MATH1307 Forecasting Final project

Wong Yi Wei S3966890

2023-10-22

1.0.0 Task 1

1.1.0 Introduction

The aim of this investigation is to give the best 4 weeks ahead forecasts in terms of R squared, AIC, BIC, and MASE values for the mortality series. We will be using the data set `mort` which contains the weekly mortality, temperature, pollutants particle size, and two chemical emissions data between 2010-2020 in Paris, France. To perform these investigations, we will first prep the data for analysis and do further testing through various analysis methods.

1.2.0 Data Description and Preprocessing

The dataset `mort.csv` is used for analysis in task 1. Hence, we will import the data set using the `read_csv` function.

```
#Import and check data
data1<-read_csv("mort .csv")
```

```
## New names:
## Rows: 508 Columns: 6
## ── Column specification
## ──────────────────────────────────────────────────────── Delimiter: "," dbl
## (6): ...1, mortality, temp, chem1, chem2, particle size
## ℹ Use `spec()` to retrieve the full column specification for this data. ℹ
```

```
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## • `` -> `...1`
```

```
class(data1)
```

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

```
str(data1)
```

```
## spc_tbl_ [508 × 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ...1         : num [1:508] 1 2 3 4 5 6 7 8 9 10 ...
##  $ mortality    : num [1:508] 11.9 10.75 9.33 9.54 8.27 ...
##  $ temp         : num [1:508] 72.4 67.2 62.9 72.5 74.2 ...
##  $ chem1        : num [1:508] 3.37 2.59 3.29 3.04 3.39 2.57 2.35 3.38 1.5 2.56 ...
##  $ chem2        : num [1:508] 45.8 43.9 32.2 40.4 48.5 ...
##  $ particle size: num [1:508] 72.7 49.6 55.7 55.2 66 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ...1 = col_double(),
##   ..   mortality = col_double(),
##   ..   temp = col_double(),
##   ..   chem1 = col_double(),
##   ..   chem2 = col_double(),
##   ..   `particle size` = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
head(data1)
```

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

```
#Subset first column
data1<-data1[,-1]
```

Here we can see that the class, structure, and head of data is what we want. However, the first column can be subset as it is not needed. Next, using the `ts` function we will convert the data into a time series data, with each variable in it's own time series vector.

```
#Convert to time series
mort.ts<-ts(data1$mortality,start=c(2010,1),frequency=52)
temp.ts<-ts(data1$temp,start=c(2010,1),frequency=52)
chem1.ts<-ts(data1$chem1,start=c(2010,1),frequency=52)
chem2.ts<-ts(data1$chem2,start=c(2010,1),frequency=52)
psize.ts<-ts(data1$`particle size`,start=c(2010,1),frequency=52)
data1.ts<-ts(data1,start=c(2010,1),frequency=52)
```

1.3.0 Data Visualisation

Here we will plot each time series objects as a graph to gain a further understanding of the data through visual inspection. We will use the `plot` function to plot the graphs.

1.3.1 Mortality

```
plot(mort.ts,main="Time Series Plot for Mortality Series",
     ylab="Mortality",xlab="Year",type="o")
```

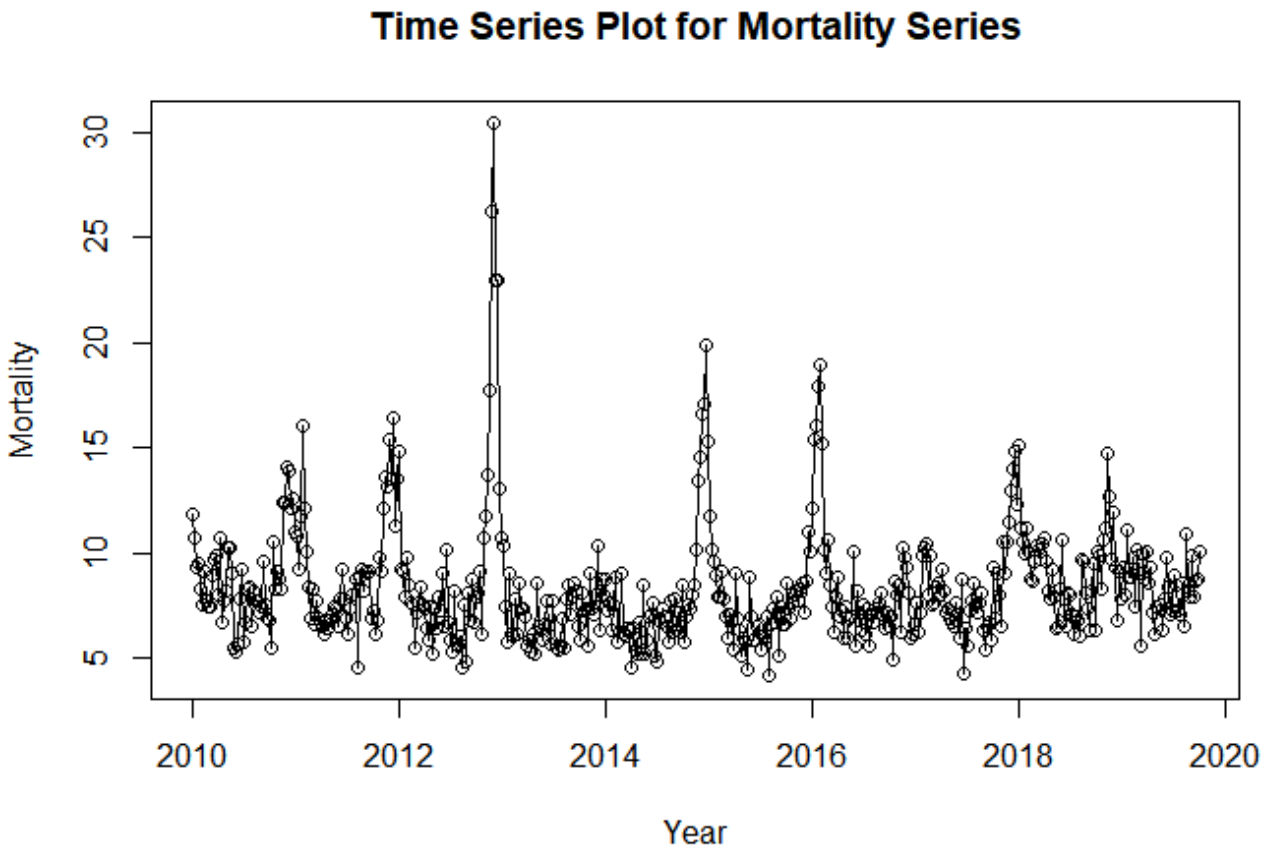

Figure 1

By `Figure 1`, which shows the time series plot of mortality series, we can make the following observations:

- The series has no apparent trends
- There are presence of seasonality, but the seasonal pattern is not constant.
- Since there are seasonality, changing variance and moving average behavior is not obvious.
- There is an possible intervention point around 2013.

```
acf(mort.ts, lag.max = 48, main="Sample ACF Plot for Mortality Series")
```

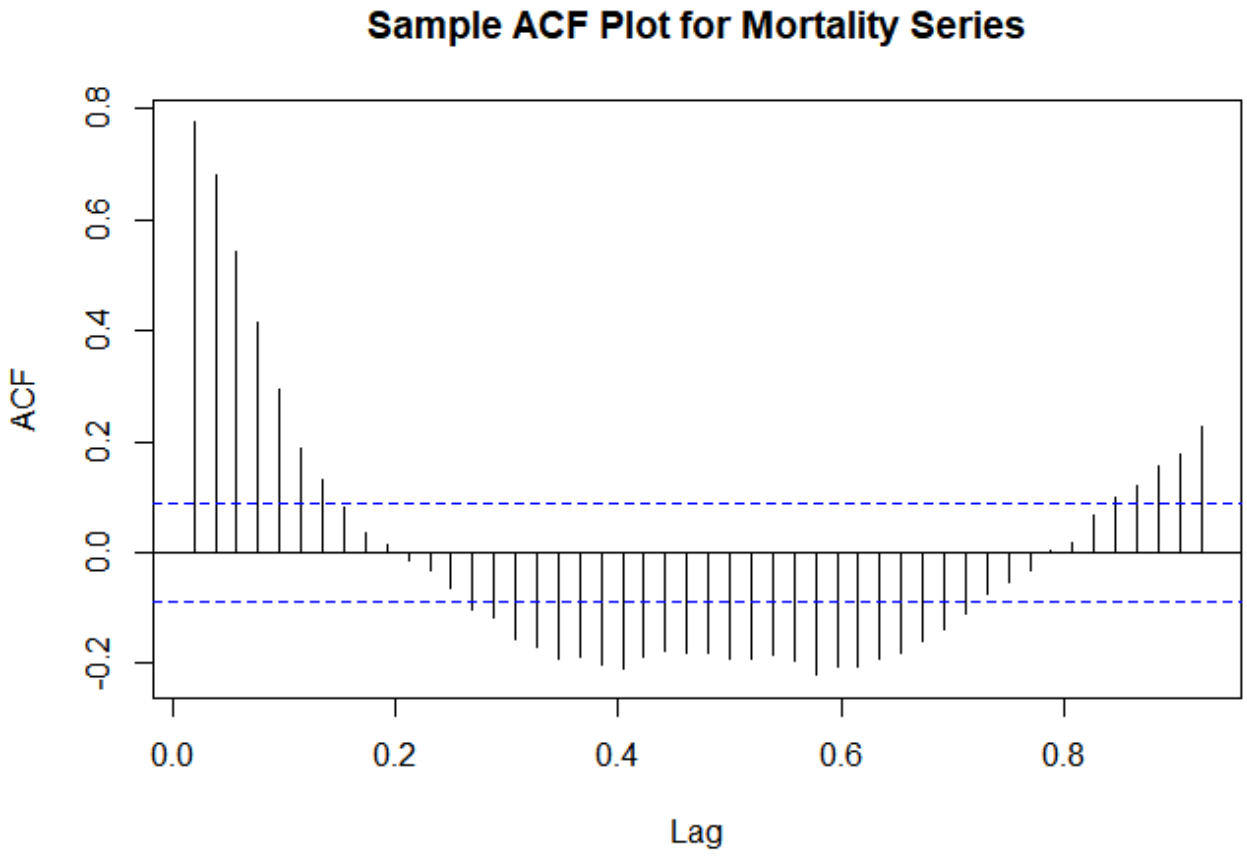

Figure 2

`Figure 2` shows the sample ACF plot for mortality series. Here we can observe that there is no apparent trend in the series, but presence of seasonal patterns.

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

```
## Warning in adf.test(mort.ts, k = ar(mort.ts)$order): 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 the above output, we used the ADF test for the mortality series. In this approach, we can conclude that the mortality series is stationary at 5% level of significance as p value is less than 0.05.

1.3.2 Temperature

```
plot(temp.ts,main="Time Series Plot for Temperature Series",
     ylab="Temperature (°F)",xlab="Year",type="o")
```

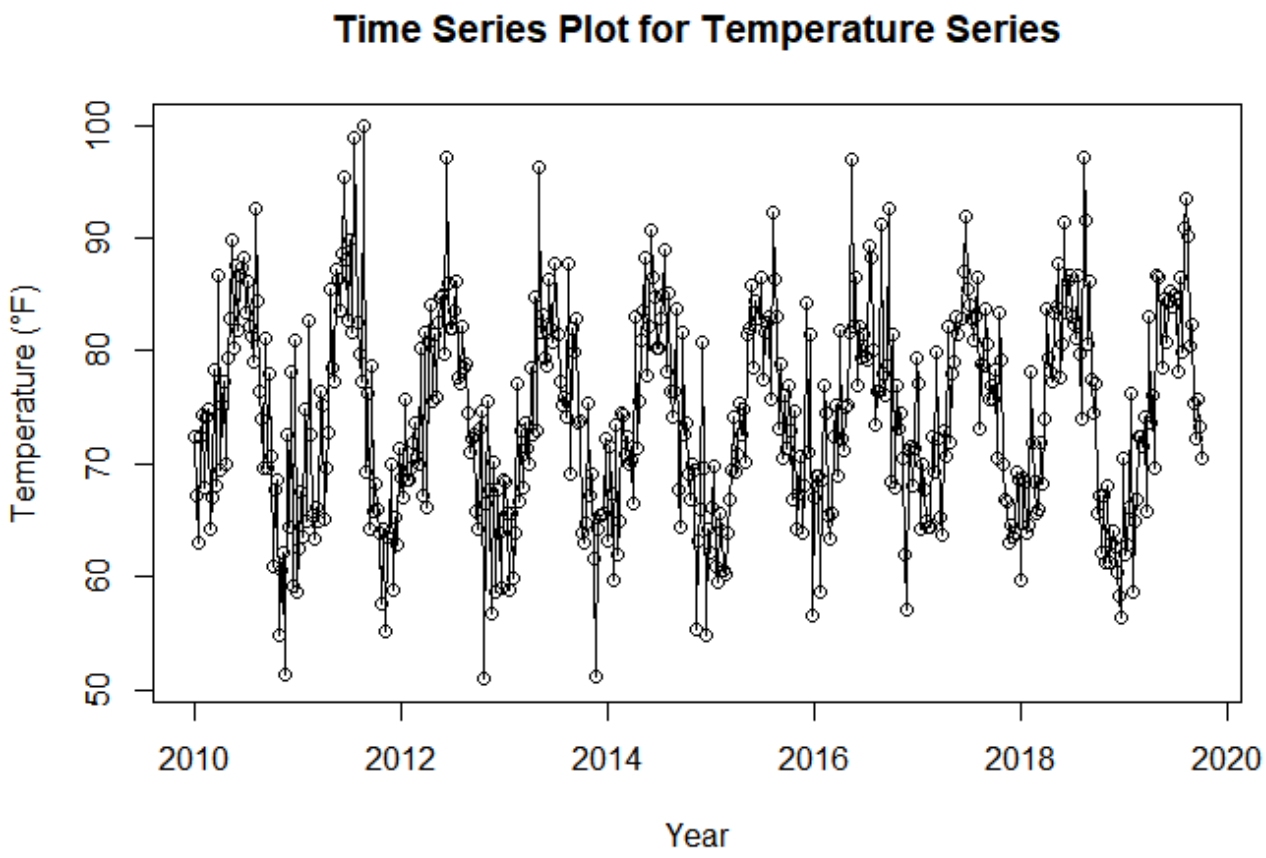

Figure 3

By `Figure 3`, which shows the time series plot of temperature series, we can make the following observations:

- The series has no apparent trends and intervention points
- There are presence of seasonality.
- Since there are seasonality, changing variance and moving average behavior is not obvious.

```
acf(temp.ts, lag.max = 48, main="Sample ACF Plot for Temperature Series")
```

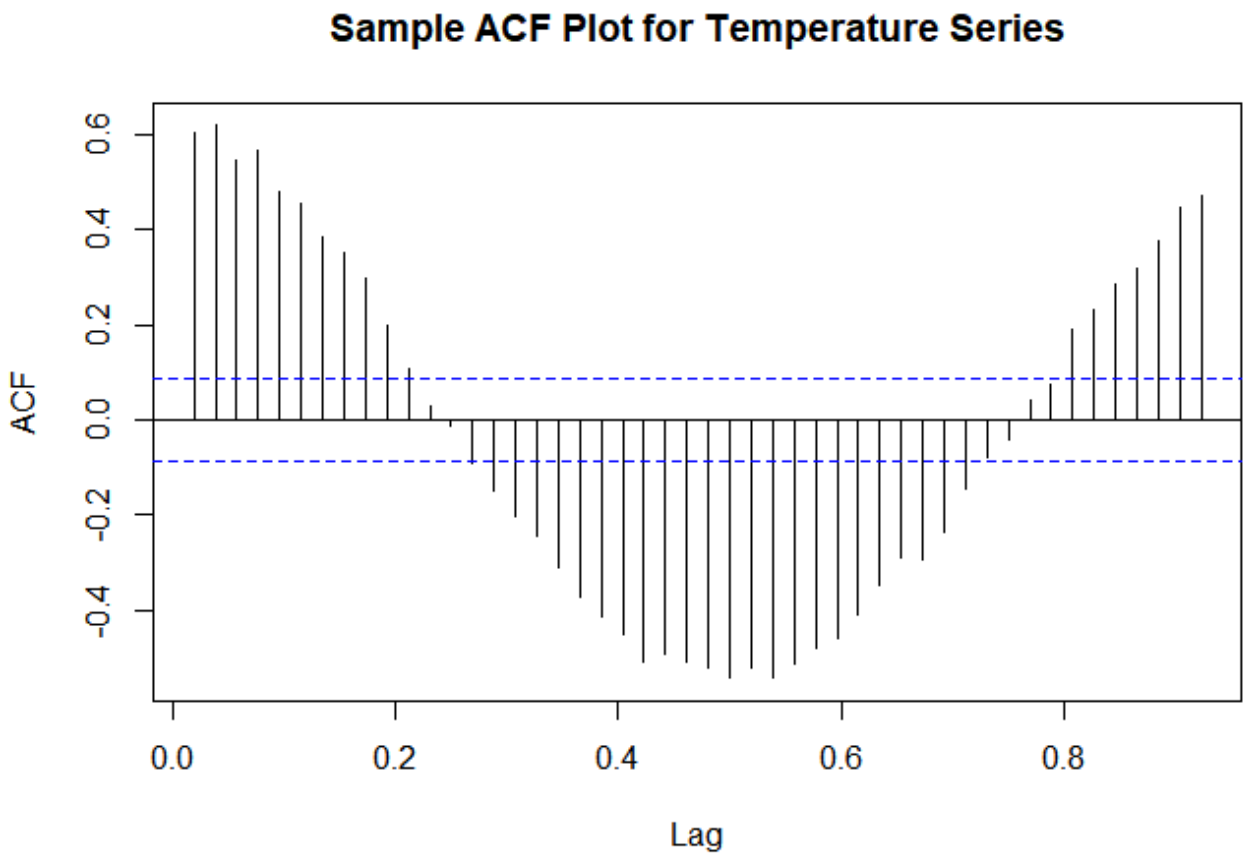

Figure 4

`Figure 4` shows the sample ACF plot for temperature series. Here we can observe that there is no apparent trend in the series, but presence of seasonal patterns.

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

```
## Warning in adf.test(temp.ts, k = ar(temp.ts)$order): 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 the above output, we used the ADF test for the temperature series. In this approach, we can conclude that the mortality series is stationary at 5% level of significance as p value is less than 0.05.

1.3.3 Chemical Emission 1

```
plot(chem1.ts,main="Time Series Plot for Chemical Emission 1 Series",
     ylab="Chemical Emission 1",xlab="Year",type="o")
```

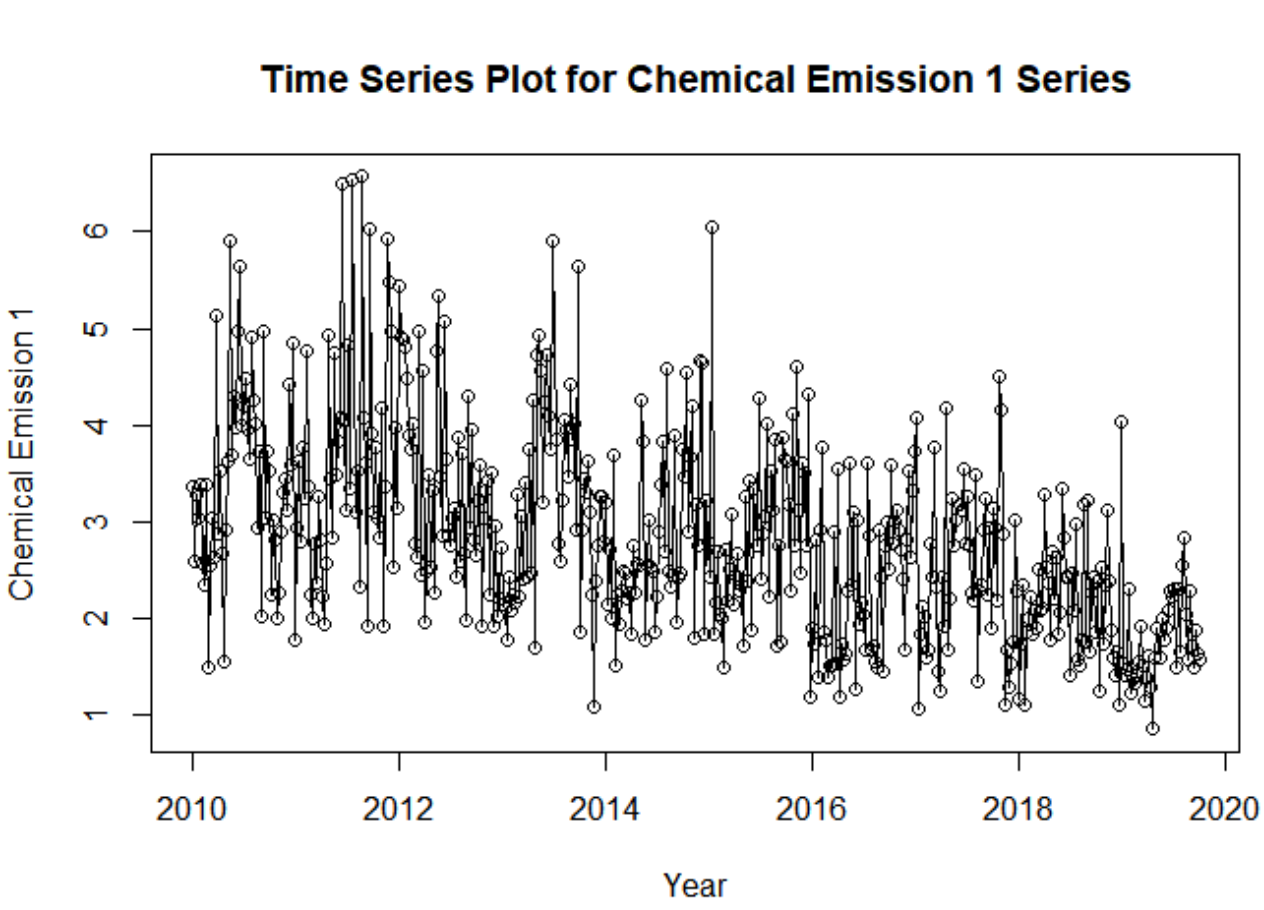

Figure 5

By `Figure 5`, which shows the time series plot of chemical emissions 1 series, we can make the following observations:

- The series has a downward trend but it is not apparent, however there is no intervention points observed
- There are presence of seasonality.
- Since there are seasonality, changing variance and moving average behavior is not obvious.

```
acf(chem1.ts, lag.max = 48, main="Sample ACF Plot for Chemical Emission 1 Series")
```

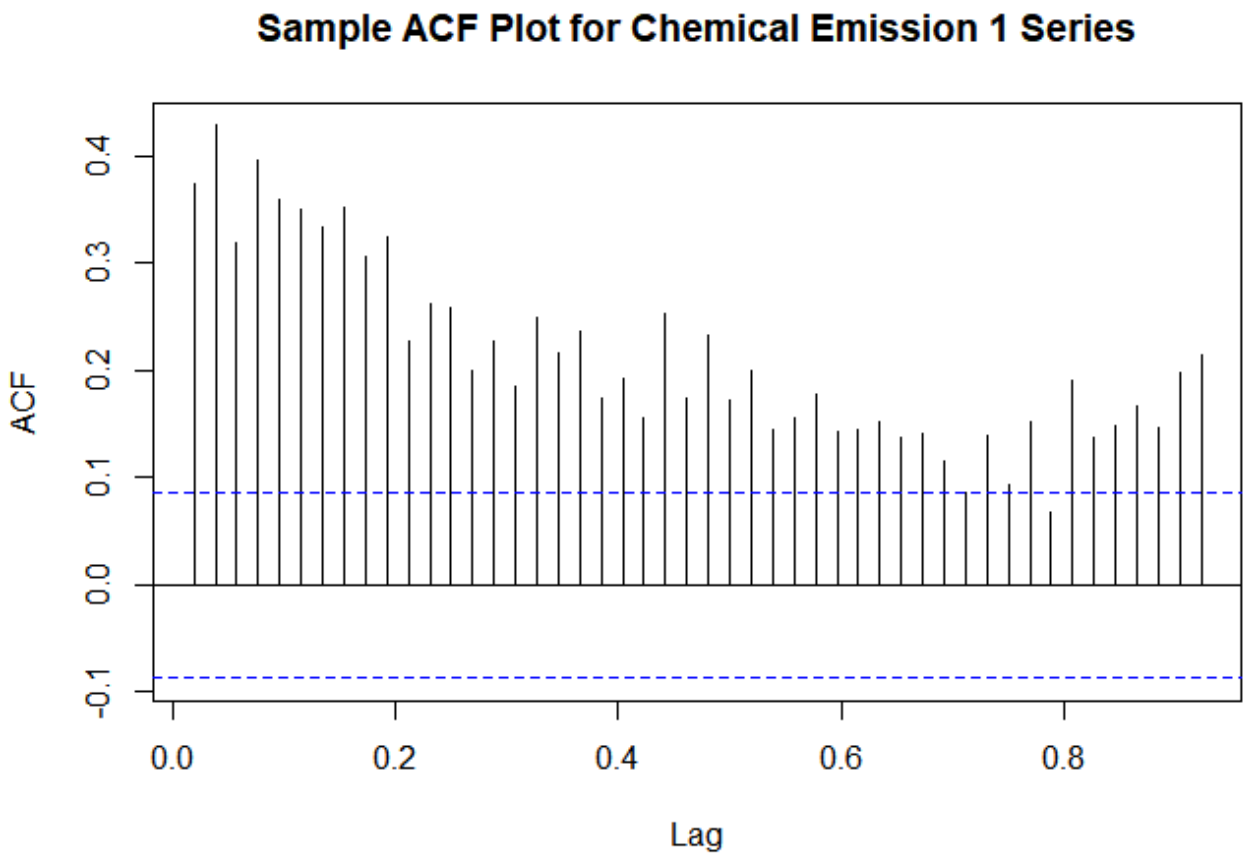

Figure 6

`Figure 6` shows the sample ACF plot for chemical emission 1 series. Here we can observe that there is no apparent trend in the series, but presence of seasonal patterns.

