## dynlm(formula = Y.t ~ rbo_hum + L(Y.t, k = 1) + K.t + +K.t.1 +
##     trend(Y.t))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.126654 -0.017633 -0.001396  0.028915  0.087269
```

```
##
## Coefficients: (1 not defined because of singularities)
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.640328   0.379675  -1.687   0.1041
## rbo_hum        0.008736   0.006747   1.295   0.2072
## L(Y.t, k = 1)  0.302180   0.181460   1.665   0.1083
## K.t                  NA         NA      NA       NA
## K.t.1         -0.049666   0.051866  -0.958   0.3474
## trend(Y.t)    -0.002647   0.001290  -2.053   0.0507 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04794 on 25 degrees of freedom
## Multiple R-squared:  0.4602, Adjusted R-squared:  0.3739
## F-statistic: 5.329 on 4 and 25 DF,  p-value: 0.003034
```

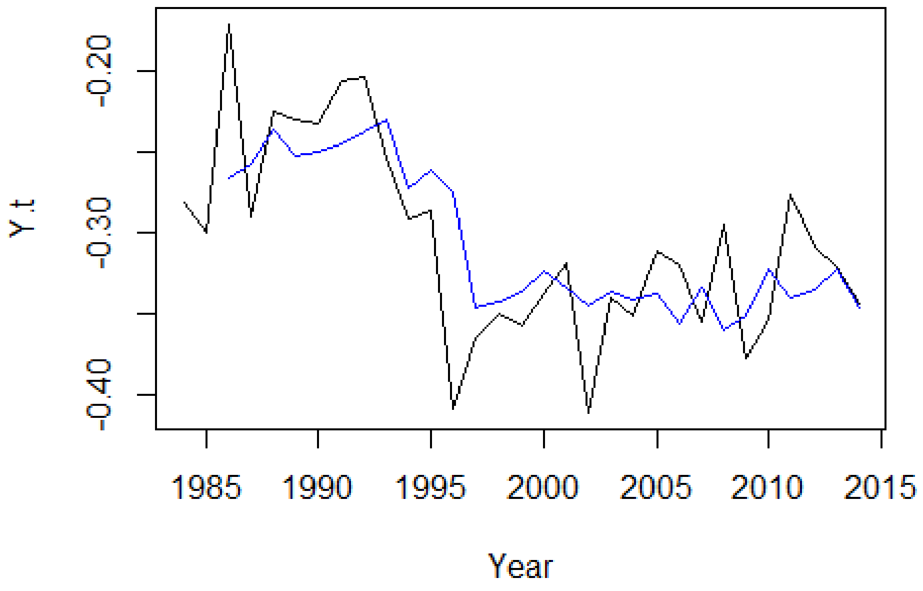
```r
summary(dynlm_rbo_13)
```

```
##
## Time series regression with "ts" data:
## Start = 1986, End = 2014
##
## Call:
## dynlm(formula = Y.t ~ rbo_rain + L(Y.t, k = 1) + L(Y.t, k = 2) +
##     K.t + +K.t.1 + trend(Y.t))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.13393 -0.02226 -0.00303  0.02587  0.09442
##
## Coefficients: (2 not defined because of singularities)
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.226020   0.093820  -2.409    0.024 *
## rbo_rain       0.036815   0.024675   1.492    0.149
## L(Y.t, k = 1)  0.148857   0.196002   0.759    0.455
## L(Y.t, k = 2)  0.276226   0.185541   1.489    0.150
## K.t                  NA         NA      NA       NA
## K.t.1                NA         NA      NA       NA
## trend(Y.t)    -0.002255   0.001334  -1.690    0.104
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04638 on 24 degrees of freedom
## Multiple R-squared:  0.5149, Adjusted R-squared:  0.434
## F-statistic: 6.368 on 4 and 24 DF,  p-value: 0.00122
```

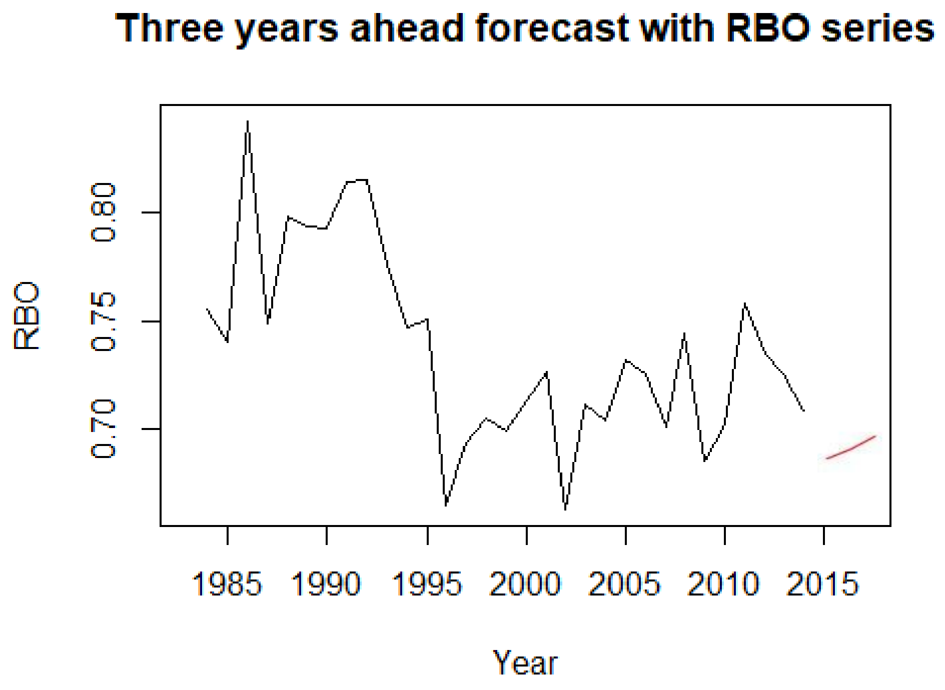
* Comparing the dynlm_rbo_16 and dynlm_rbo_13 with lowest mase dynlm_rbo_13 displays the higher adjusted r-square 0.434 and Multiple R-squared 0.5149, with slightly lower RSE dynlm_rbo_13 seems to be a better fit for forecasting.

- Taking into account the results of the Breusch-Godfrey test, the p-value exceeds the 5% significance level, leading to the acceptance of the null hypothesis. This implies the existence of no serial correlation within the residuals. Specifically, the autocorrelation function (ACF) confirming the presence of no serial correlation in the residuals, which aligns with the findings of the Breusch-Godfrey test having all lags below the 95% confidence interval. Additionally, the histograms indicate a departure from normal distribution.In summary, the overall conclusion is that the model might be the better fit for the data and the presence of no serial correlation makes the model ready for plotting.