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

```
## Warning in adf.test(chem1.ts, k = ar(chem1.ts)$order): p-value smaller than
## printed p-value
```

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

In the above output, we used the ADF test for the chemical emission 1 series. In this approach, we can conclude that the mortality series is stationary at 5% level of significance as p value is less than 0.05.

1.3.4 Chemical Emission 2

```
plot(chem2.ts,main="Time Series Plot for Chemical Emission 2 Series",
     ylab="Chemical Emission 2",xlab="Year",type="o")
```

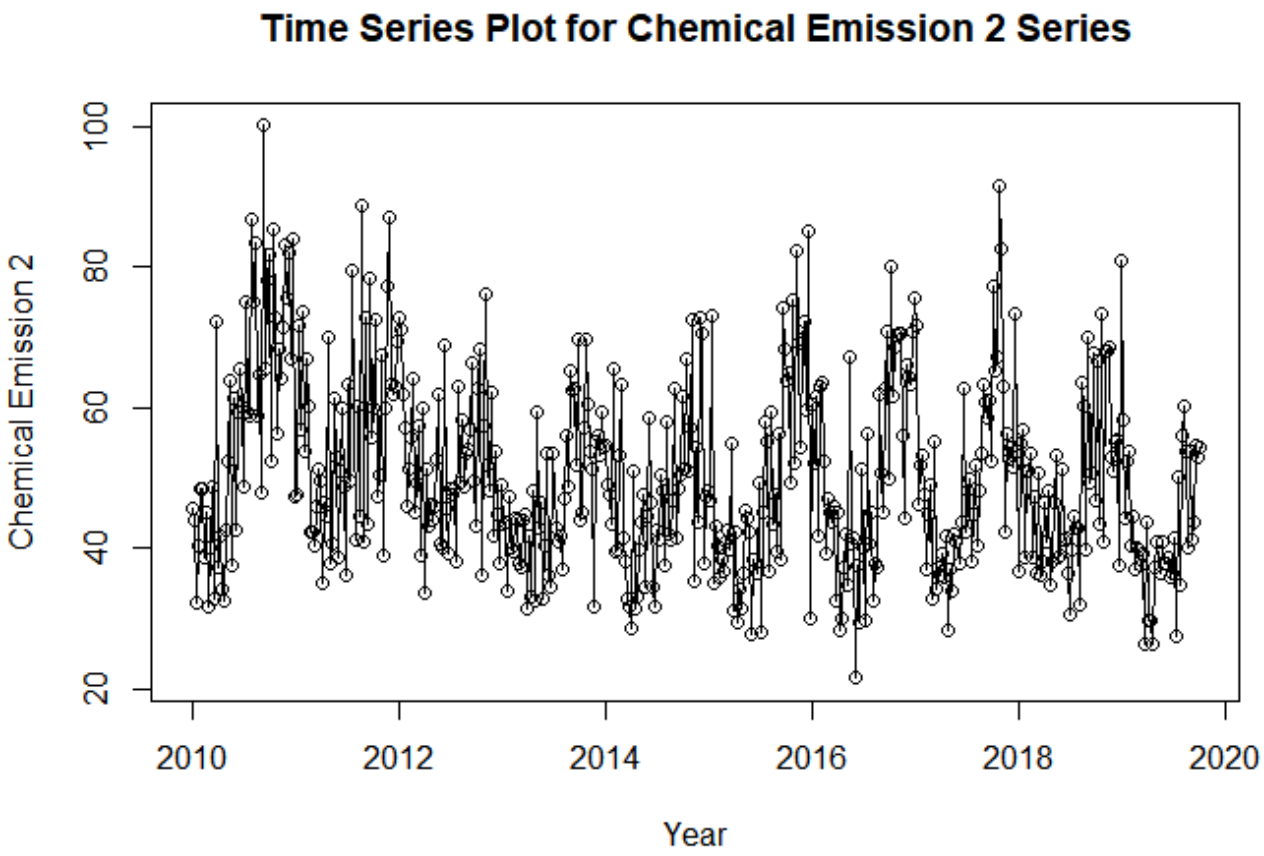

Figure 7

By `Figure 7`, which shows the time series plot of chemical emissions 2 series, we can make the following observations:

- The series has no apparent trend and no intervention points observed
- There are presence of seasonality.
- Since there are seasonality, changing variance and moving average behavior is not obvious.

```
acf(chem2.ts, lag.max = 48, main="Sample ACF Plot for Chemical Emission 2 Series")
```

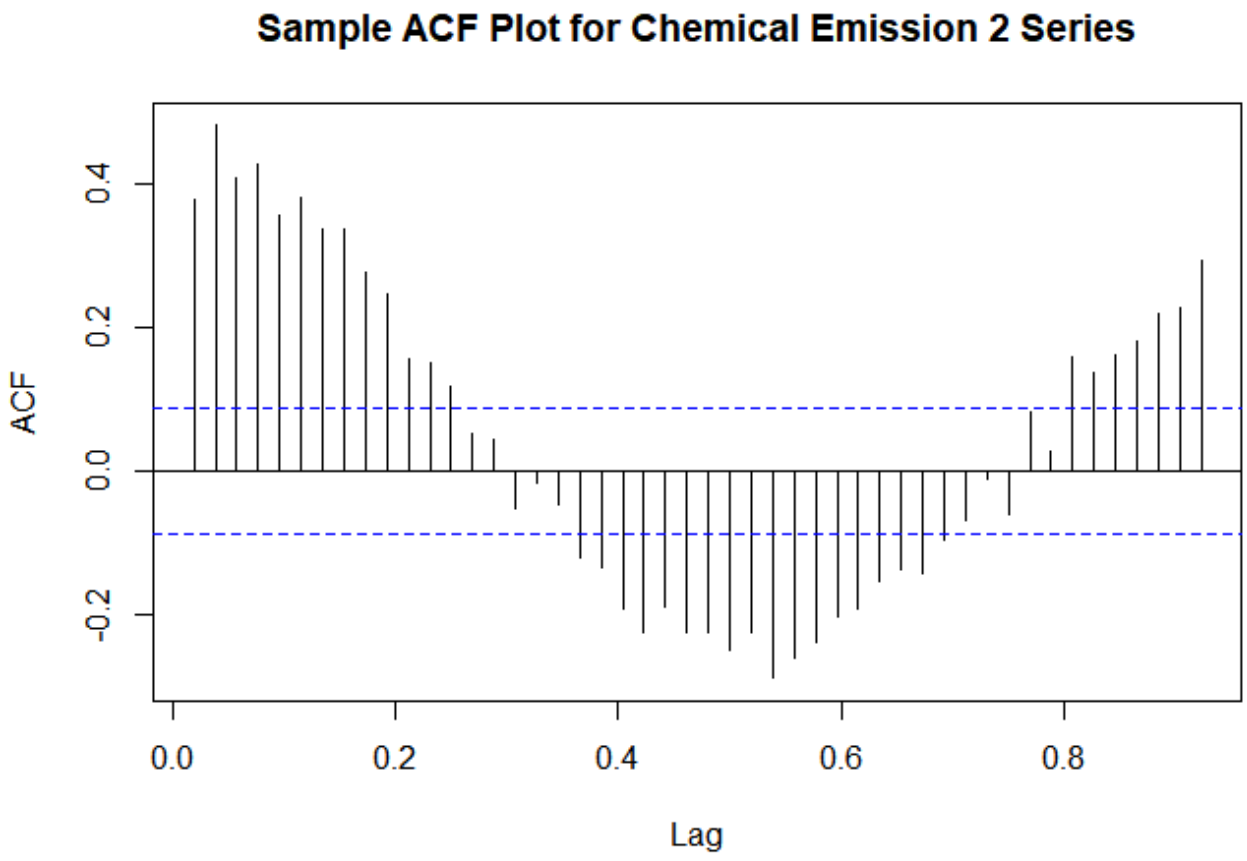

Figure 8

`Figure 8` shows the sample ACF plot for chemical emission 2 series. Here we can observe that there is no apparent trend in the series, but presence of seasonal patterns.

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

```
## Warning in adf.test(chem2.ts, k = ar(chem2.ts)$order): p-value smaller than
## printed p-value
```

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

In the above output, we used the ADF test for the chemical emission 2 series. In this approach, we can conclude that the mortality series is stationary at 5% level of significance as p value is less than 0.05.

1.3.5 Particle Size

```
plot(psize.ts,main="Time Series Plot for Particle Size Series",
     ylab="Particle Size",xlab="Year",type="o")
```

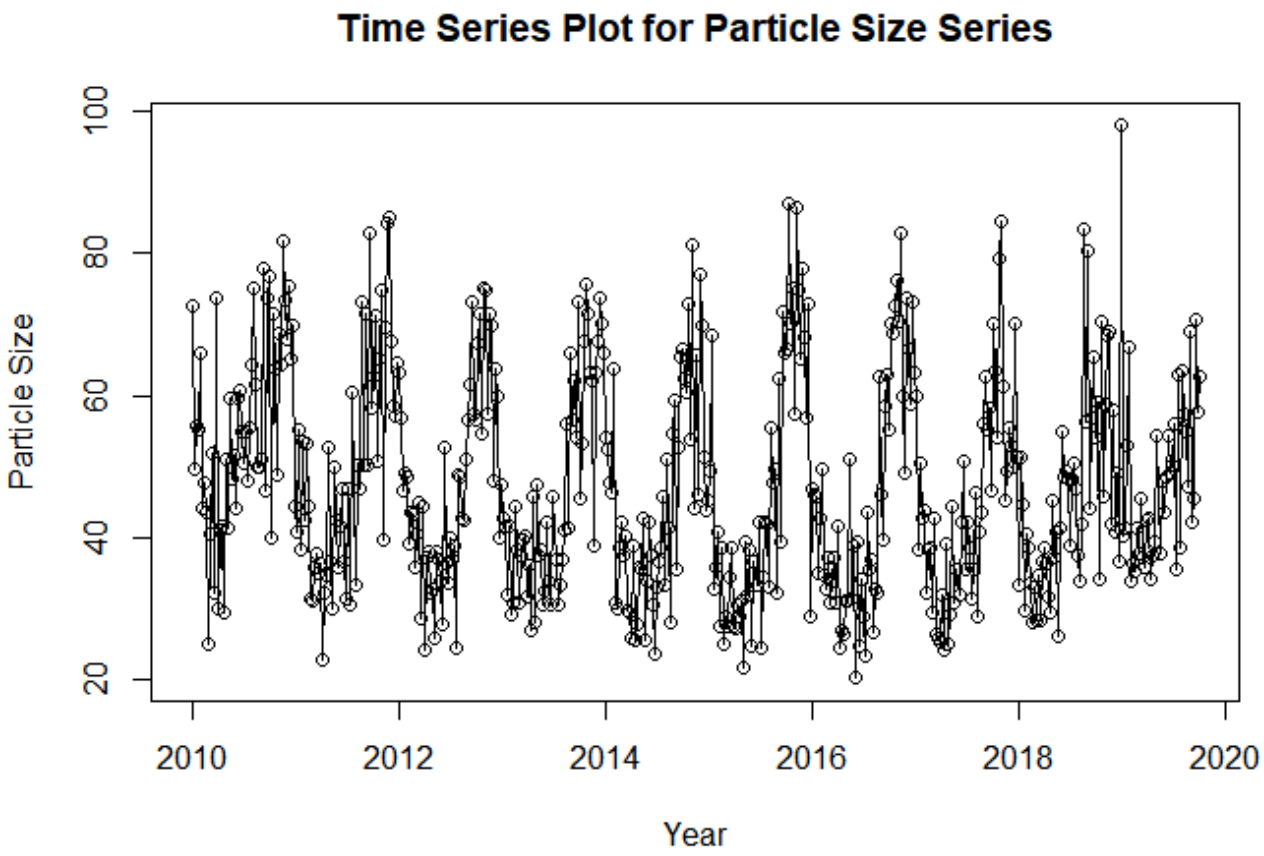

Figure 9

By `Figure 9`, which shows the time series plot of particle size series, we can make the following observations:

- The series has no apparent trend and no intervention points observed
- There are presence of seasonality.
- Since there are seasonality, changing variance and moving average behavior is not obvious.

```
acf(psize.ts, lag.max = 48, main="Sample ACF Plot for Particle Size Series")
```

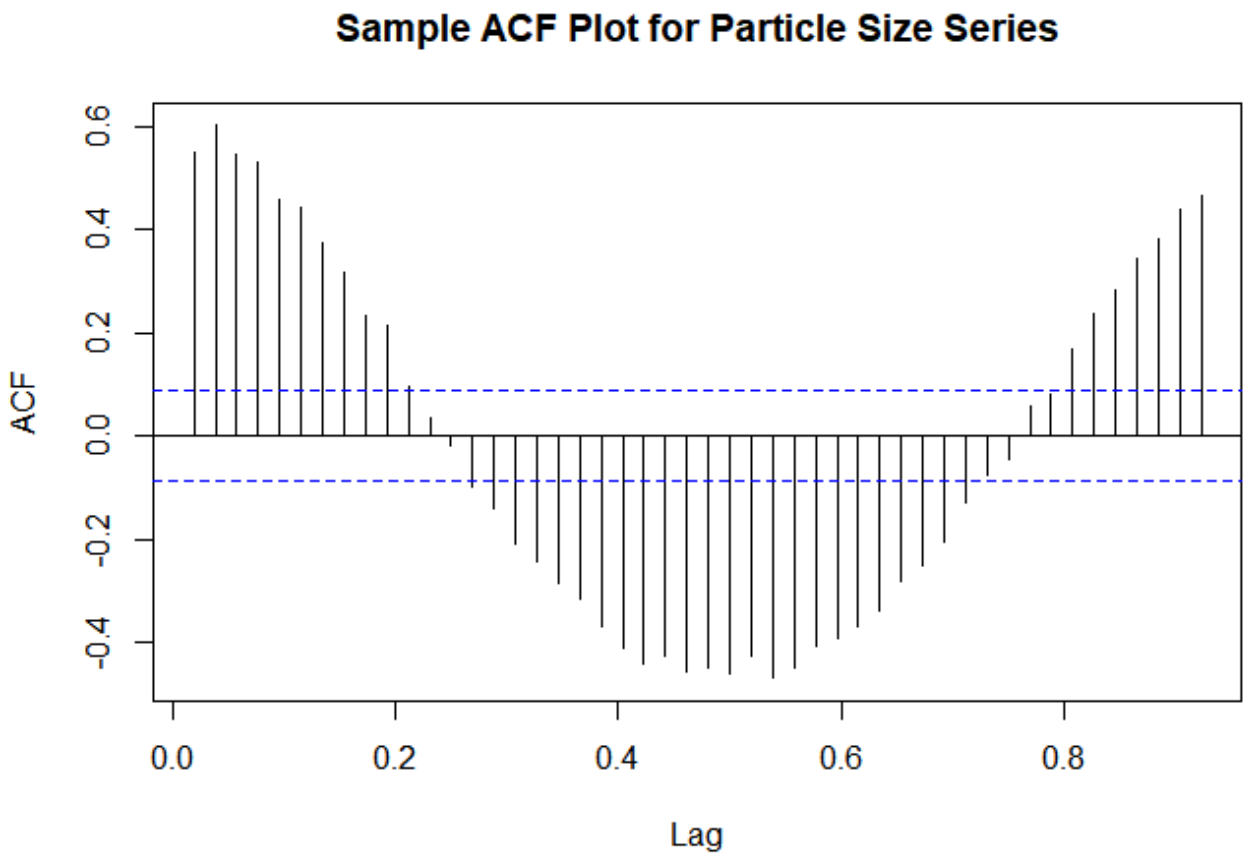

Figure 10

`Figure 10` shows the sample ACF plot for particle size series. Here we can observe that there is no apparent trend in the series, but presence of seasonal patterns.

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

```
## Warning in adf.test(psize.ts, k = ar(psize.ts)$order): p-value smaller than
## printed p-value
```

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

In the above output, we used the ADF test for the particle size series. In this approach, we can conclude that the mortality series is stationary at 5% level of significance as p value is less than 0.05.

1.3.6 Scaled Data Series

```
data1.scaled<-scale(data1.ts)

data1.col<-c("darkolivegreen","coral2","blueviolet","deeppink3","aquamarine")
```

```
plot(data1.scaled,plot.type="s",col=data1.col,main="Time Series Plot of Scaled Data Series")
legend("topleft",cex=0.7,lty=1,col=data1.col,c("Mortality","Temperature","Chem1","Chem2","Particle
        Size"))
```

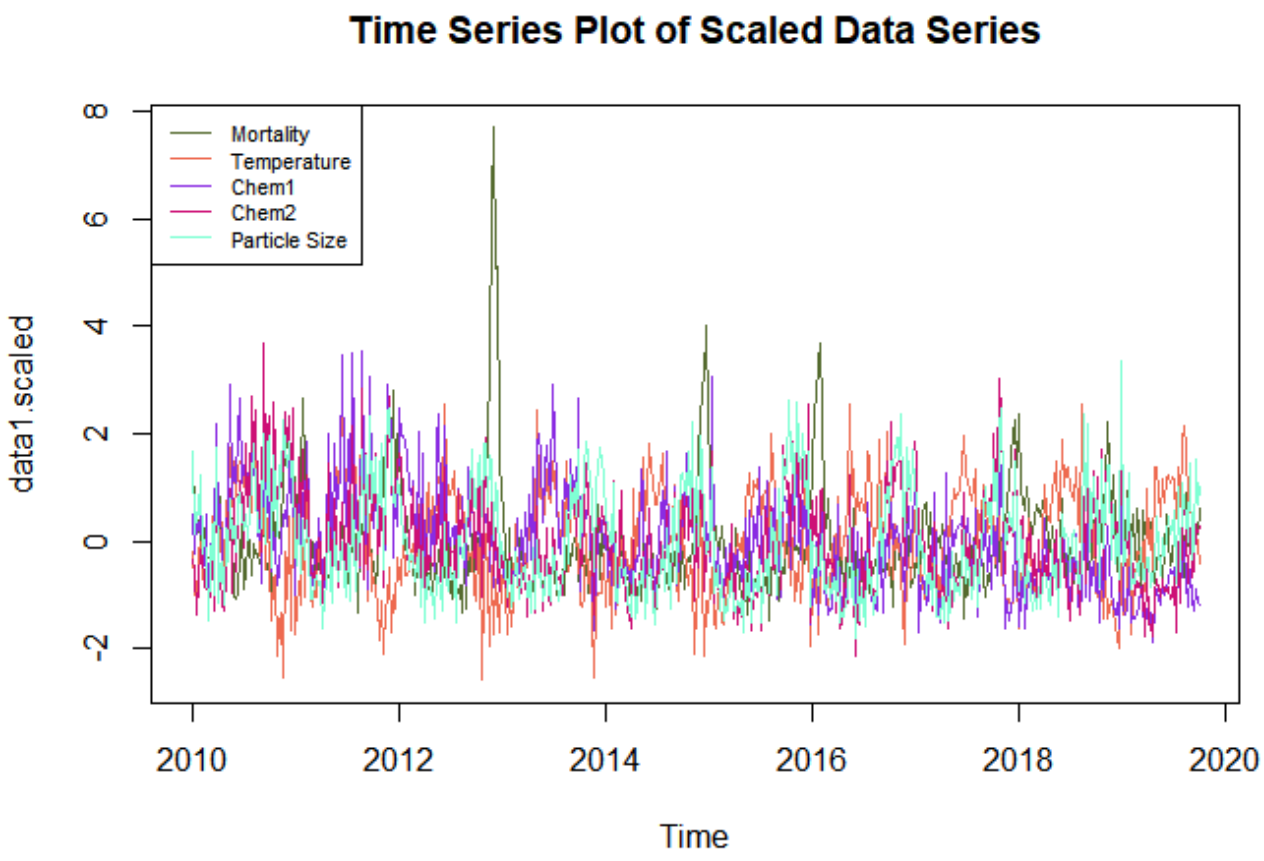

Figure 11

`Figure 11` contains a time series plot of all series. To compare these data and fit them in a graph, scaling and centering is done to the data by using the `scale` function.

1.4.0 Decomposition

To look into the impact of the components in the time series data, we will use STL decomposition method to have a further insight of the independent seasonal and trend characteristics of the time series data. We will use the functions `stl` to decompose and visualize the time series data. We will not be using the classical decomposition method as there are other methods more efficient than this method.

1.4.1 Mortality

```
mort.stl<-stl(mort.ts,t.window=15,s.window="periodic",robust=TRUE)
plot(mort.stl, main="STL Decomposition of Mortality Series")
```

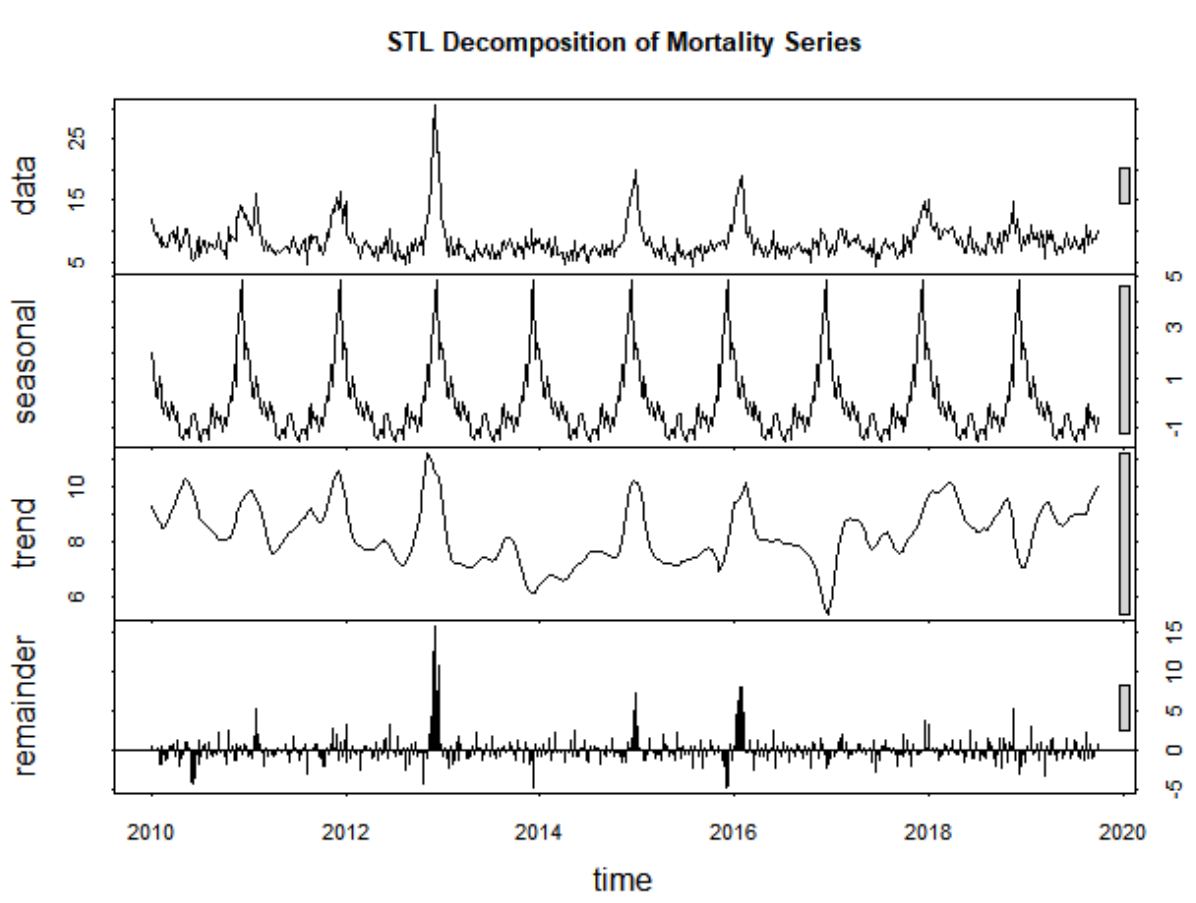

Figure 12

`Figure 12` shows the STL graph for mortality, where we can observe that the there is trend, and the seasonal effect shows a constant pattern behaviour. The remainder shows an intervention point around 2013.

1.4.2 Temperature

```
temp.stl<-stl(temp.ts,t.window=15,s.window="periodic",robust=TRUE)
plot(temp.stl,main="STL Decomposition of Temperature Series")
```

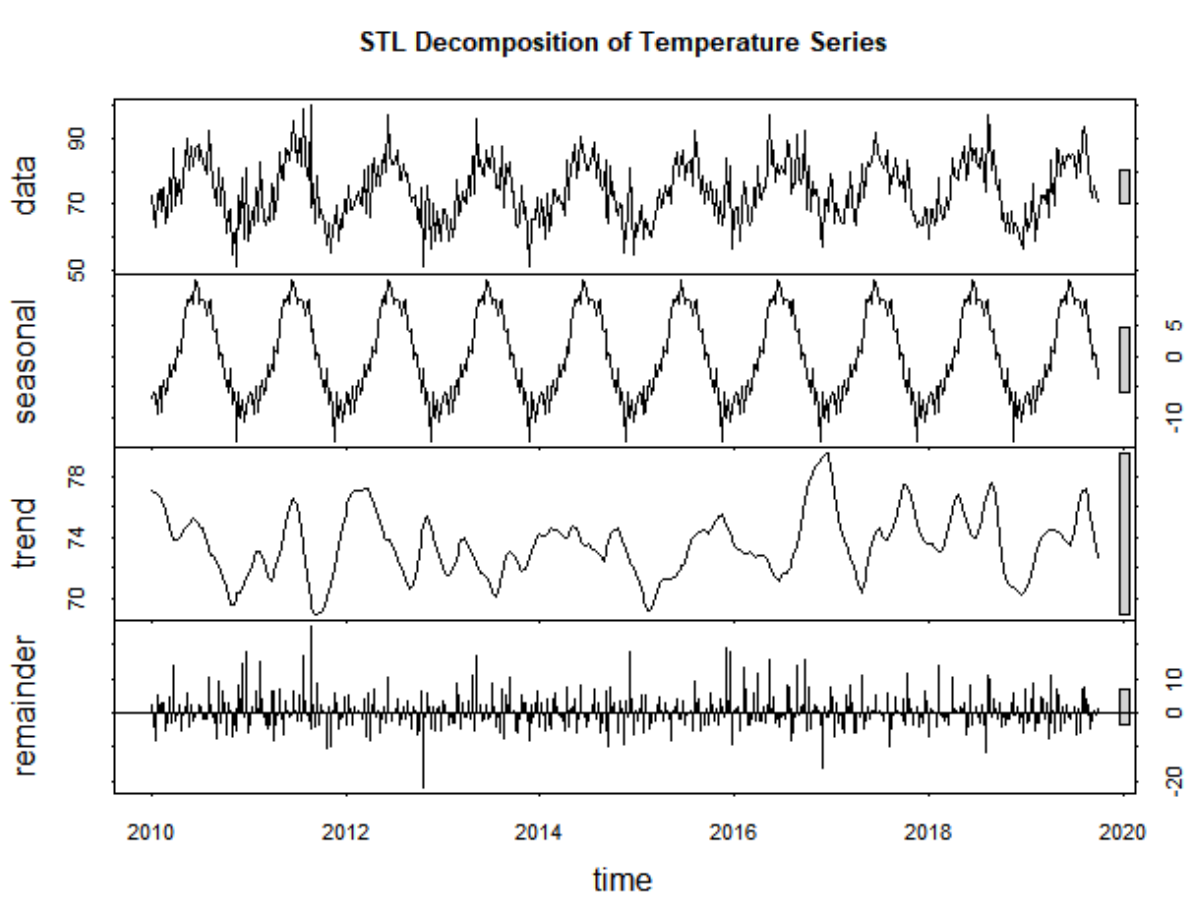

Figure 13

`Figure 13` shows the STL graph for temperature, where we can observe that the trend follows the data, and there is a seasonality pattern.

1.4.3 Chemical Emission 1

```
chem1.stl<-stl(chem1.ts,t.window=15,s.window="periodic",robust=TRUE)
plot(chem1.stl,main="STL Decomposition of Chemical Emission 1 Series")
```

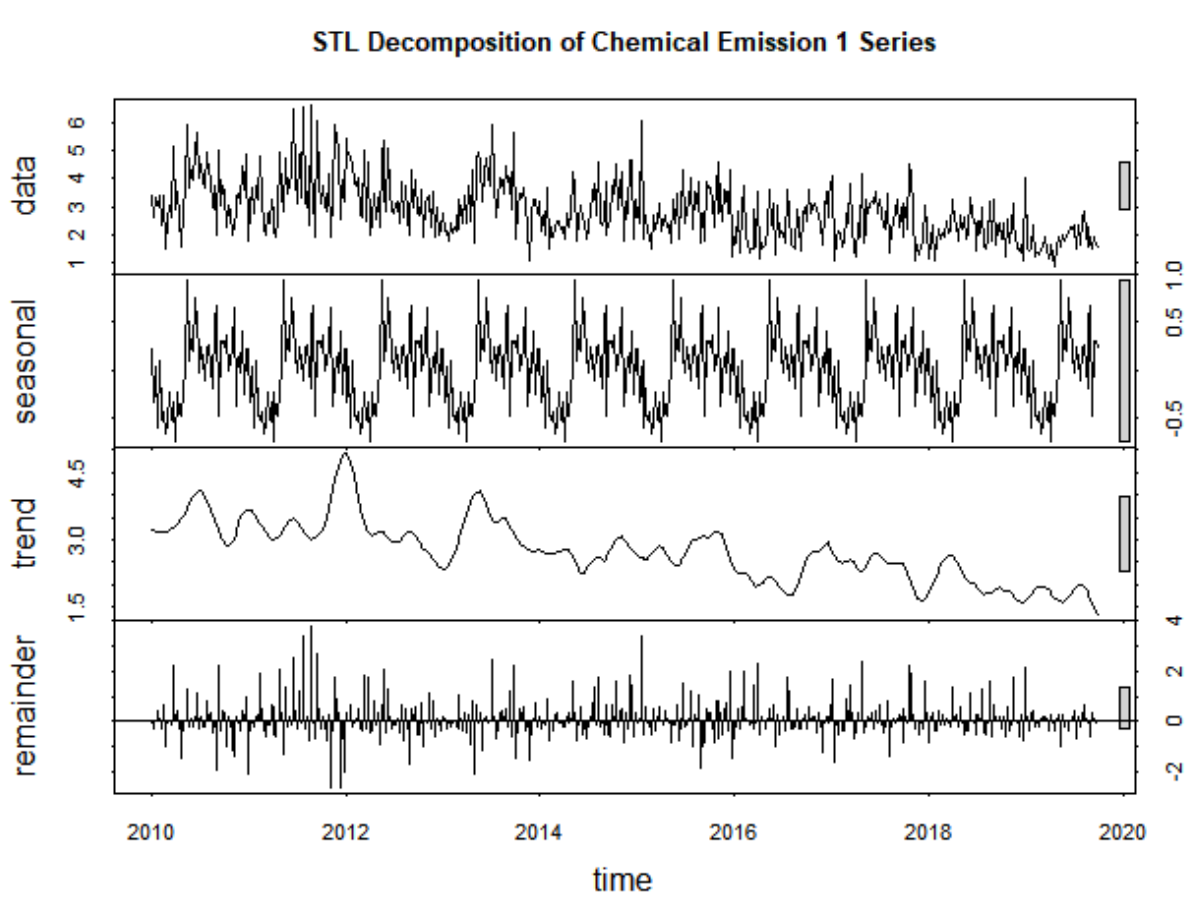

Figure 14

`Figure 14` shows the STL graph for chemical emission 1, where we can observe that the there is a downward trend, and there is a seasonality pattern.

1.4.4 Chemical Emission 2

```
chem2.stl<-stl(chem2.ts,t.window=15,s.window="periodic",robust=TRUE)
plot(chem2.stl,main="STL Decomposition of Chemical Emission 2 Series")
```

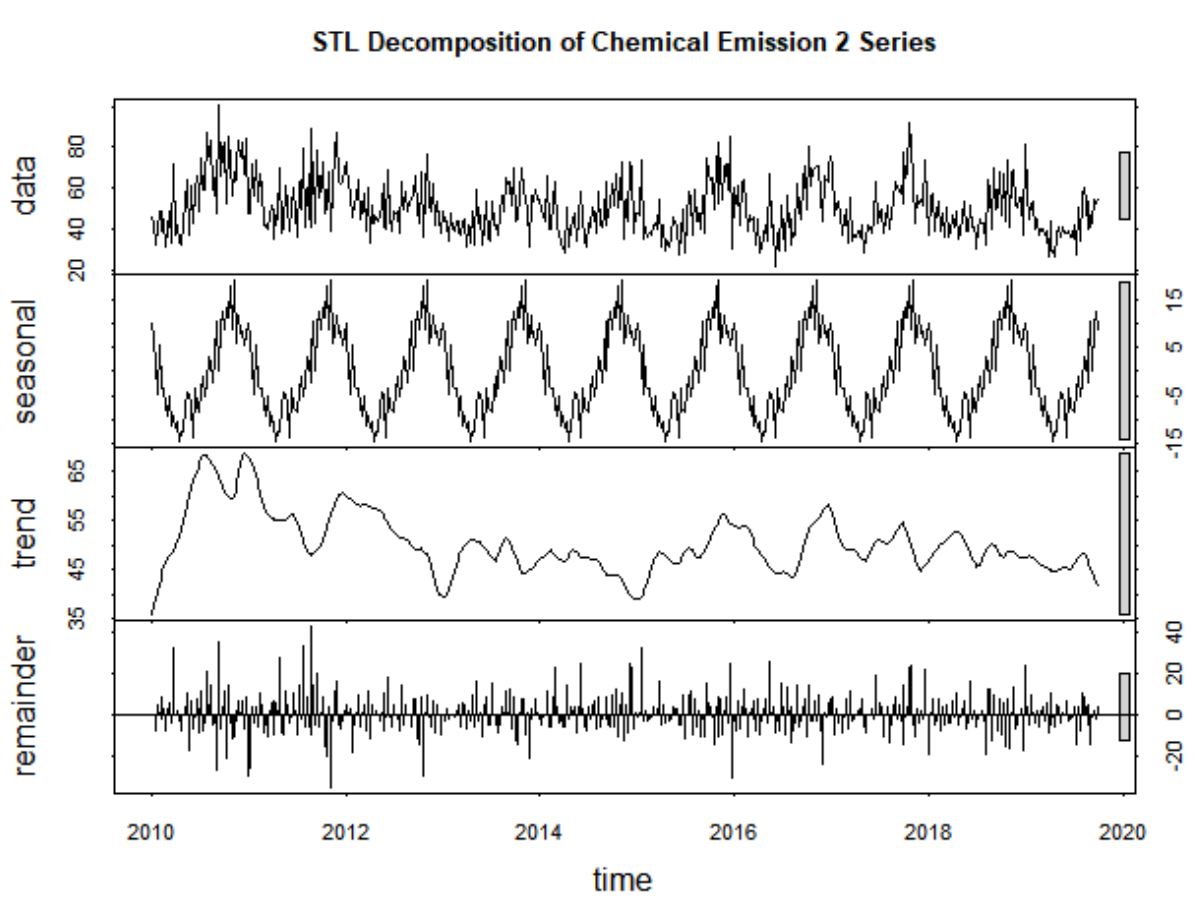

Figure 15

`Figure 15` shows the STL graph for chemical emission 2, where we can observe that the trend follows the data, and there is a seasonality pattern.

1.4.5 Particle Size

```
psize.stl<-stl(psize.ts,t.window=15,s.window="periodic",robust=TRUE)
plot(psize.stl,main="STL Decomposition of Particle Size Series")
```

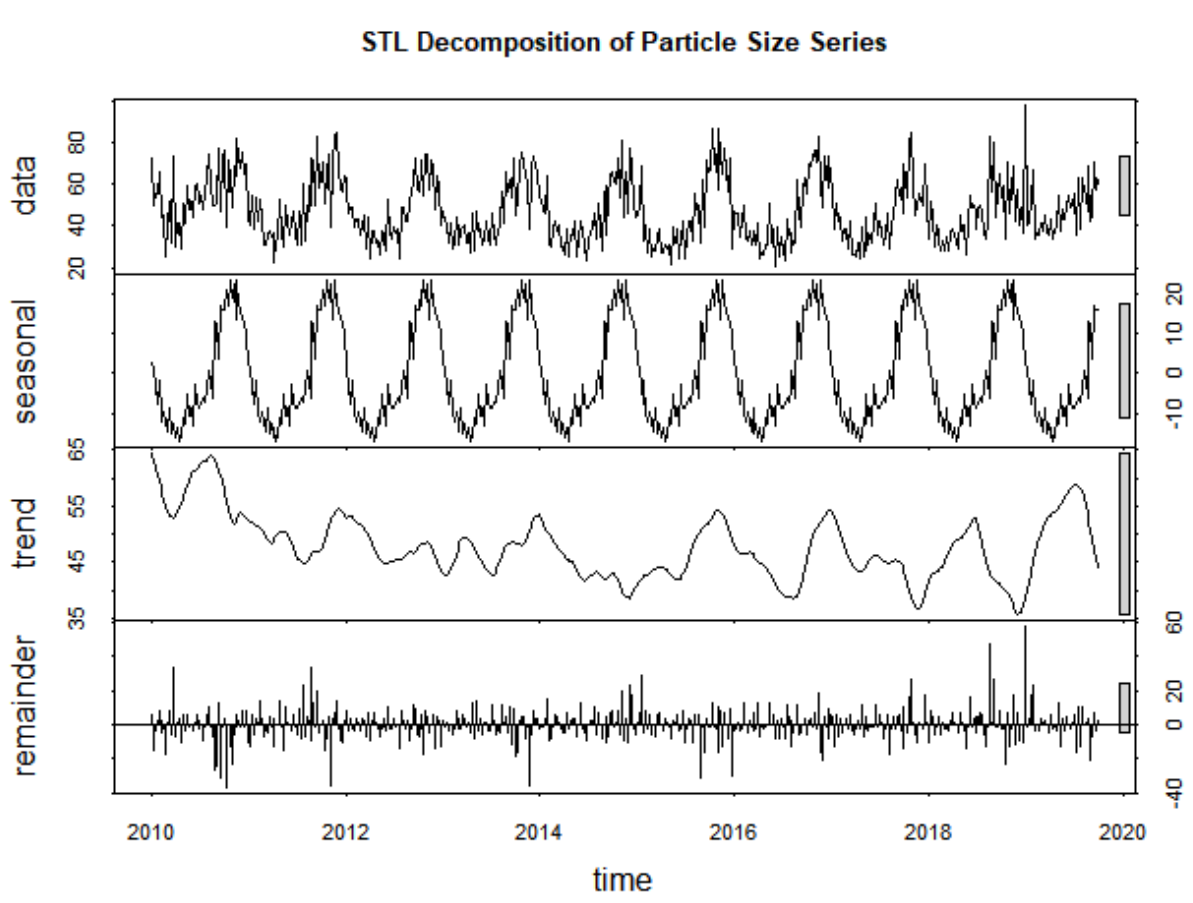

Figure 16

`Figure 16` shows the STL graph for particle size, where we can observe that the trend follows the data, and there is a seasonality pattern.

1.5.0 Time Series Regression

In this section, we will first look at the correlation matrix for the data set by using the `cor` function.

```
cor(data1.ts)
```

```
##                 mortality        temp      chem1     chem2 particle size
## mortality      1.00000000 -0.33055755 0.00409079 0.2157476    0.26498535
## temp          -0.33055755  1.00000000 0.40437401 0.1141372   -0.01723095
## chem1          0.00409079  0.40437401 1.00000000 0.6128658    0.46793404
## chem2          0.21574756  0.11413717 0.61286575 1.0000000    0.80750391
## particle size  0.26498535 -0.01723095 0.46793404 0.8075039    1.00000000
```

From the correlation matrix output above, we can observe that with mortality data there is a low negative correlation with temperature of -0.33055755, little to no correlation with chemical emission 1 of 0.00409079, little to no correlation with chemical emission 2 of 0.21574756, and little to no correlation with particle size of 0.26498535.

Thus, mortality should be used as dependent variable y, and temperature, chemical emission 1, chemical emission 2, and particle size as independent variable x.

1.5.1 Finite Distributed Lag Model

For Finite DLM, we will be using mortality as the dependent variable, and the rest of the variables will be the independent variable. To perform this test, we will first create a loop function and use the `dlm` function to identify most appropriate number of lags for the DLM.

```
for (i in 12){
  model1<-dlm(x=as.vector(mort.ts), y=as.vector(temp.ts)+as.vector(chem1.ts)+
                as.vector(chem2.ts)+as.vector(psize.ts), q=i)
  cat("q =",i,"AIC =",AIC(model1$model),"BIC =",BIC(model1$model),"MASE=",
      MASE(model1)$MASE, "\n")
}
```

```
## q = 12 AIC = 4742.193 BIC = 4805.291 MASE= 0.8204343
```

```
model1.2<-dlm(x=as.vector(mort.ts), y=as.vector(temp.ts)+as.vector(chem1.ts)+
                as.vector(chem2.ts)+as.vector(psize.ts), q=12)
summary(model1.2)
```

```
##
## Call:
## lm(formula = model.formula, data = design)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -65.070 -19.908  -2.939  18.870  89.896
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 198.98371    6.06894  32.787  < 2e-16 ***
## x.t           2.49649    0.72757   3.431 0.000652 ***
## x.1          -0.62094    0.85706  -0.724 0.469111
## x.2          -0.34874    0.88150  -0.396 0.692559
## x.3          -0.09847    0.88111  -0.112 0.911060
## x.4          -0.68849    0.88136  -0.781 0.435087
## x.5          -0.05768    0.88270  -0.065 0.947928
## x.6          -0.08889    0.88430  -0.101 0.919970
## x.7          -0.16515    0.88237  -0.187 0.851613
## x.8          -0.32754    0.88629  -0.370 0.711872
## x.9          -0.83778    0.88901  -0.942 0.346472
## x.10          0.03685    0.88913   0.041 0.966954
## x.11         -0.82669    0.86632  -0.954 0.340436
## x.12         -1.30313    0.73082  -1.783 0.075199 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.38 on 482 degrees of freedom
## Multiple R-squared:  0.1257, Adjusted R-squared:  0.1021
```

```
## F-statistic: 5.332 on 13 and 482 DF,  p-value: 5.364e-09
##
## AIC and BIC values for the model:
##        AIC      BIC
## 1 4742.193 4805.291
```

From the output above, we can see that the recommended amount of lag is 12. Hence, we used the `dlm` function with a argument `q=12`. By using the `summary` function, we can observe that in `model1.2` the lag weights of the predictors are significant at the 5% statistical level as p-value < 0.05. Furthermore, there is a adjusted R-square value of 0.1021, which accounts for 10.21% of the variability in the data.

```
checkresiduals(model1.2$model$residuals)
```

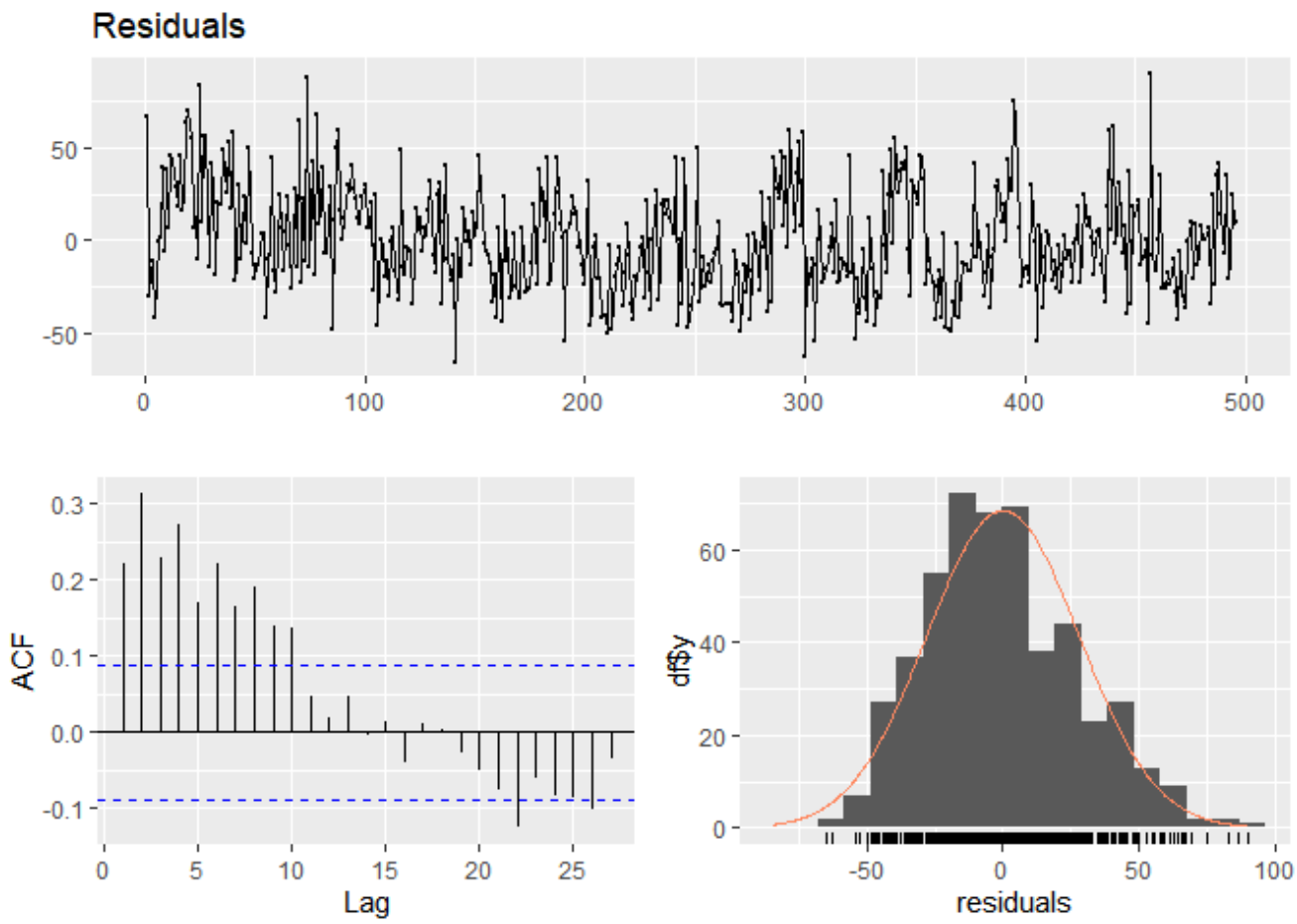

Figure 17

```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 227.23, df = 10, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 10
```

```
bgtest(model1.2$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model1.2$model
## LM test = 24.375, df = 1, p-value = 7.929e-07
```

```
shapiro.test(model1.2$model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model1.2$model$residuals
## W = 0.98432, p-value = 3.476e-05
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 17`. The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are highly significant. Furthermore, from the Shapiro-Wilk test we can conclude that there is enough evidence to reject null hypothesis of normality as p-value < 0.05.