```
par(mfrow =c(1,1))
plot(Y.t, xlab = "Year", main= "")
lines(dynlm_rbo_13$fitted.values,col= "blue")
```



```
q = 3
n = nrow(dynlm_rbo_13$model)
dynlm_rbo_13.frc = array(NA , (n + q))
dynlm_rbo_13.frc[1:n] = Y.t[3:length(Y.t)]
trend = array(NA,q)
trend.start = dynlm_rbo_13$model[n,"trend(Y.t)"]
trend = seq(trend.start , trend.start + q/12, 1/12)

for(i in 1:q){
dynlm_rbo_13.new =c(1,1,dynlm_rbo_13.frc[n-2+i],dynlm_rbo_13.frc[n-2+i],1,1,trend[i])
dynlm_rbo_13.frc[n+i] = as.vector(dynlm_rbo_13$coefficients) %*% dynlm_rbo_13.new
}

plot(rbo,xlim=c(1983,2017),
ylab='RBO',xlab='Year',
main = "Three years ahead forecast with RBO series")
lines(ts(dynlm_rbo_13.frc[(n+1):(n+q)],start=c(2015)),col="red")
```



References:

Hudson, I. Time Series Regression Models I - Distributed lag models - By using dLagM R-package [Lab solutions]. MATH1307: Forecasting. MATH1307 Forecasting
RMIT University, School of Science, Mathematical Sciences.

Hudson, I. Time Series Regression Models II - Dynamic Models. MATH1307: Forecasting. MATH1307 Forecasting. RMIT University, School of Science, Mathematical Sciences.

Hudson, I. Exponential Smoothing Methods. MATH1307: Forecasting. MATH1307 Forecasting. RMIT University, School of Science, Mathematical Sciences.

Hudson, I. Linear Innovations State Space Models. MATH1307: Forecasting. MATH1307 Forecasting. RMIT University, School of Science, Mathematical Sciences.

Hudson, I. Time Series Regression Models II - Dynamic Models. MATH1307: Forecasting. MATH1307 Forecasting. RMIT University, School of Science, Mathematical Sciences.

G.G. Judge, W.E. Griffiths. Undergraduate Econometrics. Wiley, 2000.

STHDA- A quick start guide to analyze, format and visualize a correlation matrix using R software. Retrieved September 20 2023 from http://www.sthda.com/english/wiki/correlation-matrix-a-quick-start-guide-to-analyze-format-and-visualize-a-correlation-matrix-using-r-software

Demirhan, H. (2023, October 2). Package 'dLagM': Time Series Regression Models with Distributed Lag Model. Retrieved from https://cran.rproject.org/web/packages/dLagM/dLagM.pdf

Achim Zeilei. (2023, October 13). Package 'dynlm': Dynamic linear models and time series regression. Retrieved from https://cran.r-project.org/web/packages/dynlm/dynlm.pdf

datacamp. aggregate.zoo: Compute Summary Statistics of zoo Objects. Retrieved October 1 2023 from https://www.rdocumentation.org/packages/zoo/versions/1.8-12/topics/aggregate.zoo