```
VIF.model1.2<-vif(model1.2$model)
VIF.model1.2
```

```
##      x.t      x.1      x.2      x.3      x.4      x.5      x.6      x.7
## 2.721520 3.775450 3.993960 3.991150 3.993991 4.004302 4.019145 4.002234
##      x.8      x.9     x.10     x.11     x.12
## 4.031383 4.054103 4.055991 3.854371 2.750330
```

```
VIF.model1.2>10
```

```
##   x.t   x.1   x.2   x.3   x.4   x.5   x.6   x.7   x.8   x.9  x.10  x.11  x.12
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the `vif` function, we can observe that the estimates of the finite DLM coefficients are not suffering from the multicollinearity. The reason for this is because all the data in the VIF are less than 10.

We can conclude that the finite DLM of lag 12 is not successful at capturing the autocorrelation and seasonality in the series.

1.5.2 Polynomial Distributed Lag Model

Polynomial DLM imposes a polynomial shape on the lag distribution to reduce the effect of multicollinearity. For the Polynomial DLM, we will be using mortality as our dependent variable, and particle size as our dependent variable as it has the highest positive correlation value.

```
model2.1<-polyDlm(x=as.vector(mort.ts),y=as.vector(psize.ts),q=12,k=2,show.beta=TRUE)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value  P(>|t|)
## beta.0    0.5940     0.1210   4.920 1.19e-06
## beta.1    0.4260     0.0772   5.510 5.81e-08
## beta.2    0.2740     0.0480   5.700 2.04e-08
## beta.3    0.1390     0.0397   3.500 5.09e-04
## beta.4    0.0209     0.0468   0.447 6.55e-01
## beta.5   -0.0801     0.0549  -1.460 1.45e-01
## beta.6   -0.1640     0.0580  -2.830 4.83e-03
## beta.7   -0.2310     0.0549  -4.210 2.99e-05
## beta.8   -0.2820     0.0468  -6.020 3.53e-09
## beta.9   -0.3150     0.0397  -7.940 1.47e-14
## beta.10  -0.3310     0.0479  -6.910 1.52e-11
## beta.11  -0.3310     0.0771  -4.290 2.16e-05
## beta.12  -0.3130     0.1210  -2.600 9.68e-03
```

```
summary(model2.1, diagnostics=TRUE)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -27.677 -10.219  -2.315   9.733  55.632
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 52.356177   2.990360  17.508  < 2e-16 ***
## z.t0         0.594250   0.120805   4.919 1.19e-06 ***
## z.t1        -0.177172   0.052981  -3.344 0.000889 ***
## z.t2         0.008461   0.004347   1.946 0.052192 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.16 on 492 degrees of freedom
## Multiple R-squared:  0.1361, Adjusted R-squared:  0.1309
## F-statistic: 25.84 on 3 and 492 DF,  p-value: 1.544e-15
```

In the `model2.1` output above, we have used the `polyDLM` function to perform the DLM. Similarly in `model1.2` we used a `q=12` similar to Finite DLM and `k=2`. Besides that, an argument of `show.beta=TRUE` is used to generate estimates of lag weights and their standard deviations using standard regression.

In the summary for `model2.1`, the lag weights of the predictors are significant at the 5% statistical level as p-value < 0.05. Furthermore, the adjusted R-squared shows a value of 0.1309, which accounts for 13.09% of the variability in the data.

```
checkresiduals(model2.1$model$residuals)
```

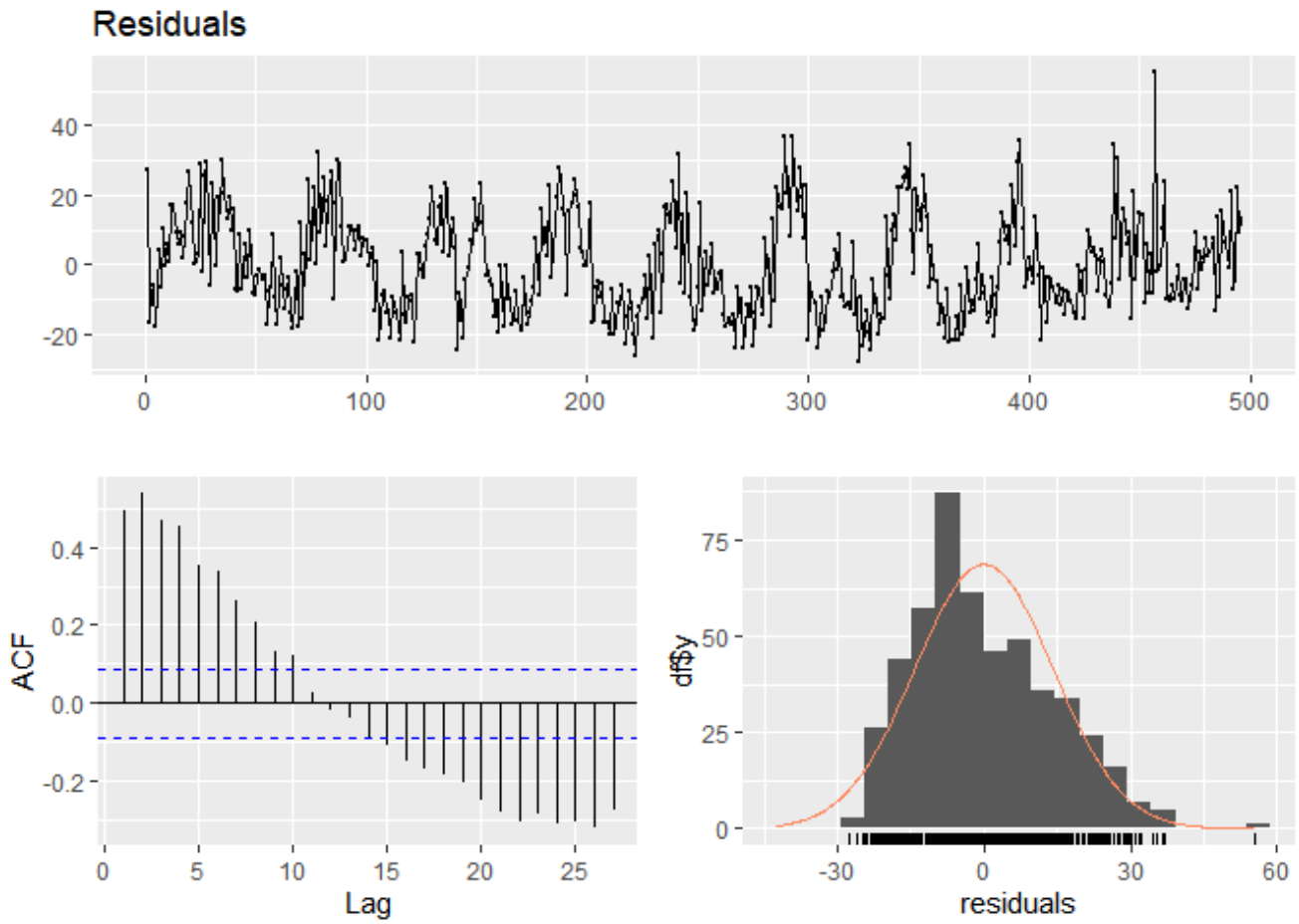

Figure 18

```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 672.47, df = 10, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 10
```

```
bgtest(model2.1$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model2.1$model
## LM test = 122.06, df = 1, p-value < 2.2e-16
```

```
shapiro.test(model2.1$model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model2.1$model$residuals
## W = 0.97203, p-value = 3.958e-08
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 18`. The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are highly significant. Furthermore, from the Shapiro-Wilk test we can conclude that there is enough evidence to reject null hypothesis of normality as p-value < 0.05.

```
VIF.model2.1<-vif(model2.1$model)
VIF.model2.1
```

```
##      z.t0      z.t1      z.t2
##  22.63754 189.97734 100.73846
```

```
VIF.model2.1>10
```

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

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the `vif` function, we can observe that there is an issue with multicollinearity as VIF values are greater than 10 for z.t1, z.t2 and z.t3.

Similar to the Finite DLM, this model is not successful at capturing the autocorrelation and seasonality in the series.

1.5.3 Koyck Distributed Lag Model

In the Koyck DLM similar to Polynomial DLM, we will use the dependent variable mortality and independent variable particle size. The Koyck DLM transformation is useful when dealing with infinite DLM.

```
model3.1<-koyckDlm(x=as.vector(mort.ts),y=as.vector(psize.ts))
summary(model3.1, diagnostics=TRUE)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -34.459  -8.821  -1.400   7.284  56.519
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 19.41244    2.41585   8.035 6.66e-15 ***
## Y.1          0.53767    0.03923  13.704  < 2e-16 ***
## X.t          0.29528    0.26587   1.111    0.267
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.52 on 504 degrees of freedom
## Multiple R-Squared: 0.3163,  Adjusted R-squared: 0.3136
## Wald test: 113.7 on 2 and 504 DF,  p-value: < 2.2e-16
##
## Diagnostic tests:
##                  df1 df2  statistic      p-value
## Weak instruments   1 504 686.492895 3.969781e-96
## Wu-Hausman         1 503   3.872098 4.964406e-02
##
##                             alpha      beta       phi
## Geometric coefficients:  41.98822 0.2952818 0.5376693
```

In the `model3.1` output above, we have used the function `koyckDLM` to perform the Koyck DLM Transformation. Since the Koyck DLM approach does not produce a standard error, we have added an argument `diagnostics=TRUE` as it will show the F test, Wu-Hausman test, and the Sargan Test.

From the summary output, we can see that in the Weak Instruments line we can conclude that the model at the first stage of the least-squares fitting is significant at 5% level of significance.

Besides that, we can also observe that in the Wu-Hausman test, there is a significant correlation between the explanatory variable and the error term at 5% level as p-value < 0.05. Furthermore, the adjusted R-squared shows a value of 0.3136, which accounts for 31.36% of the variability in the data.

```
checkresiduals(model3.1$model$residuals)
```

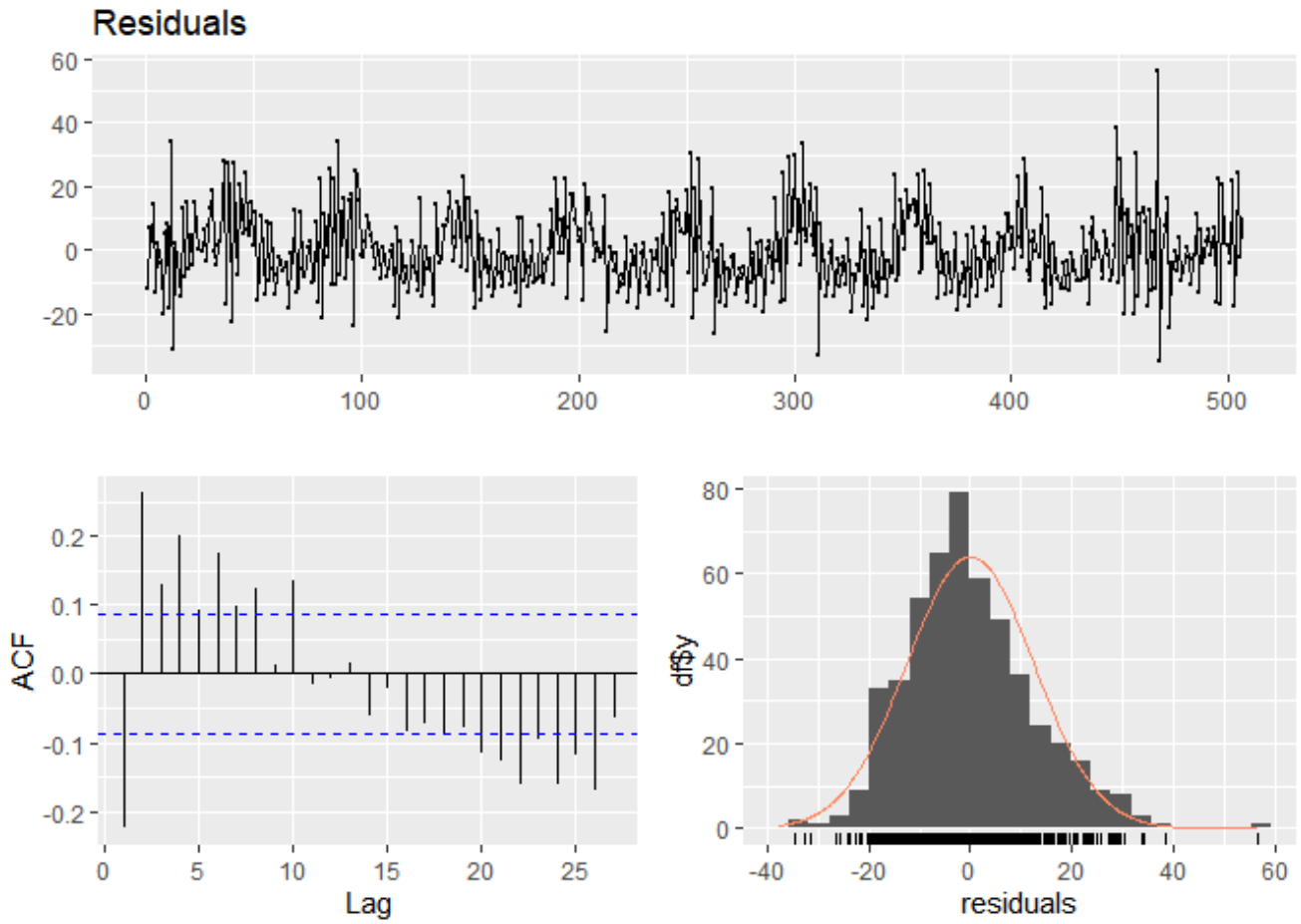

Figure 19

```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 133.36, df = 10, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 10
```

```
bgtest(model3.1$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model3.1$model
## LM test = 74.821, df = 1, p-value < 2.2e-16
```

```
shapiro.test(model3.1$model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
```

```
## data:  model3.1$model$residuals
## W = 0.98238, p-value = 8.146e-06
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 19`. The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are highly significant. Furthermore, from the Shapiro-Wilk test we can conclude that there is enough evidence to reject null hypothesis of normality as p-value < 0.05.

```
VIF.model3.1<-vif(model3.1$model)
VIF.model3.1
```

```
##      Y.1      X.t
## 1.138845 1.138845
```

```
VIF.model3.1>10
```

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

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the `vif` function, we can observe that there the data is not suffering from multicollinearity. The reason for this is because all the data in the VIF are less than 10.

Similar to the models before, this model is not successful at capturing the autocorrelation and seasonality in the series.

1.5.4 Autoregressive Distributed Lag Model

Autoregressive DLM is normally used when a suitable solution with other DLMs cannot be found for infinite DLM. Similarly to Finite DLM, we will be using the predictors temperature, chemical emission 1, chemical emission 2, and particle size as the independent variable, and the mortality will be the dependent variable. To perform this test, we will first create a loop function for p and q and use the `ardlDLM` function to identify most appropriate number of lags for the DLM.

```
for (i in 1:5){
  for(j in 1:5){
    model4.1 = ardlDlm(x = as.vector(mort.ts), y = as.vector(temp.ts)+as.vector(chem1.ts)+
                as.vector(chem2.ts)+as.vector(psize.ts), p = i , q = j)
    cat("p =", i, "q =", j, "AIC =", AIC(model4.1$model), "BIC =", BIC(model4.1$model), "MASE
        =", MASE(model4.1)$MASE, "\n")
  }
}
```

```
## p = 1 q = 1 AIC = 4829.803 BIC = 4850.945 MASE = 0.8162666
## p = 1 q = 2 AIC = 4767.509 BIC = 4792.869 MASE = 0.7780571
## p = 1 q = 3 AIC = 4745.252 BIC = 4774.824 MASE = 0.7649847
## p = 1 q = 4 AIC = 4725.539 BIC = 4759.319 MASE = 0.7559747
## p = 1 q = 5 AIC = 4716.788 BIC = 4754.773 MASE = 0.7547836
## p = 2 q = 1 AIC = 4821.092 BIC = 4846.452 MASE = 0.8148253
## p = 2 q = 2 AIC = 4764.657 BIC = 4794.243 MASE = 0.7750697
## p = 2 q = 3 AIC = 4742.672 BIC = 4776.469 MASE = 0.7635743
## p = 2 q = 4 AIC = 4722.715 BIC = 4760.718 MASE = 0.7545837
## p = 2 q = 5 AIC = 4713.936 BIC = 4756.141 MASE = 0.753091
## p = 3 q = 1 AIC = 4813.125 BIC = 4842.697 MASE = 0.8124292
## p = 3 q = 2 AIC = 4757.329 BIC = 4791.125 MASE = 0.7739711
## p = 3 q = 3 AIC = 4743.383 BIC = 4781.404 MASE = 0.7619616
## p = 3 q = 4 AIC = 4723.311 BIC = 4765.537 MASE = 0.7520091
## p = 3 q = 5 AIC = 4714.343 BIC = 4760.77 MASE = 0.7504787
## p = 4 q = 1 AIC = 4803.087 BIC = 4836.868 MASE = 0.8081695
## p = 4 q = 2 AIC = 4748.593 BIC = 4786.596 MASE = 0.7701754
## p = 4 q = 3 AIC = 4735.251 BIC = 4777.477 MASE = 0.7608335
## p = 4 q = 4 AIC = 4721.633 BIC = 4768.081 MASE = 0.7478292
## p = 4 q = 5 AIC = 4712.366 BIC = 4763.013 MASE = 0.7452085
## p = 5 q = 1 AIC = 4794.29 BIC = 4832.275 MASE = 0.808486
## p = 5 q = 2 AIC = 4739.945 BIC = 4782.151 MASE = 0.7695905
## p = 5 q = 3 AIC = 4726.683 BIC = 4773.11 MASE = 0.7608595
## p = 5 q = 4 AIC = 4713.612 BIC = 4764.259 MASE = 0.7480214
## p = 5 q = 5 AIC = 4713.349 BIC = 4768.217 MASE = 0.7445786
```

From the loop function output above, we can see that p = 4, q = 5 contains the lowest AIC value, p = 1, q = 5 contains the lowest BIC value, and p = 5, q = 5 contains the lowest MASE value. Hence, we will be using these 3 values for the Autoregressive DLM.

```
model4.2<-ardlDlm(x=as.vector(mort.ts),y=as.vector(temp.ts)+as.vector(chem1.ts)+
                as.vector(chem2.ts)+as.vector(psize.ts),p=4,q=5)
summary(model4.2)
```

```
##
## 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
## -83.983 -18.478  -2.095  16.613  93.543
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 62.612571  10.319217   6.068 2.59e-09 ***
## X.t          1.655917   0.667363   2.481 0.013424 *
```

```
## X.1         -0.721036   0.783147  -0.921 0.357663
## X.2         -0.538083   0.804660  -0.669 0.503995
## X.3         -0.008027   0.783379  -0.010 0.991829
## X.4         -1.318301   0.667097  -1.976 0.048693 *
## Y.1          0.088112   0.044919   1.962 0.050376 .
## Y.2          0.232695   0.044446   5.235 2.44e-07 ***
## Y.3          0.137367   0.045188   3.040 0.002492 **
## Y.4          0.168712   0.044639   3.780 0.000176 ***
## Y.5          0.060061   0.044837   1.340 0.181014
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.86 on 492 degrees of freedom
## Multiple R-squared:  0.268,  Adjusted R-squared:  0.2531
## F-statistic: 18.01 on 10 and 492 DF,  p-value: < 2.2e-16
```

From the above output for `model4.2`, we can see that the lag weights are significant at the 5% statistical level as p-value < 0.05. Furthermore, the adjusted R-squared shows a value of 0.2531, which accounts for 25.31% of the variability in the data.

```
checkresiduals(model4.2$model$residuals)
```

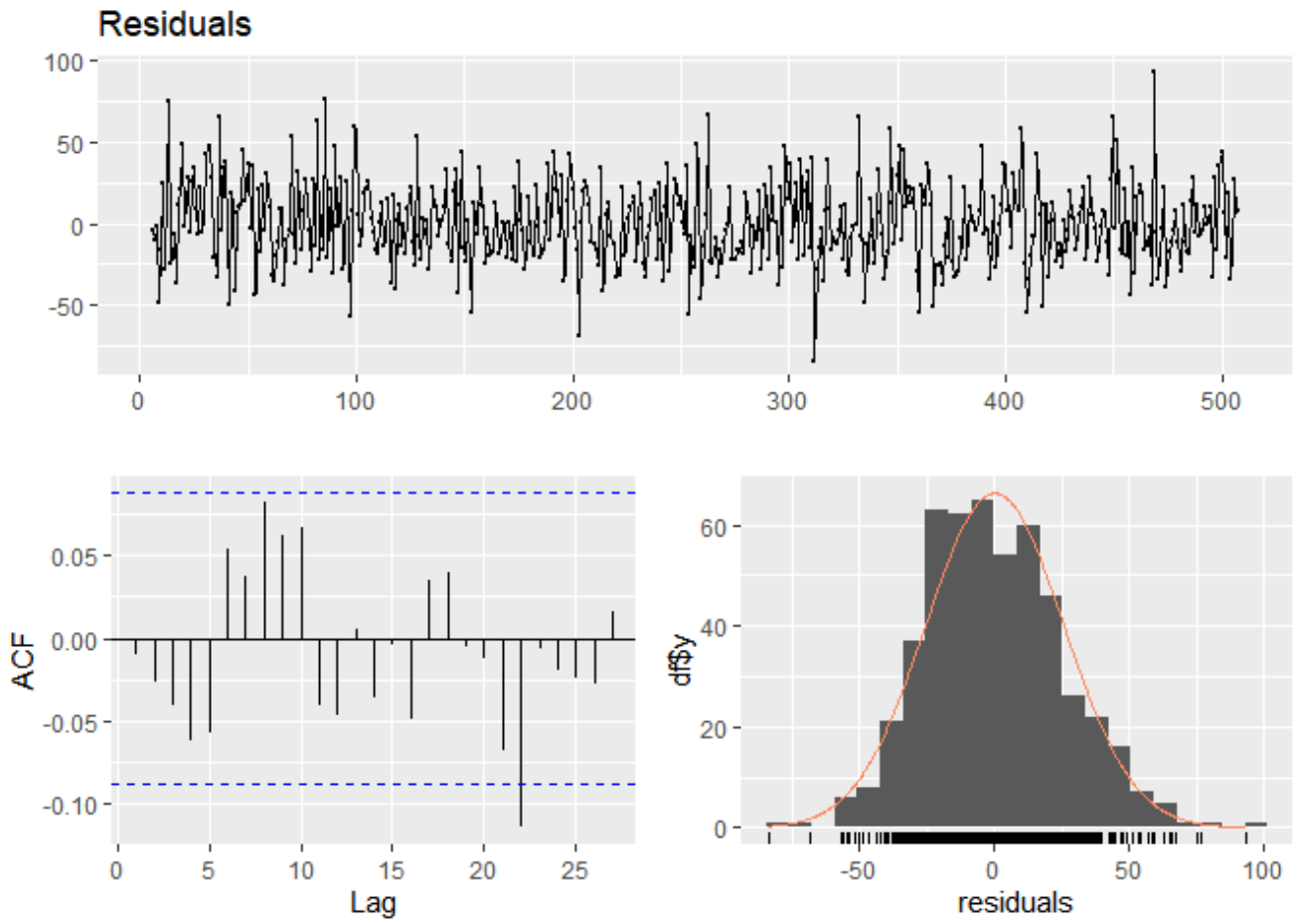

Figure 20

```
##
##  Ljung-Box test
##
```

```
## data:  Residuals
## Q* = 14.654, df = 10, p-value = 0.1452
##
## Model df: 0.   Total lags used: 10
```

```
bgtest(model4.2$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model4.2$model
## LM test = 3.7439, df = 1, p-value = 0.053
```

```
shapiro.test(model4.2$model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model4.2$model$residuals
## W = 0.9928, p-value = 0.01613
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 20`. The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that there are no serial correlation in residuals.. Furthermore, from the Shapiro-Wilk test we can conclude that there is enough evidence to reject null hypothesis of normality as p-value < 0.05.

```
VIF.model4.2<-vif(model4.2$model)
VIF.model4.2
```

```
##       X.t L(X.t, 1) L(X.t, 2) L(X.t, 3) L(X.t, 4) L(y.t, 1) L(y.t, 2) L(y.t, 3)
##  2.761733  3.800382  4.013203  3.804491  2.762515  1.356411  1.327718  1.371303
## L(y.t, 4) L(y.t, 5)
##  1.338175  1.350619
```

```
VIF.model4.2>10
```

```
##       X.t L(X.t, 1) L(X.t, 2) L(X.t, 3) L(X.t, 4) L(y.t, 1) L(y.t, 2) L(y.t, 3)
##     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE
```

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

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the `vif` function, we can observe that there the data is not suffering from multicollinearity. The reason for this is because the data in the VIF are less than 10.

```
model4.3<-ardlDlm(x=as.vector(mort.ts),y=as.vector(temp.ts)+as.vector(chem1.ts)+
                as.vector(chem2.ts)+as.vector(psize.ts),p=1,q=5)
summary(model4.3)
```

```
##
## 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
## -82.392 -19.053  -2.249  17.869  91.423
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 54.85879   10.08234   5.441 8.34e-08 ***
## X.t          1.59806    0.65353   2.445 0.014822 *
## X.1         -1.82384    0.64522  -2.827 0.004893 **
## Y.1          0.10999    0.04465   2.463 0.014109 *
## Y.2          0.23725    0.04419   5.369 1.22e-07 ***
## Y.3          0.13902    0.04508   3.084 0.002156 **
## Y.4          0.15514    0.04451   3.486 0.000534 ***
## Y.5          0.05612    0.04513   1.244 0.214193
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.05 on 495 degrees of freedom
## Multiple R-squared:  0.2526, Adjusted R-squared:  0.2421
## F-statistic:  23.9 on 7 and 495 DF,  p-value: < 2.2e-16
```

From the above output for `model4.3`, we can see that the lag weights are significant at the 5% statistical level as p-value < 0.05. Furthermore, the adjusted R-squared shows a value of 0.2421, which accounts for 24.21% of the variability in the data.

```
checkresiduals(model4.3$model$residuals)
```

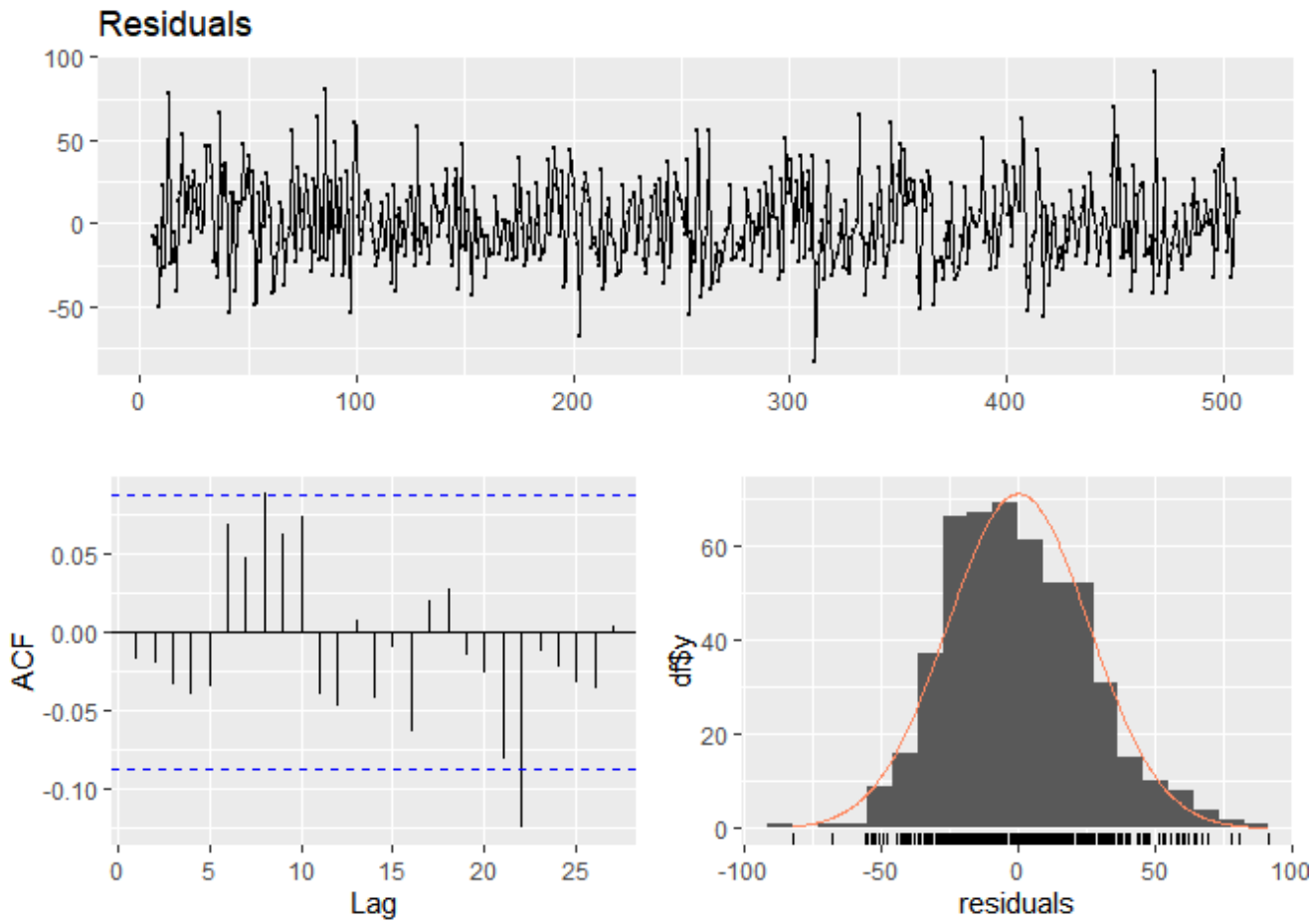

Figure 21

```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 14.837, df = 10, p-value = 0.1381
##
## Model df: 0.   Total lags used: 10
```

```
bgtest(model4.3$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model4.3$model
## LM test = 7.3822, df = 1, p-value = 0.006587
```

```
shapiro.test(model4.3$model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
```

```
## data:  model4.3$model$residuals
## W = 0.99083, p-value = 0.003254
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 21`. The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that there are no serial correlation in residuals.. Furthermore, from the Shapiro-Wilk test we can conclude that there is enough evidence to reject null hypothesis of normality as p-value < 0.05.

```
VIF.model4.3<-vif(model4.3$model)
VIF.model4.3
```

```
##       X.t L(X.t, 1) L(y.t, 1) L(y.t, 2) L(y.t, 3) L(y.t, 4) L(y.t, 5)
##  2.609934  2.542153  1.321004  1.293342  1.344775  1.310977  1.348188
```

```
VIF.model4.3>10
```

```
##       X.t L(X.t, 1) L(y.t, 1) L(y.t, 2) L(y.t, 3) L(y.t, 4) L(y.t, 5)
##     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE
```

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the `vif` function, we can observe that there the data is not suffering from multicollinearity. The reason for this is because the data in the VIF are less than 10.

```
model4.4<-ardlDlm(x=as.vector(mort.ts),y=as.vector(temp.ts)+as.vector(chem1.ts)+
                as.vector(chem2.ts)+as.vector(psize.ts),p=5,q=5)
summary(model4.4)
```

```
##
## 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
## -83.203 -18.143  -1.796  17.065  92.634
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 64.58512   10.50725   6.147 1.64e-09 ***
## X.t          1.60748    0.66913   2.402 0.016661 *
```

```
## X.1         -0.75415    0.78386  -0.962 0.336471
## X.2         -0.54005    0.80467  -0.671 0.502442
## X.3          0.17580    0.80479   0.218 0.827178
## X.4         -0.90630    0.78473  -1.155 0.248687
## X.5         -0.66968    0.67171  -0.997 0.319272
## Y.1          0.08400    0.04511   1.862 0.063189 .
## Y.2          0.23058    0.04450   5.182 3.21e-07 ***
## Y.3          0.13437    0.04529   2.967 0.003153 **
## Y.4          0.16660    0.04469   3.728 0.000215 ***
## Y.5          0.06767    0.04548   1.488 0.137426
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.86 on 491 degrees of freedom
## Multiple R-squared:  0.2694, Adjusted R-squared:  0.2531
## F-statistic: 16.46 on 11 and 491 DF,  p-value: < 2.2e-16
```

From the above output for `model4.4`, we can see that the lag weights are significant at the 5% statistical level as p-value < 0.05. Furthermore, the adjusted R-squared shows a value of 0.2531, which accounts for 25.31% of the variability in the data.

```
checkresiduals(model4.4$model$residuals)
```

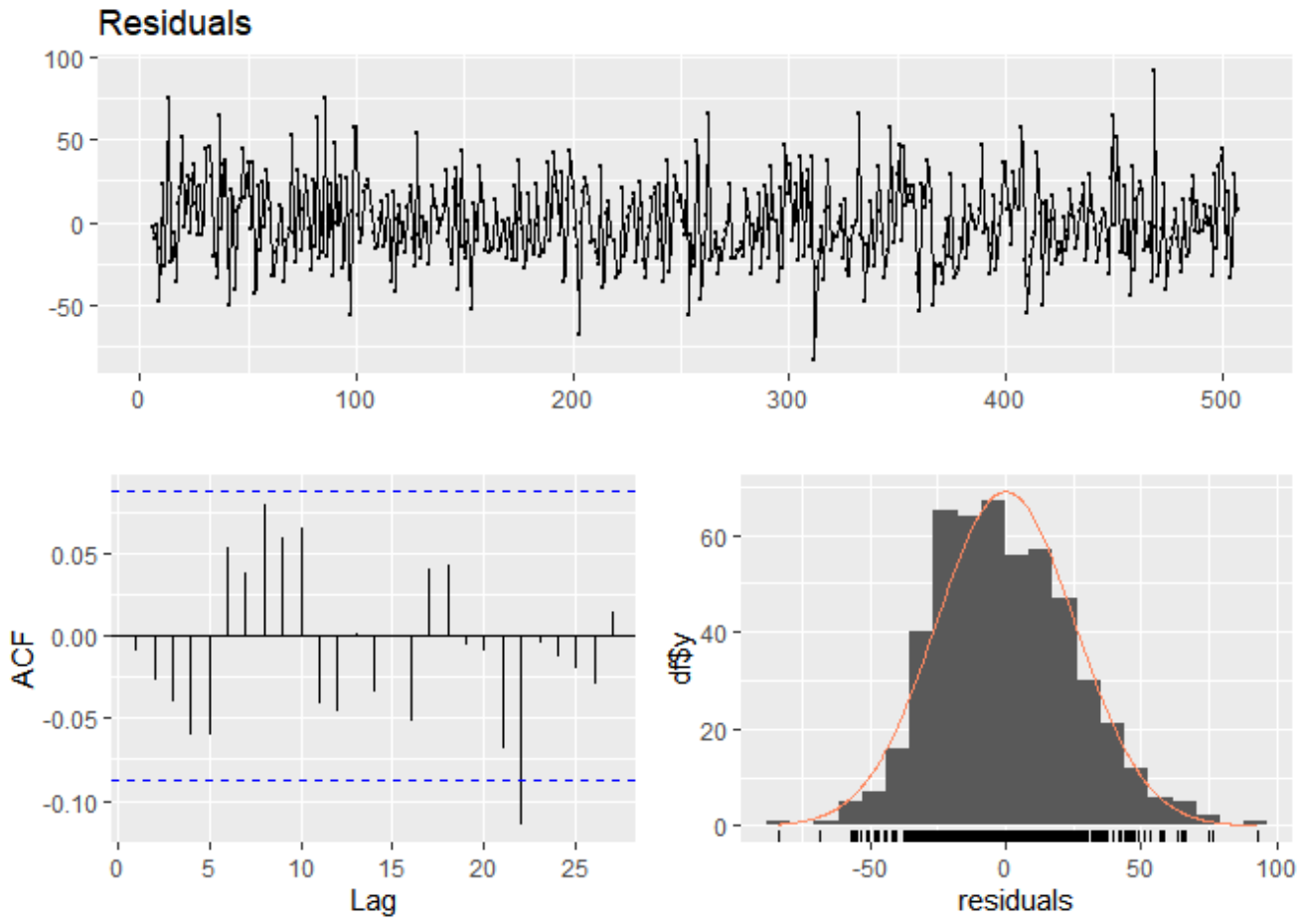

Figure 22

```
##
##  Ljung-Box test
```

```
##
## data:  Residuals
## Q* = 14.345, df = 10, p-value = 0.1578
##
## Model df: 0.   Total lags used: 10
```

```
bgtest(model4.4$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model4.4$model
## LM test = 4.9119, df = 1, p-value = 0.02667
```

```
shapiro.test(model4.4$model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model4.4$model$residuals
## W = 0.99254, p-value = 0.01302
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 22`. The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that there are no serial correlation in residuals.. Furthermore, from the Shapiro-Wilk test we can conclude that there is enough evidence to reject null hypothesis of normality as p-value < 0.05.

```
VIF.model4.4<-vif(model4.4$model)
VIF.model4.4
```

```
##       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)
##  2.776368  3.807219  4.013227  4.015255  3.822645  2.807692  1.367868  1.330746
## L(y.t, 3) L(y.t, 4) L(y.t, 5)
##  1.377369  1.341174  1.389780
```

```
VIF.model4.4>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
```

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the `vif` function, we can observe that there the data is not suffering from multicollinearity. The reason for this is because the data in the VIF are less than 10.

1.6.0 Dynamic Lag Models

Since there are intervention points as analyzed in the earlier sections, we will fit the dynamic linear models as the model is for assessing the effect of the intervention on the time series. We will compute the following dynamic lag models and choose the best one with the lowest mase value for further analysis.

```
Y.t=log(mort.ts)
T=156
S.t=1*(seq(Y.t) >= T)
S.t.1=Lag(S.t,+1)

model5.1<-dynlm(Y.t ~ L(Y.t , k = 1) + S.t + trend(Y.t) + season(Y.t))
model5.2<-dynlm(Y.t ~ L(Y.t , k = 2 ) + S.t + trend(Y.t) + season(Y.t))
model5.3<-dynlm(Y.t ~ L(Y.t , k = 1 ) + S.t + S.t.1 + trend(Y.t) + season(Y.t))
model5.4<-dynlm(Y.t ~ L(Y.t , k = 1 ) + S.t + season(Y.t))
model5.5<-dynlm(Y.t ~ L(Y.t , k = 1 ) + S.t + trend(Y.t))

model5.mase<-MASE(lm(model5.1),lm(model5.2),lm(model5.3),lm(model5.4),lm(model5.5))
model5.mase<-arrange(model5.mase)
model5.mase
```

```
##                n      MASE
## lm(model5.1) 507 0.8025763
## lm(model5.2) 506 0.8000460
## lm(model5.3) 507 0.8021698
## lm(model5.4) 507 0.8258294
## lm(model5.5) 507 0.9125745
```

From the output above we can see that model5.1 with trnd and seasonality contains the lowest mase value. Hence, we will use the model for further diagnostic checking.

```
summary(model5.1)
```

```
##
## 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
## -0.60597 -0.11483 -0.00756  0.12356  0.50786
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.3101816  0.1198662  10.930  < 2e-16 ***
## L(Y.t, k = 1)  0.4539374  0.0414884  10.941  < 2e-16 ***
## S.t           -0.1562487  0.0323025  -4.837 1.81e-06 ***
## trend(Y.t)     0.0222769  0.0052153   4.271 2.37e-05 ***
## season(Y.t)2  -0.0721005  0.0875179  -0.824 0.410467
## season(Y.t)3  -0.1471540  0.0875933  -1.680 0.093655 .
## season(Y.t)4  -0.0831864  0.0879098  -0.946 0.344516
## season(Y.t)5   0.0291965  0.0878596   0.332 0.739810
## season(Y.t)6  -0.1402922  0.0875624  -1.602 0.109811
## season(Y.t)7  -0.1856875  0.0878493  -2.114 0.035088 *
## season(Y.t)8  -0.1949754  0.0882953  -2.208 0.027730 *
## season(Y.t)9  -0.1492422  0.0886179  -1.684 0.092851 .
## season(Y.t)10 -0.1885968  0.0884826  -2.131 0.033590 *
## season(Y.t)11 -0.2642669  0.0886720  -2.980 0.003035 **
## season(Y.t)12 -0.1244557  0.0893411  -1.393 0.164293
## season(Y.t)13 -0.1831459  0.0886044  -2.067 0.039302 *
## season(Y.t)14 -0.2601853  0.0886941  -2.934 0.003522 **
## season(Y.t)15 -0.1554507  0.0893192  -1.740 0.082470 .
## season(Y.t)16 -0.3199919  0.0888061  -3.603 0.000349 ***
## season(Y.t)17 -0.2707110  0.0899172  -3.011 0.002753 **
## season(Y.t)18 -0.2893320  0.0900288  -3.214 0.001404 **
## season(Y.t)19 -0.2028102  0.0902663  -2.247 0.025135 *
## season(Y.t)20 -0.2356590  0.0895503  -2.632 0.008789 **
## season(Y.t)21 -0.3200813  0.0895375  -3.575 0.000388 ***
## season(Y.t)22 -0.1725658  0.0903340  -1.910 0.056726 .
## season(Y.t)23 -0.2452988  0.0893196  -2.746 0.006268 **
## season(Y.t)24 -0.2228024  0.0895155  -2.489 0.013170 *
## season(Y.t)25 -0.2531024  0.0894131  -2.831 0.004852 **
## season(Y.t)26 -0.2792180  0.0896299  -3.115 0.001955 **
## season(Y.t)27 -0.3075376  0.0899770  -3.418 0.000688 ***
## season(Y.t)28 -0.2947566  0.0904311  -3.259 0.001201 **
## season(Y.t)29 -0.2178343  0.0905136  -2.407 0.016500 *
## season(Y.t)30 -0.2576868  0.0897893  -2.870 0.004299 **
## season(Y.t)31 -0.2827229  0.0898513  -3.147 0.001761 **
## season(Y.t)32 -0.3399015  0.0901213  -3.772 0.000184 ***
## season(Y.t)33 -0.1269682  0.0908543  -1.397 0.162951
## season(Y.t)34 -0.1525367  0.0891423  -1.711 0.087738 .
## season(Y.t)35 -0.3085286  0.0887325  -3.477 0.000556 ***
## season(Y.t)36 -0.1895706  0.0897698  -2.112 0.035257 *
## season(Y.t)37 -0.2177013  0.0892574  -2.439 0.015111 *
## season(Y.t)38 -0.1912719  0.0892675  -2.143 0.032672 *
## season(Y.t)39 -0.2893001  0.0890667  -3.248 0.001248 **
## season(Y.t)40 -0.1745185  0.0897912  -1.944 0.052563 .
```

```
## season(Y.t)41 -0.2591092  0.0915249  -2.831 0.004847 **
## season(Y.t)42 -0.1564170  0.0918510  -1.703 0.089266 .
## season(Y.t)43 -0.1726189  0.0911793  -1.893 0.058972 .
## season(Y.t)44 -0.1507047  0.0910253  -1.656 0.098490 .
## season(Y.t)45  0.0008681  0.0908203   0.010 0.992378
## season(Y.t)46 -0.0953329  0.0900613  -1.059 0.290378
## season(Y.t)47  0.0636976  0.0901298   0.707 0.480097
## season(Y.t)48  0.0776647  0.0898364   0.865 0.387765
## season(Y.t)49  0.1056920  0.0898586   1.176 0.240133
## season(Y.t)50  0.0093049  0.0899689   0.103 0.917672
## season(Y.t)51 -0.0148336  0.0898624  -0.165 0.868963
## season(Y.t)52  0.0016555  0.0897838   0.018 0.985297
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1904 on 452 degrees of freedom
## Multiple R-squared:  0.5895, Adjusted R-squared:  0.5405
## F-statistic: 12.02 on 54 and 452 DF,  p-value: < 2.2e-16
```

From the summary output for `model5.1`, we can see the first lag is significant as p-value < 0.05. Furthermore, the adjusted R-squared shows a value of 0.5405, which accounts for 54.05% of the variability in the data.

```
checkresiduals(model5.1)
```

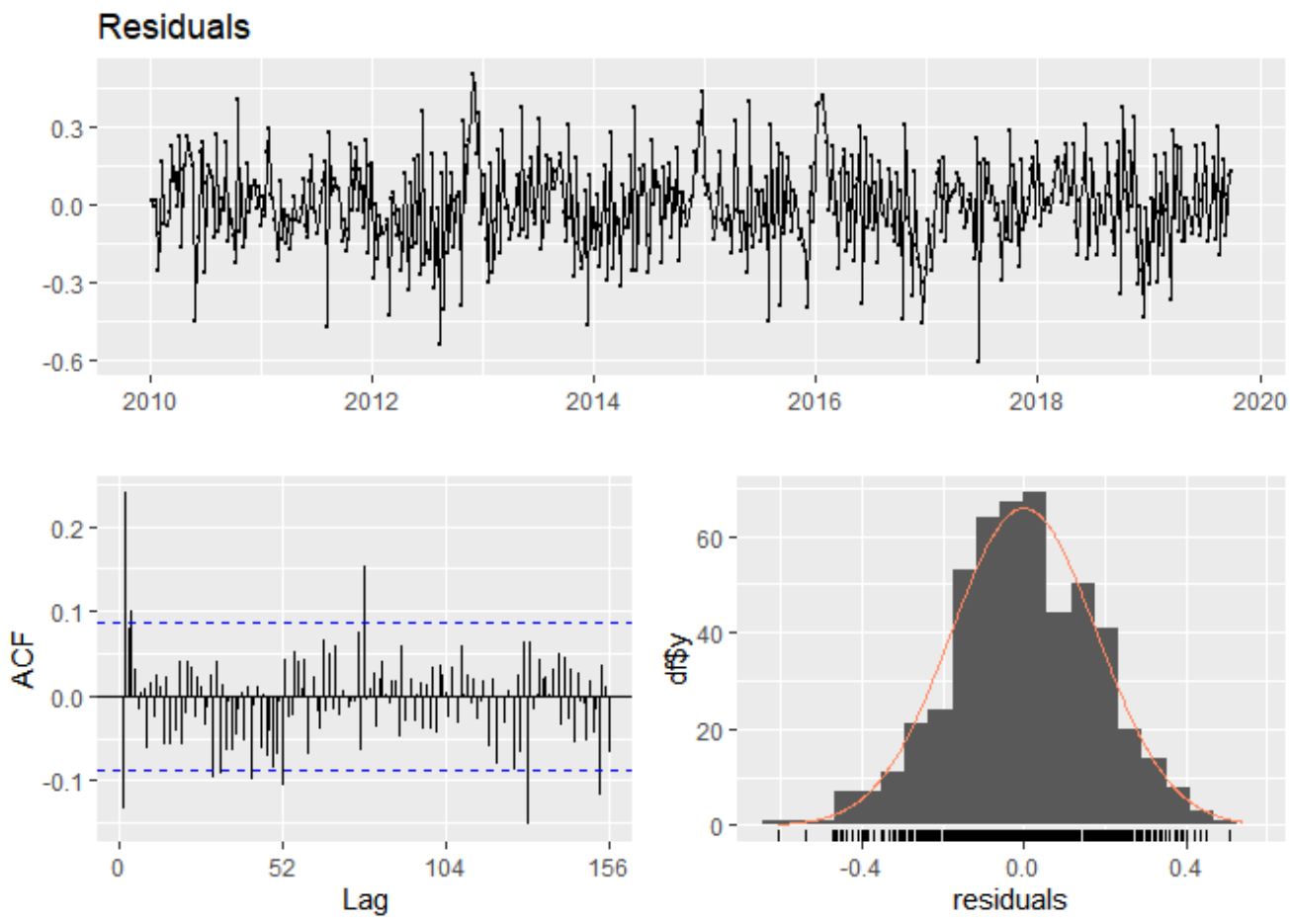

Figure 23

```
##
##  Breusch-Godfrey test for serial correlation of order up to 101
##
## data:  Residuals
## LM test = 147.59, df = 101, p-value = 0.001735
```

```
bgtest(model5.1$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model5.1$model
## LM test = 42.334, df = 1, p-value = 7.696e-11
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 23`. The Breush-Godfrey test is also conducted here using the `bgtest` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are highly significant.

```
VIF.model5.1<-vif(model5.1)
VIF.model5.1
```

```
##                   GVIF Df GVIF^(1/(2*Df))
## L(Y.t, k = 1) 1.900147  1        1.378458
## S.t           3.084695  1        1.756330
## trend(Y.t)    3.011939  1        1.735494
## season(Y.t)   1.834989 51        1.005969
```

```
VIF.model5.1>10
```

```
##                 GVIF    Df GVIF^(1/(2*Df))
## L(Y.t, k = 1) FALSE FALSE           FALSE
## S.t           FALSE FALSE           FALSE
## trend(Y.t)    FALSE FALSE           FALSE
## season(Y.t)   FALSE  TRUE           FALSE
```

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the `vif` function, we can observe that there the data is not suffering from multicollinearity. The reason for this is because the data in the VIF are less than 10.

1.7.0 Exponential Smoothing Methods

We will now test the time series data by fitting different trend and seasonality patterns. We will be fitting the models additive seasonality, additive seasonality with damped, multiplicative seasonality, and multiplicative seasonality with damped. We will use various Holt-Winter models using the function `holt`.

1.7.1 Additive Seasonality

```
model6.1<-holt(mort.ts,seasonal="additive",h=frequency(mort.ts))
summary(model6.1)
```

```
##
## Forecast method: Holt's method
##
## Model Information:
## Holt's method
##
## Call:
##  holt(y = mort.ts, h = frequency(mort.ts), seasonal = "additive")
##
##   Smoothing parameters:
##     alpha = 0.7377
##     beta  = 1e-04
##
##   Initial states:
##     l = 10.7622
##     b = -0.0049
##
##   sigma:  1.8466
##
##      AIC     AICc      BIC
## 3794.215 3794.335 3815.368
##
## Error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.004133748 1.839292 1.399971 -2.633318 16.90087 0.6645637
##                      ACF1
## Training set -0.02464511
##
## Forecasts:
##          Point Forecast       Lo 80    Hi 80      Lo 95    Hi 95
## 2019.769       9.744178  7.37769542 12.11066  6.1249550 13.36340
## 2019.788       9.739491  6.79861180 12.68037  5.2418044 14.23718
## 2019.808       9.734804  6.31455131 13.15506  4.5039791 14.96563
## 2019.827       9.730116  5.88976366 13.57047  3.8568038 15.60343
## 2019.846       9.725429  5.50650521 13.94435  3.2731420 16.17772
## 2019.865       9.720741  5.15443526 14.28705  2.7371788 16.70430
## 2019.885       9.716054  4.82690130 14.60521  2.2387402 17.19337
## 2019.904       9.711367  4.51932426 14.90341  1.7708231 17.65191
## 2019.923       9.706679  4.22839295 15.18497  1.3283634 18.08499
## 2019.942       9.701992  3.95162101 15.45236  0.9075586 18.49642
```

```
## 2019.962       9.697304  3.68708514 15.70752  0.5054673 18.88914
## 2019.981       9.692617  3.43326125 15.95197  0.1197585 19.26548
## 2020.000       9.687930  3.18891730 16.18694 -0.2514520 19.62731
## 2020.019       9.683242  2.95304049 16.41344 -0.6097130 19.97620
## 2020.038       9.678555  2.72478624 16.63232 -0.9563164 20.31343
## 2020.058       9.673867  2.50344153 16.84429 -1.2923526 20.64009
## 2020.077       9.669180  2.28839784 17.04996 -1.6187522 20.95711
## 2020.096       9.664492  2.07913094 17.24985 -1.9363169 21.26530
## 2020.115       9.659805  1.87518536 17.44442 -2.2457434 21.56535
## 2020.135       9.655118  1.67616242 17.63407 -2.5476414 21.85788
## 2020.154       9.650430  1.48171076 17.81915 -2.8425481 22.14341
## 2020.173       9.645743  1.29151888 17.99997 -3.1309401 22.42243
## 2020.192       9.641055  1.10530903 18.17680 -3.4132422 22.69535
## 2020.212       9.636368  0.92283234 18.34990 -3.6898348 22.96257
## 2020.231       9.631681  0.74386477 18.51950 -3.9610607 23.22442
## 2020.250       9.626993  0.56820377 18.68578 -4.2272297 23.48122
## 2020.269       9.622306  0.39566553 18.84895 -4.4886228 23.73323
## 2020.288       9.617618  0.22608259 19.00915 -4.7454962 23.98073
## 2020.308       9.612931  0.05930193 19.16656 -4.9980838 24.22395
## 2020.327       9.608244 -0.10481676 19.32130 -5.2466003 24.46309
## 2020.346       9.603556 -0.26640248 19.47351 -5.4912430 24.69836
## 2020.365       9.598869 -0.42557417 19.62331 -5.7321937 24.92993
## 2020.385       9.594181 -0.58244177 19.77080 -5.9696206 25.15798
## 2020.404       9.589494 -0.73710716 19.91610 -6.2036796 25.38267
## 2020.423       9.584807 -0.88966493 20.05928 -6.4345152 25.60413
## 2020.442       9.580119 -1.04020313 20.20044 -6.6622621 25.82250
## 2020.462       9.575432 -1.18880386 20.33967 -6.8870460 26.03791
## 2020.481       9.570744 -1.33554381 20.47703 -7.1089840 26.25047
## 2020.500       9.566057 -1.48049479 20.61261 -7.3281860 26.46030
## 2020.519       9.561370 -1.62372409 20.74646 -7.5447550 26.66749
## 2020.538       9.556682 -1.76529493 20.87866 -7.7587875 26.87215
## 2020.558       9.551995 -1.90526675 21.00926 -7.9703746 27.07436
## 2020.577       9.547307 -2.04369555 21.13831 -8.1796018 27.27422
## 2020.596       9.542620 -2.18063414 21.26587 -8.3865499 27.47179
## 2020.615       9.537932 -2.31613240 21.39200 -8.5912953 27.66716
## 2020.635       9.533245 -2.45023750 21.51673 -8.7939100 27.86040
## 2020.654       9.528558 -2.58299412 21.64011 -8.9944623 28.05158
## 2020.673       9.523870 -2.71444458 21.76219 -9.1930171 28.24076
## 2020.692       9.519183 -2.84462907 21.88299 -9.3896357 28.42800
## 2020.712       9.514495 -2.97358576 22.00258 -9.5843766 28.61337
## 2020.731       9.509808 -3.10135094 22.12097 -9.7772952 28.79691
## 2020.750       9.505121 -3.22795918 22.23820 -9.9684445 28.97869
```

The output here is the model with additive seasonality. We can see here that this model has a MASE value of 0.6645637 and AIC score of 3794.215.

```
checkresiduals(model6.1)
```

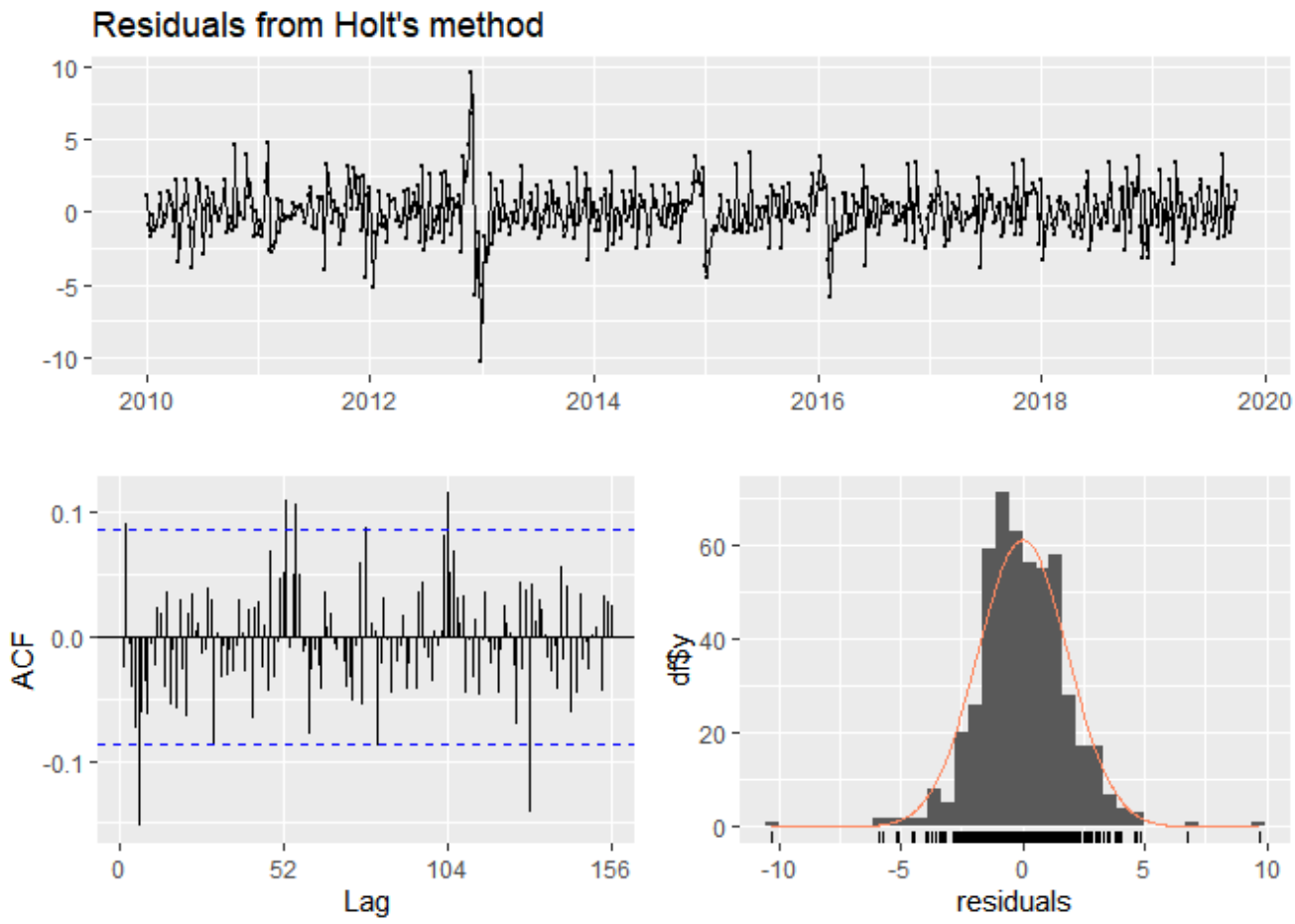

Figure 24

```
##
##  Ljung-Box test
##
## data:  Residuals from Holt's method
## Q* = 102.05, df = 102, p-value = 0.48
##
## Model df: 0.   Total lags used: 102
```

`Figure 24` above shows the residual plot for `model6.1`. Here is what we can observe from the output:

- Based on the p-values, we cannot reject the hypothesis of stationary series as the p-value is greater that 0.05.
- There are significant lags observed at the ACF plot.

1.7.2 Additive Seasonality Damped

```
model6.2<-holt(mort.ts,seasonal="additive",damped=TRUE,h=frequency(mort.ts))
summary(model6.2)
```

```
##
## Forecast method: Damped Holt's method
##
```

```
## Model Information:
## Damped Holt's method
##
## Call:
##  holt(y = mort.ts, h = frequency(mort.ts), damped = TRUE, seasonal = "additive")
##
##   Smoothing parameters:
##     alpha = 0.7343
##     beta  = 1e-04
##     phi   = 0.8058
##
##   Initial states:
##     l = 11.6518
##     b = -0.6308
##
##   sigma:  1.8466
##
##      AIC     AICc      BIC
## 3795.223 3795.391 3820.606
##
## Error measures:
##                       ME     RMSE      MAE       MPE    MAPE      MASE
## Training set 0.001909679 1.837497 1.396362 -2.670503 16.8666 0.6628509
##                      ACF1
## Training set -0.02200273
##
## Forecasts:
##          Point Forecast       Lo 80    Hi 80      Lo 95    Hi 95
## 2019.769       9.745906  7.37938414 12.11243  6.1266229 13.36519
## 2019.788       9.746016  6.80985737 12.68217  5.2555490 14.23648
## 2019.808       9.746104  6.33404256 13.15816  4.5278065 14.96440
## 2019.827       9.746175  5.91684959 13.57550  3.8897273 15.60262
## 2019.846       9.746232  5.54080093 13.95166  3.3145802 16.17788
## 2019.865       9.746278  5.19569012 14.29687  2.7867544 16.70580
## 2019.885       9.746315  4.87494190 14.61769  2.2961927 17.19644
## 2019.904       9.746345  4.57402574 14.91866  1.8359653 17.65672
## 2019.923       9.746369  4.28966280 15.20308  1.4010570 18.09168
## 2019.942       9.746389  4.01938935 15.47339  0.9876991 18.50508
## 2019.962       9.746404  3.76129841 15.73151  0.5929748 18.89983
## 2019.981       9.746417  3.51387800 15.97896  0.2145712 19.27826
## 2020.000       9.746427  3.27590529 16.21695 -0.1493820 19.64224
## 2020.019       9.746435  3.04637458 16.44650 -0.5004233 19.99329
## 2020.038       9.746442  2.82444689 16.66844 -0.8398359 20.33272
## 2020.058       9.746447  2.60941360 16.88348 -1.1687037 20.66160
## 2020.077       9.746451  2.40066979 17.09223 -1.4879521 20.98085
## 2020.096       9.746455  2.19769412 17.29522 -1.7983784 21.29129
## 2020.115       9.746458  2.00003351 17.49288 -2.1006757 21.59359
## 2020.135       9.746460  1.80729124 17.68563 -2.3954508 21.88837
## 2020.154       9.746462  1.61911762 17.87381 -2.6832385 22.17616
## 2020.173       9.746463  1.43520250 18.05772 -2.9645131 22.45744
## 2020.192       9.746464  1.25526931 18.23766 -3.2396978 22.73263
## 2020.212       9.746465  1.07907019 18.41386 -3.5091716 23.00210
```

```
## 2020.231       9.746466  0.90638192 18.58655 -3.7732759 23.26621
## 2020.250       9.746467  0.73700272 18.75593 -4.0323194 23.52525
## 2020.269       9.746467  0.57074937 18.92218 -4.2865822 23.77952
## 2020.288       9.746468  0.40745500 19.08548 -4.5363196 24.02925
## 2020.308       9.746468  0.24696703 19.24597 -4.7817650 24.27470
## 2020.327       9.746468  0.08914560 19.40379 -5.0231322 24.51607
## 2020.346       9.746468 -0.06613794 19.55907 -5.2606179 24.75355
## 2020.365       9.746469 -0.21900221 19.71194 -5.4944038 24.98734
## 2020.385       9.746469 -0.36955686 19.86249 -5.7246573 25.21759
## 2020.404       9.746469 -0.51790352 20.01084 -5.9515340 25.44447
## 2020.423       9.746469 -0.66413656 20.15707 -6.1751782 25.66812
## 2020.442       9.746469 -0.80834382 20.30128 -6.3957242 25.88866
## 2020.462       9.746469 -0.95060723 20.44355 -6.6132973 26.10624
## 2020.481       9.746469 -1.09100333 20.58394 -6.8280147 26.32095
## 2020.500       9.746469 -1.22960378 20.72254 -7.0399858 26.53292
## 2020.519       9.746469 -1.36647575 20.85941 -7.2493134 26.74225
## 2020.538       9.746469 -1.50168235 20.99462 -7.4560941 26.94903
## 2020.558       9.746469 -1.63528292 21.12822 -7.6604185 27.15336
## 2020.577       9.746469 -1.76733336 21.26027 -7.8623722 27.35531
## 2020.596       9.746469 -1.89788641 21.39082 -8.0620359 27.55497
## 2020.615       9.746469 -2.02699189 21.51993 -8.2594857 27.75242
## 2020.635       9.746469 -2.15469690 21.64764 -8.4547937 27.94773
## 2020.654       9.746469 -2.28104605 21.77398 -8.6480280 28.14097
## 2020.673       9.746469 -2.40608163 21.89902 -8.8392534 28.33219
## 2020.692       9.746469 -2.52984378 22.02278 -9.0285313 28.52147
## 2020.712       9.746469 -2.65237062 22.14531 -9.2159200 28.70886
## 2020.731       9.746469 -2.77369844 22.26664 -9.4014748 28.89441
## 2020.750       9.746469 -2.89386175 22.38680 -9.5852488 29.07819
```

The output here is the model with additive seasonality and damped. We can see here that this model has a MASE value of 0.6628509 and AIC score of 3795.223.

```
checkresiduals(model6.2)
```

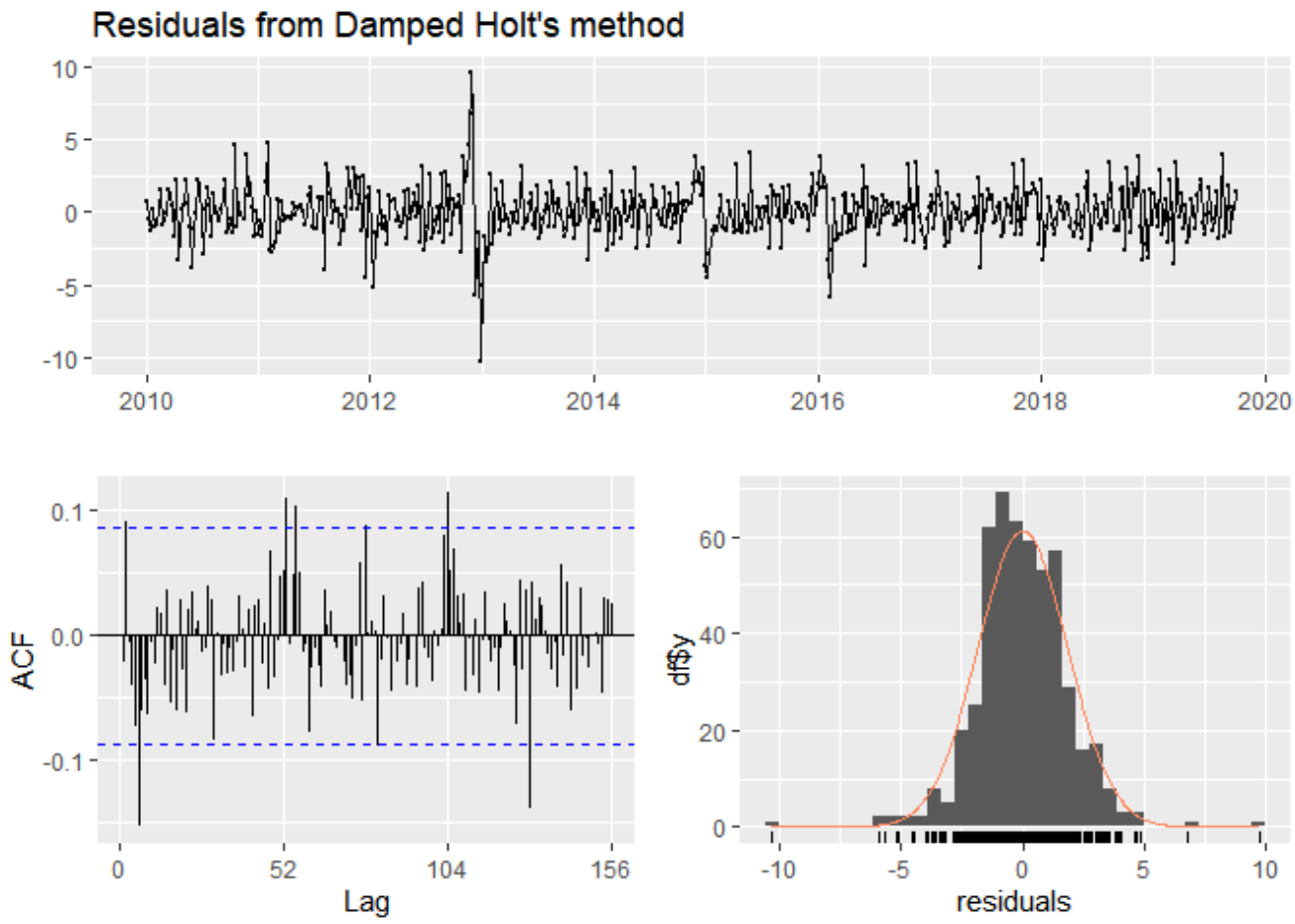

Figure 25

```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt's method
## Q* = 101.89, df = 102, p-value = 0.4845
##
## Model df: 0.   Total lags used: 102
```

`Figure 25` above shows the residual plot for `model6.2`. Here is what we can observe from the output:

- Based on the p-values, we cannot reject the hypothesis of stationary series as the p-value is greater that 0.05.
- There are significant lags observed at the ACF plot.

1.7.3 Multiplicative Seasonality

```
model6.3<-holt(mort.ts,seasonal="multiplicative",h=frequency(mort.ts))
summary(model6.3)
```

```
##
## Forecast method: Holt's method
##
```

```
## Model Information:
## Holt's method
##
## Call:
##  holt(y = mort.ts, h = frequency(mort.ts), seasonal = "multiplicative")
##
##   Smoothing parameters:
##     alpha = 0.7377
##     beta  = 1e-04
##
##   Initial states:
##     l = 10.7622
##     b = -0.0049
##
##   sigma:  1.8466
##
##      AIC     AICc      BIC
## 3794.215 3794.335 3815.368
##
## Error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.004133748 1.839292 1.399971 -2.633318 16.90087 0.6645637
##                      ACF1
## Training set -0.02464511
##
## Forecasts:
##          Point Forecast       Lo 80    Hi 80      Lo 95    Hi 95
## 2019.769       9.744178  7.37769542 12.11066  6.1249550 13.36340
## 2019.788       9.739491  6.79861180 12.68037  5.2418044 14.23718
## 2019.808       9.734804  6.31455131 13.15506  4.5039791 14.96563
## 2019.827       9.730116  5.88976366 13.57047  3.8568038 15.60343
## 2019.846       9.725429  5.50650521 13.94435  3.2731420 16.17772
## 2019.865       9.720741  5.15443526 14.28705  2.7371788 16.70430
## 2019.885       9.716054  4.82690130 14.60521  2.2387402 17.19337
## 2019.904       9.711367  4.51932426 14.90341  1.7708231 17.65191
## 2019.923       9.706679  4.22839295 15.18497  1.3283634 18.08499
## 2019.942       9.701992  3.95162101 15.45236  0.9075586 18.49642
## 2019.962       9.697304  3.68708514 15.70752  0.5054673 18.88914
## 2019.981       9.692617  3.43326125 15.95197  0.1197585 19.26548
## 2020.000       9.687930  3.18891730 16.18694 -0.2514520 19.62731
## 2020.019       9.683242  2.95304049 16.41344 -0.6097130 19.97620
## 2020.038       9.678555  2.72478624 16.63232 -0.9563164 20.31343
## 2020.058       9.673867  2.50344153 16.84429 -1.2923526 20.64009
## 2020.077       9.669180  2.28839784 17.04996 -1.6187522 20.95711
## 2020.096       9.664492  2.07913094 17.24985 -1.9363169 21.26530
## 2020.115       9.659805  1.87518536 17.44442 -2.2457434 21.56535
## 2020.135       9.655118  1.67616242 17.63407 -2.5476414 21.85788
## 2020.154       9.650430  1.48171076 17.81915 -2.8425481 22.14341
## 2020.173       9.645743  1.29151888 17.99997 -3.1309401 22.42243
## 2020.192       9.641055  1.10530903 18.17680 -3.4132422 22.69535
## 2020.212       9.636368  0.92283234 18.34990 -3.6898348 22.96257
## 2020.231       9.631681  0.74386477 18.51950 -3.9610607 23.22442
```

```
## 2020.250       9.626993  0.56820377 18.68578 -4.2272297 23.48122
## 2020.269       9.622306  0.39566553 18.84895 -4.4886228 23.73323
## 2020.288       9.617618  0.22608259 19.00915 -4.7454962 23.98073
## 2020.308       9.612931  0.05930193 19.16656 -4.9980838 24.22395
## 2020.327       9.608244 -0.10481676 19.32130 -5.2466003 24.46309
## 2020.346       9.603556 -0.26640248 19.47351 -5.4912430 24.69836
## 2020.365       9.598869 -0.42557417 19.62331 -5.7321937 24.92993
## 2020.385       9.594181 -0.58244177 19.77080 -5.9696206 25.15798
## 2020.404       9.589494 -0.73710716 19.91610 -6.2036796 25.38267
## 2020.423       9.584807 -0.88966493 20.05928 -6.4345152 25.60413
## 2020.442       9.580119 -1.04020313 20.20044 -6.6622621 25.82250
## 2020.462       9.575432 -1.18880386 20.33967 -6.8870460 26.03791
## 2020.481       9.570744 -1.33554381 20.47703 -7.1089840 26.25047
## 2020.500       9.566057 -1.48049479 20.61261 -7.3281860 26.46030
## 2020.519       9.561370 -1.62372409 20.74646 -7.5447550 26.66749
## 2020.538       9.556682 -1.76529493 20.87866 -7.7587875 26.87215
## 2020.558       9.551995 -1.90526675 21.00926 -7.9703746 27.07436
## 2020.577       9.547307 -2.04369555 21.13831 -8.1796018 27.27422
## 2020.596       9.542620 -2.18063414 21.26587 -8.3865499 27.47179
## 2020.615       9.537932 -2.31613240 21.39200 -8.5912953 27.66716
## 2020.635       9.533245 -2.45023750 21.51673 -8.7939100 27.86040
## 2020.654       9.528558 -2.58299412 21.64011 -8.9944623 28.05158
## 2020.673       9.523870 -2.71444458 21.76219 -9.1930171 28.24076
## 2020.692       9.519183 -2.84462907 21.88299 -9.3896357 28.42800
## 2020.712       9.514495 -2.97358576 22.00258 -9.5843766 28.61337
## 2020.731       9.509808 -3.10135094 22.12097 -9.7772952 28.79691
## 2020.750       9.505121 -3.22795918 22.23820 -9.9684445 28.97869
```

The output here is the model with multiplicative seasonality. We can see here that this model has a MASE value of 0.6645637 and AIC score of 3794.215.

```
checkresiduals(model6.3)
```

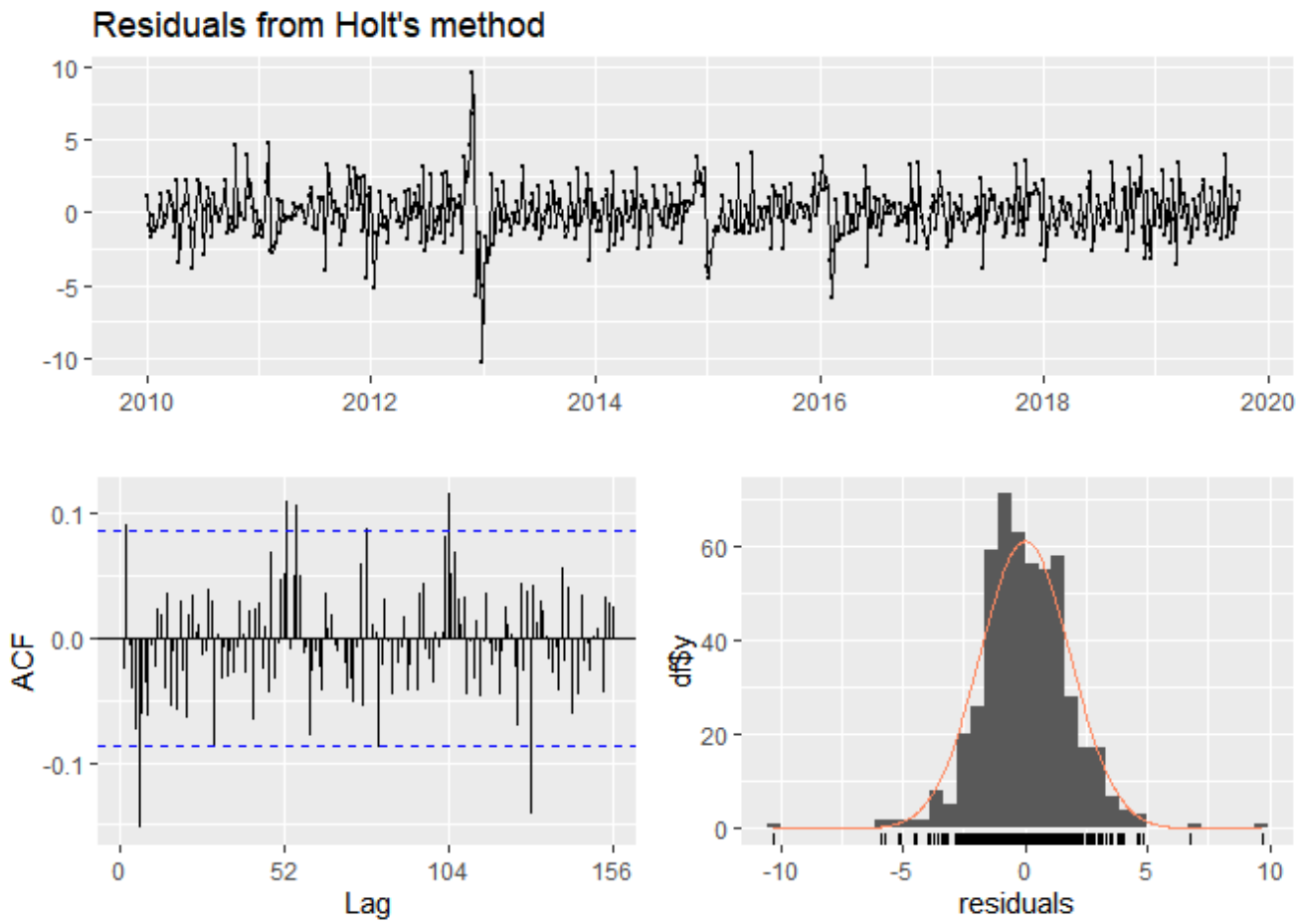

Figure 26

```
##
##  Ljung-Box test
##
## data:  Residuals from Holt's method
## Q* = 102.05, df = 102, p-value = 0.48
##
## Model df: 0.   Total lags used: 102
```

`Figure 26` above shows the residual plot for `model6.3`. Here is what we can observe from the output:

- Based on the p-values, we cannot reject the hypothesis of stationary series as the p-value is greater that 0.05.
- There are significant lags observed at the ACF plot.

1.7.4 Multiplicative Seasonality Damped

```
model6.4<-holt(mort.ts,seasonal="multiplicative",damped=TRUE,h=frequency(mort.ts))
summary(model6.4)
```

```
##
## Forecast method: Damped Holt's method
##
```

```
## Model Information:
## Damped Holt's method
##
## Call:
##  holt(y = mort.ts, h = frequency(mort.ts), damped = TRUE, seasonal = "multiplicative")
##
##   Smoothing parameters:
##     alpha = 0.7343
##     beta  = 1e-04
##     phi   = 0.8058
##
##   Initial states:
##     l = 11.6518
##     b = -0.6308
##
##   sigma:  1.8466
##
##      AIC     AICc      BIC
## 3795.223 3795.391 3820.606
##
## Error measures:
##                       ME     RMSE      MAE       MPE    MAPE      MASE
## Training set 0.001909679 1.837497 1.396362 -2.670503 16.8666 0.6628509
##                      ACF1
## Training set -0.02200273
##
## Forecasts:
##          Point Forecast       Lo 80    Hi 80      Lo 95    Hi 95
## 2019.769       9.745906  7.37938414 12.11243  6.1266229 13.36519
## 2019.788       9.746016  6.80985737 12.68217  5.2555490 14.23648
## 2019.808       9.746104  6.33404256 13.15816  4.5278065 14.96440
## 2019.827       9.746175  5.91684959 13.57550  3.8897273 15.60262
## 2019.846       9.746232  5.54080093 13.95166  3.3145802 16.17788
## 2019.865       9.746278  5.19569012 14.29687  2.7867544 16.70580
## 2019.885       9.746315  4.87494190 14.61769  2.2961927 17.19644
## 2019.904       9.746345  4.57402574 14.91866  1.8359653 17.65672
## 2019.923       9.746369  4.28966280 15.20308  1.4010570 18.09168
## 2019.942       9.746389  4.01938935 15.47339  0.9876991 18.50508
## 2019.962       9.746404  3.76129841 15.73151  0.5929748 18.89983
## 2019.981       9.746417  3.51387800 15.97896  0.2145712 19.27826
## 2020.000       9.746427  3.27590529 16.21695 -0.1493820 19.64224
## 2020.019       9.746435  3.04637458 16.44650 -0.5004233 19.99329
## 2020.038       9.746442  2.82444689 16.66844 -0.8398359 20.33272
## 2020.058       9.746447  2.60941360 16.88348 -1.1687037 20.66160
## 2020.077       9.746451  2.40066979 17.09223 -1.4879521 20.98085
## 2020.096       9.746455  2.19769412 17.29522 -1.7983784 21.29129
## 2020.115       9.746458  2.00003351 17.49288 -2.1006757 21.59359
## 2020.135       9.746460  1.80729124 17.68563 -2.3954508 21.88837
## 2020.154       9.746462  1.61911762 17.87381 -2.6832385 22.17616
## 2020.173       9.746463  1.43520250 18.05772 -2.9645131 22.45744
## 2020.192       9.746464  1.25526931 18.23766 -3.2396978 22.73263
## 2020.212       9.746465  1.07907019 18.41386 -3.5091716 23.00210
```

```
## 2020.231       9.746466  0.90638192 18.58655 -3.7732759 23.26621
## 2020.250       9.746467  0.73700272 18.75593 -4.0323194 23.52525
## 2020.269       9.746467  0.57074937 18.92218 -4.2865822 23.77952
## 2020.288       9.746468  0.40745500 19.08548 -4.5363196 24.02925
## 2020.308       9.746468  0.24696703 19.24597 -4.7817650 24.27470
## 2020.327       9.746468  0.08914560 19.40379 -5.0231322 24.51607
## 2020.346       9.746468 -0.06613794 19.55907 -5.2606179 24.75355
## 2020.365       9.746469 -0.21900221 19.71194 -5.4944038 24.98734
## 2020.385       9.746469 -0.36955686 19.86249 -5.7246573 25.21759
## 2020.404       9.746469 -0.51790352 20.01084 -5.9515340 25.44447
## 2020.423       9.746469 -0.66413656 20.15707 -6.1751782 25.66812
## 2020.442       9.746469 -0.80834382 20.30128 -6.3957242 25.88866
## 2020.462       9.746469 -0.95060723 20.44355 -6.6132973 26.10624
## 2020.481       9.746469 -1.09100333 20.58394 -6.8280147 26.32095
## 2020.500       9.746469 -1.22960378 20.72254 -7.0399858 26.53292
## 2020.519       9.746469 -1.36647575 20.85941 -7.2493134 26.74225
## 2020.538       9.746469 -1.50168235 20.99462 -7.4560941 26.94903
## 2020.558       9.746469 -1.63528292 21.12822 -7.6604185 27.15336
## 2020.577       9.746469 -1.76733336 21.26027 -7.8623722 27.35531
## 2020.596       9.746469 -1.89788641 21.39082 -8.0620359 27.55497
## 2020.615       9.746469 -2.02699189 21.51993 -8.2594857 27.75242
## 2020.635       9.746469 -2.15469690 21.64764 -8.4547937 27.94773
## 2020.654       9.746469 -2.28104605 21.77398 -8.6480280 28.14097
## 2020.673       9.746469 -2.40608163 21.89902 -8.8392534 28.33219
## 2020.692       9.746469 -2.52984378 22.02278 -9.0285313 28.52147
## 2020.712       9.746469 -2.65237062 22.14531 -9.2159200 28.70886
## 2020.731       9.746469 -2.77369844 22.26664 -9.4014748 28.89441
## 2020.750       9.746469 -2.89386175 22.38680 -9.5852488 29.07819
```

The output here is the model with multiplicative seasonality and damped. We can see here that this model has a MASE value of 0.6628509 and AIC score of 3795.223.

```
checkresiduals(model6.4)
```

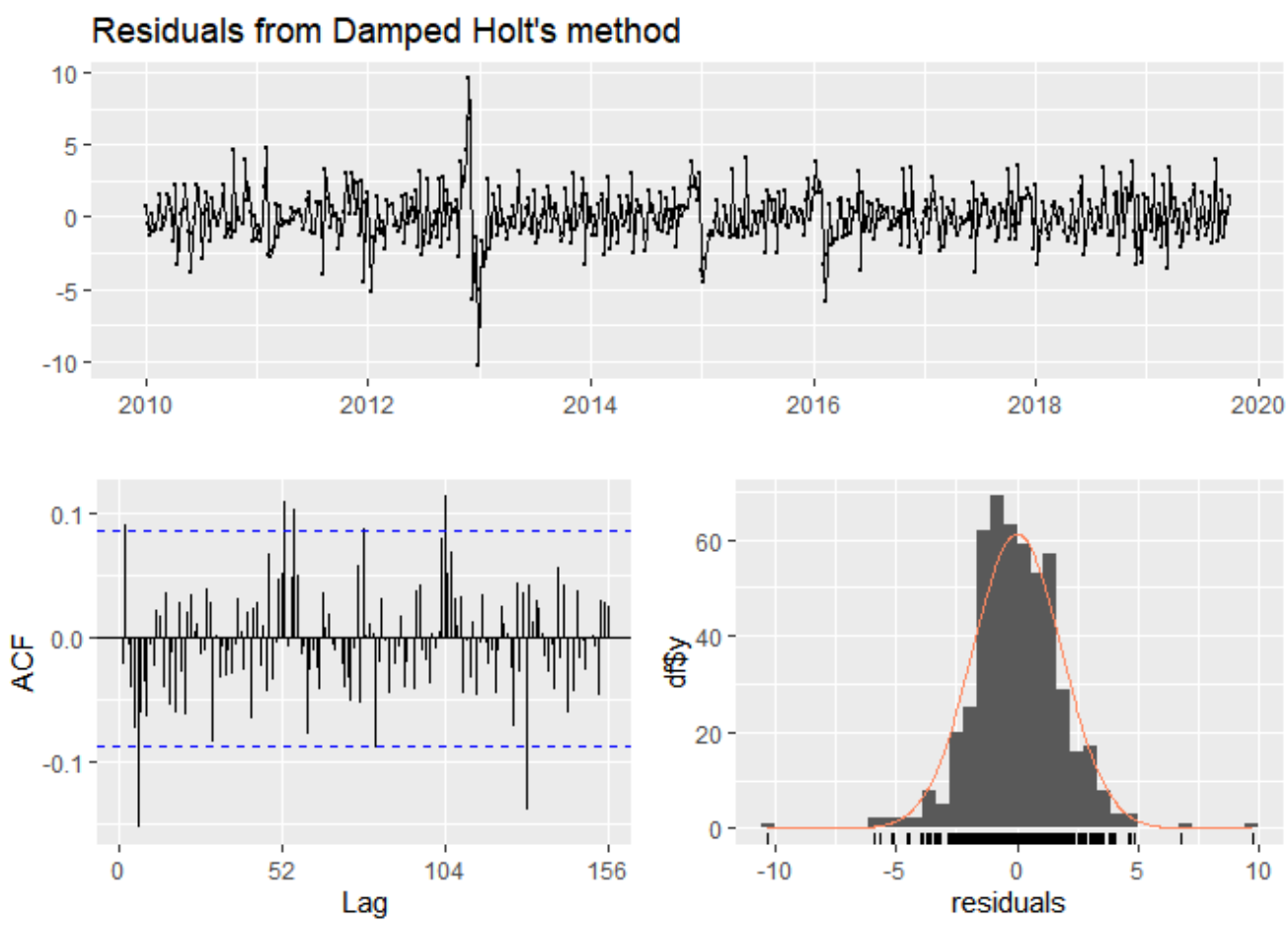

Figure 27

```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt's method
## Q* = 101.89, df = 102, p-value = 0.4845
##
## Model df: 0.   Total lags used: 102
```

`Figure 27` above shows the residual plot for `model6.4`. Here is what we can observe from the output:

- Based on the p-values, we cannot reject the hypothesis of stationary series as the p-value is greater that 0.05.
- There are significant lags observed at the ACF plot.

1.8.0 State-Space Models

We will now use state-space models with seasonality components to fit the series. We will use the function `ets` to fit state-space models corresponding to the exponential smoothing methods.

1.8.1 A,A,N

```
model7.1<-ets(mort.ts,model="AAN")
summary(model7.1)
```

```
## ETS(A,A,N)
##
## Call:
##  ets(y = mort.ts, model = "AAN")
##
##   Smoothing parameters:
##     alpha = 0.7377
##     beta  = 1e-04
##
##   Initial states:
##     l = 10.7616
##     b = -0.0047
##
##   sigma:  1.8466
##
##      AIC     AICc      BIC
## 3794.216 3794.335 3815.368
##
## Training set error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.003842306 1.839292 1.399981 -2.637078 16.90144 0.6645686
##                      ACF1
## Training set -0.02465578
```

The first model shows the summary output for `model7.1` which is a additive error model. Here we can observe that this model has a MASE value of 0.6645686 and AIC score of 3794.216.

```
checkresiduals(model7.1)
```

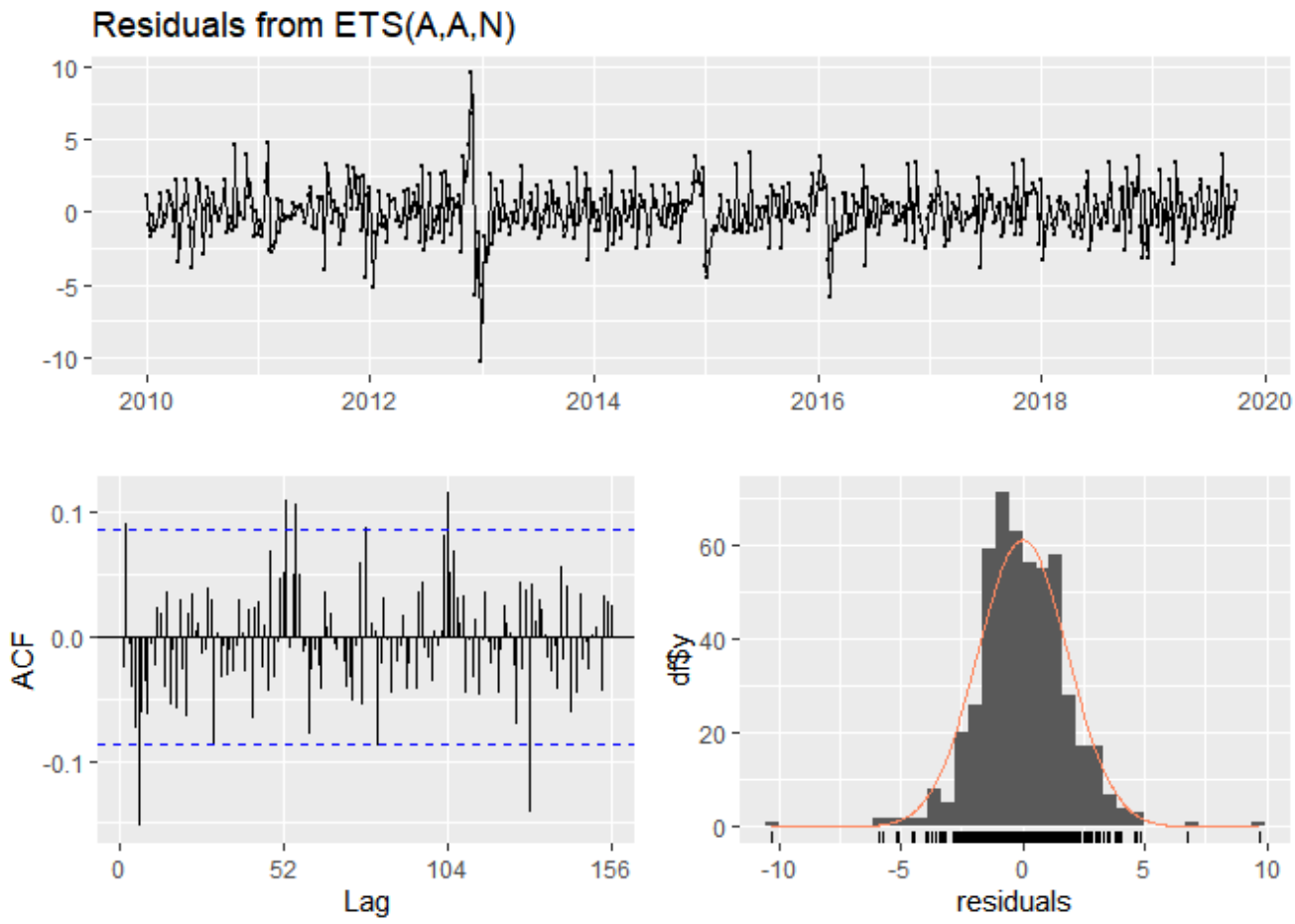

Figure 28

```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,A,N)
## Q* = 102.05, df = 102, p-value = 0.48
##
## Model df: 0.   Total lags used: 102
```

`Figure 28` above shows the residual plot for `model7.1`. Here is what we can observe from the output:

- Based on the p-values, we cannot reject the hypothesis of stationary series as the p-value is greater that 0.05.
- There are significant lags observed at the ACF plot.

1.8.2 M,A,N

```
model7.2<-ets(mort.ts,model="MAN")
summary(model7.2)
```

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

```
##  ets(y = mort.ts, model = "MAN")
##
##   Smoothing parameters:
##     alpha = 0.5396
##     beta  = 1e-04
##
##   Initial states:
##     l = 11.398
##     b = 0.0882
##
##   sigma:  0.2
##
##      AIC     AICc      BIC
## 3682.020 3682.139 3703.172
##
## Training set error measures:
##                       ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.1618487 1.893448 1.404443 -5.065489 17.01295 0.6666867
##                    ACF1
## Training set 0.1998776
```

The first model shows the summary output for `model7.2` which is a multiplicative error model. Here we can observe that this model has a MASE value of 0.6666867 and AIC score of 3682.020.

```
checkresiduals(model7.2)
```

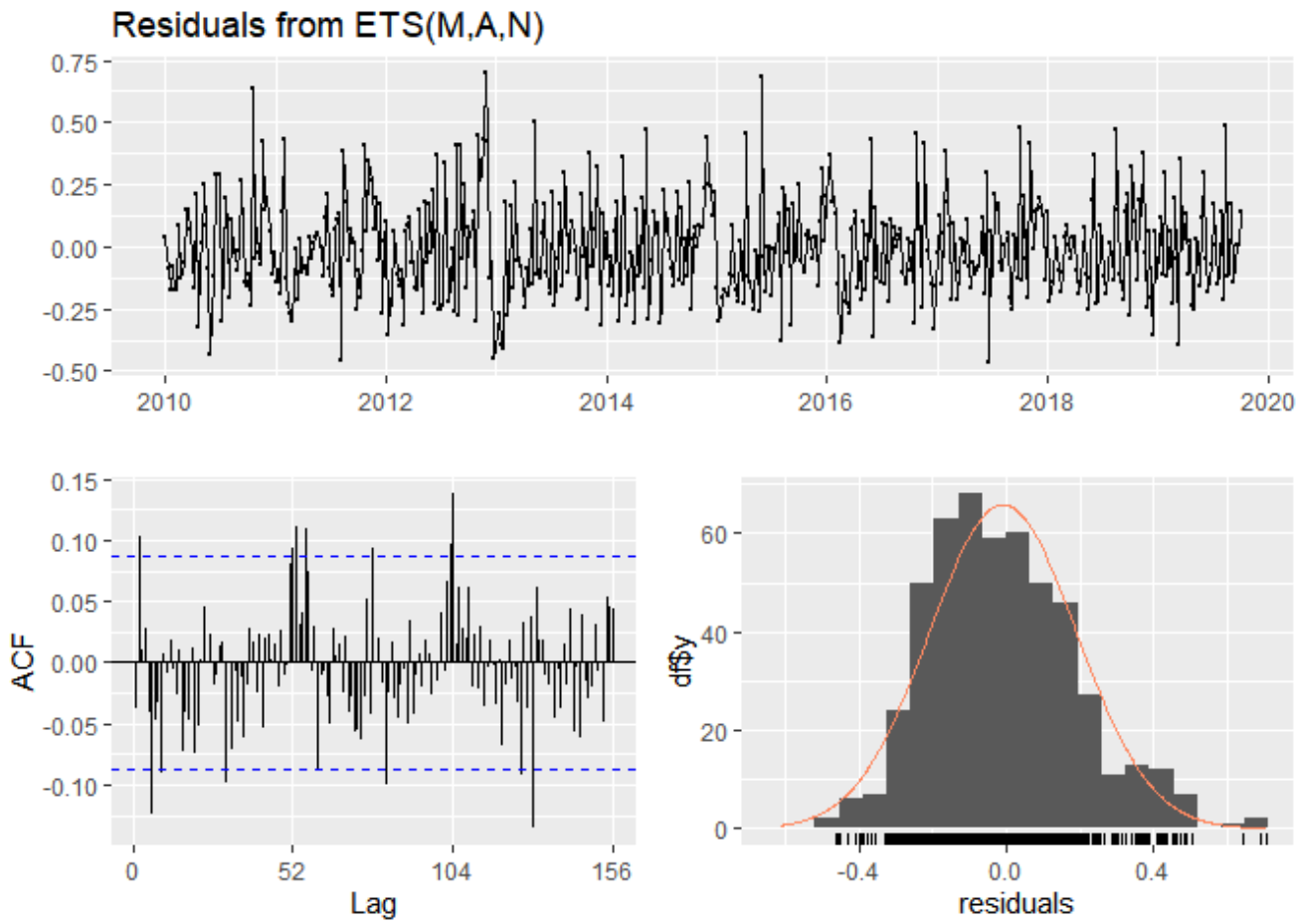

Figure 29

```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,A,N)
## Q* = 117.92, df = 102, p-value = 0.1341
##
## Model df: 0.   Total lags used: 102
```

`Figure 29` above shows the residual plot for `model7.2`. Here is what we can observe from the output:

- Based on the p-values, we cannot reject the hypothesis of stationary series as the p-value is greater that 0.05.
- There are significant lags observed at the ACF plot.

1.8.3 Auto ETS

We will now use an auto ETS model to observe what the recommended model is.

```
autoets<-ets(mort.ts)
autoets$method
```

```
## [1] "ETS(M,A,N)"
```

From the auto ETS model output above we are recommended to use the model M,A,N. This is a multiplicative error model with additive trend and no seasonality.

1.9.0 Model Comparison and Selection

Now, we will compare all models including the time series regression models, exponential smoothing method models, and state-space models with their respective MASE, AIC, and BIC values.

```
model.tsrm<-c("Finite DLM","Polynomial DLM","Koyck DLM","Autoregressive DLM (P4Q5)",
              "Autoregressive DLM (P1Q5)","Autoregressive DLM (P5Q5)")
mase.tsrm<-MASE(model1.2,model2.1,model3.1,model4.2,model4.3,model4.4)
tsrm<-data.frame(model.tsrm,mase.tsrm)
tsrm<-tsrm[-2]
colnames(tsrm)<-c("Model","MASE")

model.ss<-c("Dynamic Lag Models","HW Additive","HW Additive Damped",
            "HW Multiplicative","HW Multiplicative Damped",
            "ETS (A,A,N)","ETS(M,A,N)")
mase.ss<-c(accuracy(model5.1)[,6],accuracy(model6.1)[,6],accuracy(model6.2)[,6],
           accuracy(model6.3)[,6],accuracy(model6.4)[,6],accuracy(model7.1)[,6],
           accuracy(model7.2)[,6])
```

```
ss<-data.frame(model.ss,mase.ss)
colnames(ss)<-c("Model","MASE")

compare<-rbind(tsrm,ss)
rownames(compare)<-NULL
compare<-arrange(compare,MASE)
compare
```

```
##                         Model      MASE
## 1         HW Additive Damped 0.6628509
## 2   HW Multiplicative Damped 0.6628509
## 3                HW Additive 0.6645637
## 4          HW Multiplicative 0.6645637
## 5                ETS (A,A,N) 0.6645686
## 6                 ETS(M,A,N) 0.6666867
## 7  Autoregressive DLM (P5Q5) 0.7445786
## 8  Autoregressive DLM (P4Q5) 0.7452085
## 9  Autoregressive DLM (P1Q5) 0.7547836
## 10        Dynamic Lag Models 0.8025763
## 11                Finite DLM 0.8204343
## 12                 Koyck DLM 0.8924145
## 13            Polynomial DLM 1.0727151
```

Based on the `compare` output, we can observe that HW Additive Damped have lowest MASE values. Furthermore, with respective of the diagnostics checking from the previous sections, we shall use the lowest four mase models from the output for further analysis.

1.10.0 Forecasting

In this section we will use HW Additive Damped, HW Multiplicative Damped, HW Additive, and HW Multiplicative model to produce the best 4 weeks ahead forecast for mortality series.

```
plot(model6.2,ylab="Mortality",type="l",fcol="white",xlab="Year",main="Mortality 4 Weeks
         Ahead Forecasts")
lines(fitted(model6.1), col="red", lty=1)
lines(fitted(model6.2), col="green", lty=1)
lines(fitted(model6.3), col="cyan", lty=1)
lines(fitted(model6.4), col="brown", lty=1)
lines(model6.1$mean, type="l", col="red")
lines(model6.2$mean, type="l", col="green")
lines(model6.3$mean, type="l", col="cyan")
lines(model6.4$mean, type="l", col="brown")
legend("bottomleft",cex=0.7,lty=1, pch=0.5, col=1:5,
       c("data","HW Additive", "HW Additive Damped", "HW Multiplicative",
         "HW Multiplicative Damped"))
```

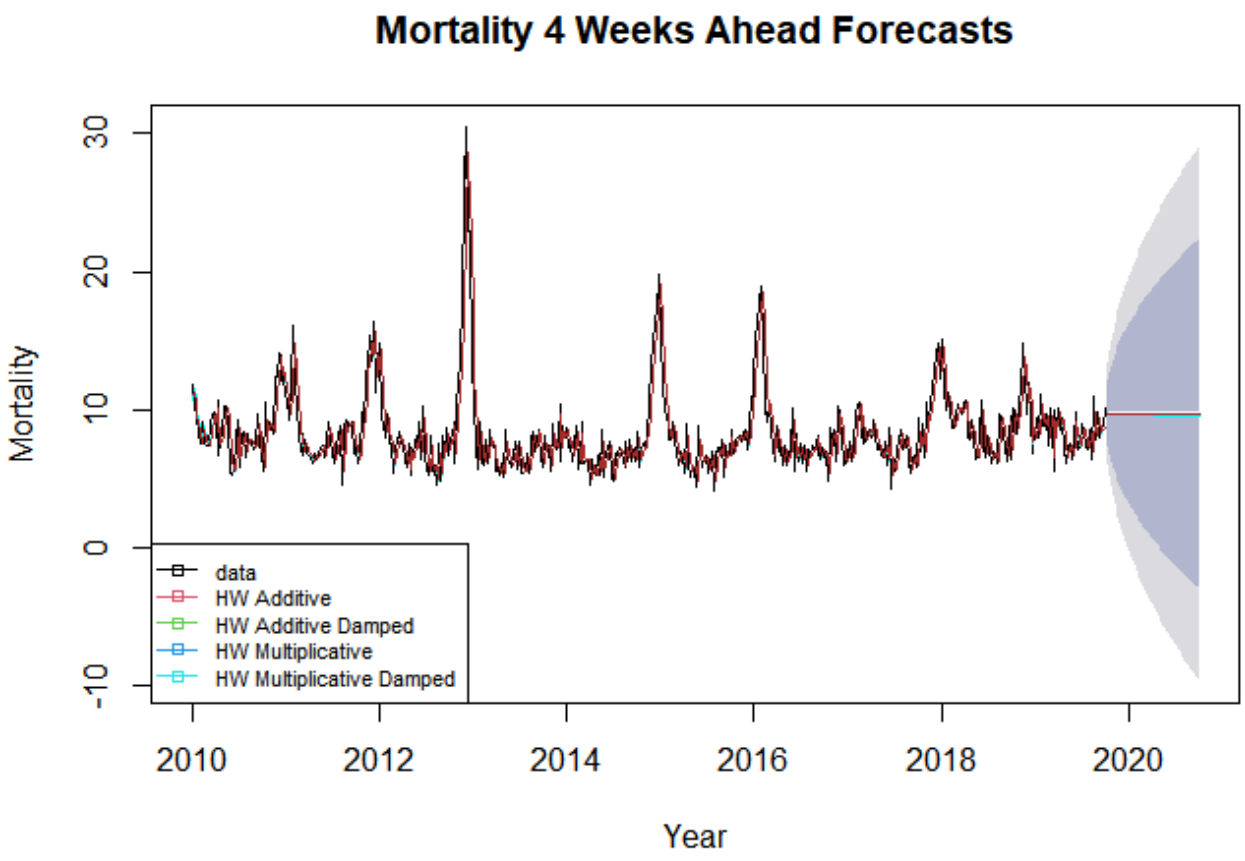

Figure 30

`Figure 30` shows us the best 4 weeks ahead forecasts using the 4 models with the lowest MASE values.

```
model.frc<-holt(mort.ts,seasonal="additive",damped=TRUE,h=4*frequency(mort.ts))
upper.95<-model.frc$upper[,2]
lower.95<-model.frc$lower[,2]
centre<-model.frc$mean
plot(mort.ts, ylim=c(-10, 30),main="Mortality 4 Weeks Ahead Forecast")
lines(centre, col = "blue")
lines(upper.95, col = "green")
lines(lower.95, col = "green")
legend("bottomleft", lty=1, cex=0.7, pch=1,
       col=c("black","blue","green"),
       c("Data","Forecasts","95% Lower and Upper Bounds"))
```

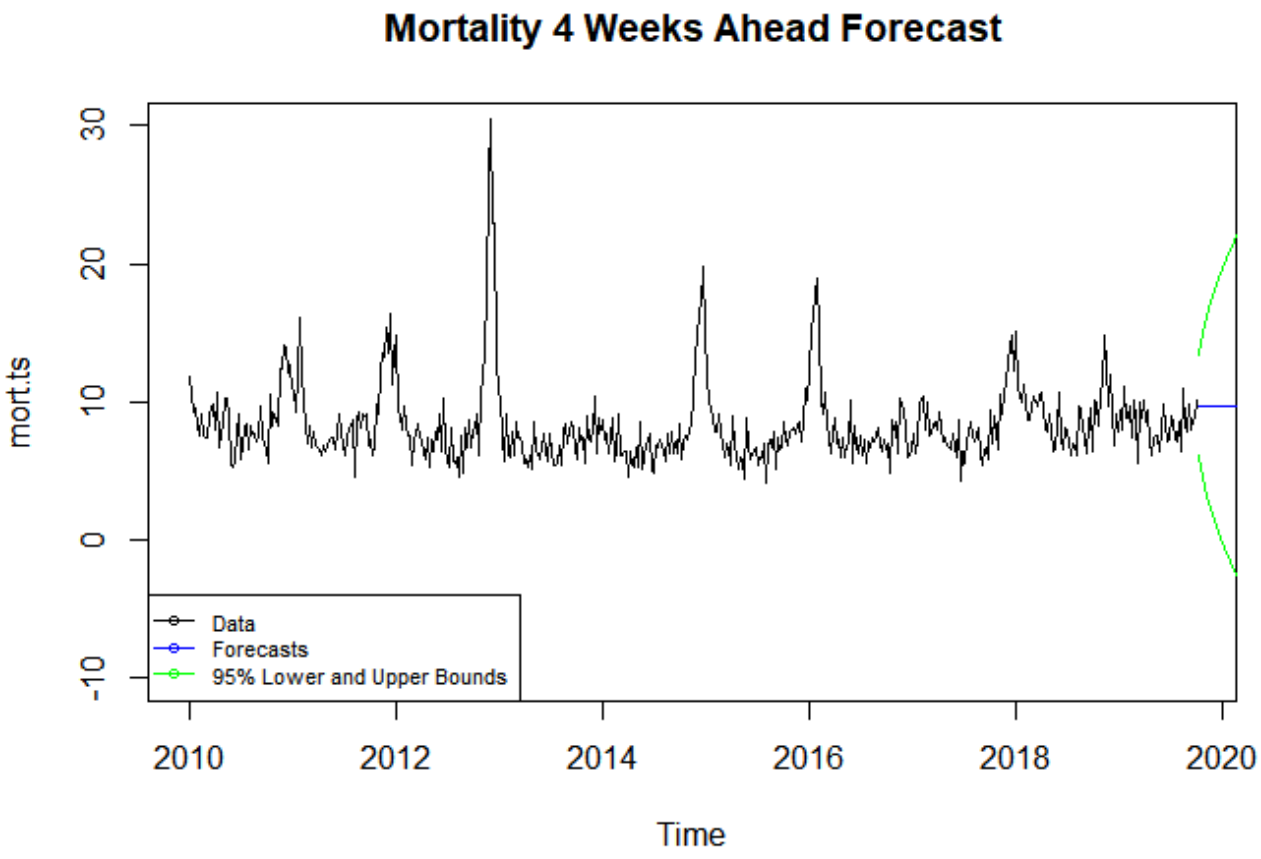

Figure 31

`Figure 31` shows the forecasts and 95% confidence interval for the mortality series.

1.11.0 Conclusion

In conclusion, through our analysis on the time series data we can see that HW Additive gives us the best model in terms of its MASE value for the four weeks forecasting.

2.0.0 Task 2

2.1.0 Introduction

The aim of this investigation for task 2 is to model and provide point forecasts and confidence intervals using univariate models for the best first flowering day (FFD) 4 years forecast in the FFD series. The data set provided `FFD.csv` contains the data for investigations on climate factors affecting the flowering day for variables rainfall, temperature, radiation level, and relative humidity. The data set collects data from 1984 to 2014. To perform these investigations, we will first prep the data for analysis and do further testing through various analysis methods.

2.2.0 Data Description and Preprocessing

The dataset `FFD.csv` and `Covariate x-values for Task 2.csv` is used for analysis in task 2. Hence, we will import the data set using the `read_csv` function.

```
ffd<-read_csv("FFD .csv")
```

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

```
ffd.covar<-read_csv("Covariate x-values for Task 2 .csv")
```

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

```
class(ffd)
```

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

```
class(ffd.covar)
```

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

```
str(ffd)
```

```
## spc_tbl_ [31 × 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Year       : num [1:31] 1984 1985 1986 1987 1988 ...
##  $ Temperature: num [1:31] 18.7 19.3 18.6 19.1 20.4 ...
##  $ Rainfall   : num [1:31] 2.49 2.48 2.42 2.32 2.47 ...
##  $ Radiation  : num [1:31] 14.9 14.7 14.5 14.7 14.7 ...
##  $ RelHumidity: num [1:31] 93.9 94.9 94.1 94.5 94.1 ...
##  $ FFD        : num [1:31] 310 322 306 306 306 293 329 293 320 323 ...
##  - attr(*, "spec")=
```

```
##   .. cols(
##   ..   Year = col_double(),
##   ..   Temperature = col_double(),
##   ..   Rainfall = col_double(),
##   ..   Radiation = col_double(),
##   ..   RelHumidity = col_double(),
##   ..   FFD = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
str(ffd.covar)
```

```
## spc_tbl_ [6 × 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Year       : num [1:6] 2015 2016 2017 2018 NA ...
##  $ Temperature: num [1:6] 20.7 20.5 20.5 20.6 NA ...
##  $ Rainfall   : num [1:6] 2.27 2.38 2.26 2.27 NA NA
##  $ Radiation  : num [1:6] 14.6 14.6 14.8 14.8 NA ...
##  $ RelHumidity: num [1:6] 94.5 94 95 95.1 NA ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Year = col_double(),
##   ..   Temperature = col_double(),
##   ..   Rainfall = col_double(),
##   ..   Radiation = col_double(),
##   ..   RelHumidity = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
head(ffd)
```

```
## # A tibble: 6 × 6
##    Year Temperature Rainfall Radiation RelHumidity   FFD
##   <dbl>       <dbl>    <dbl>     <dbl>       <dbl> <dbl>
## 1  1984        18.7     2.49      14.9        93.9   310
## 2  1985        19.3     2.48      14.7        94.9   322
## 3  1986        18.6     2.42      14.5        94.1   306
## 4  1987        19.1     2.32      14.7        94.5   306
## 5  1988        20.4     2.47      14.7        94.1   306
## 6  1989        19.6     2.74      14.8        96.1   293
```

```
head(ffd.covar)
```

```
## # A tibble: 6 × 5
##    Year Temperature Rainfall Radiation RelHumidity
##   <dbl>       <dbl>    <dbl>     <dbl>       <dbl>
```

```
## 1  2015        20.7     2.27      14.6        94.4
## 2  2016        20.5     2.38      14.6        94.0
## 3  2017        20.5     2.26      14.8        95.0
## 4  2018        20.6     2.27      14.8        95.1
## 5    NA        NA       NA        NA          NA
## 6    NA        NA       NA        NA          NA
```

```
ffd.covar<-ffd.covar[-(5:6),]
```

Here we can see that the class, structure, and head of data is what we want. However, the dataset rows 5 and 6 subset as it is not needed and is na values. Next, using the `ts` function we will convert the data into a time series data, with each variable in it's own time series vector.

```
ffd.ts<-ts(ffd[,2:6],start=c(1984,1),frequency=1)
ffd.temp.ts<-ts(ffd$Temperature,start=c(1984,1),frequency=1)
ffd.rain.ts<-ts(ffd$Rainfall,start=c(1984,1),frequency=1)
ffd.rad.ts<-ts(ffd$Radiation,start=c(1984,1),frequency=1)
ffd.hum.ts<-ts(ffd$RelHumidity,start=c(1984,1),frequency=1)
ffd.ffd.ts<-ts(ffd$FFD,start=c(1984,1),frequency=1)
ffd.covar.ts<-ts(ffd.covar[,2:5],start=c(2015,1),frequency=1)
```

2.3.0 Data Visualisation

Here we will plot each time series objects as a graph to gain a further understanding of the data through visual inspection. We will use the `plot` function to plot the graphs.

2.3.1 Temperature

```
plot(ffd.temp.ts,main="Time Series Plot for Temperature Series",
     ylab="Temperature",xlab="Year",type="o")
```

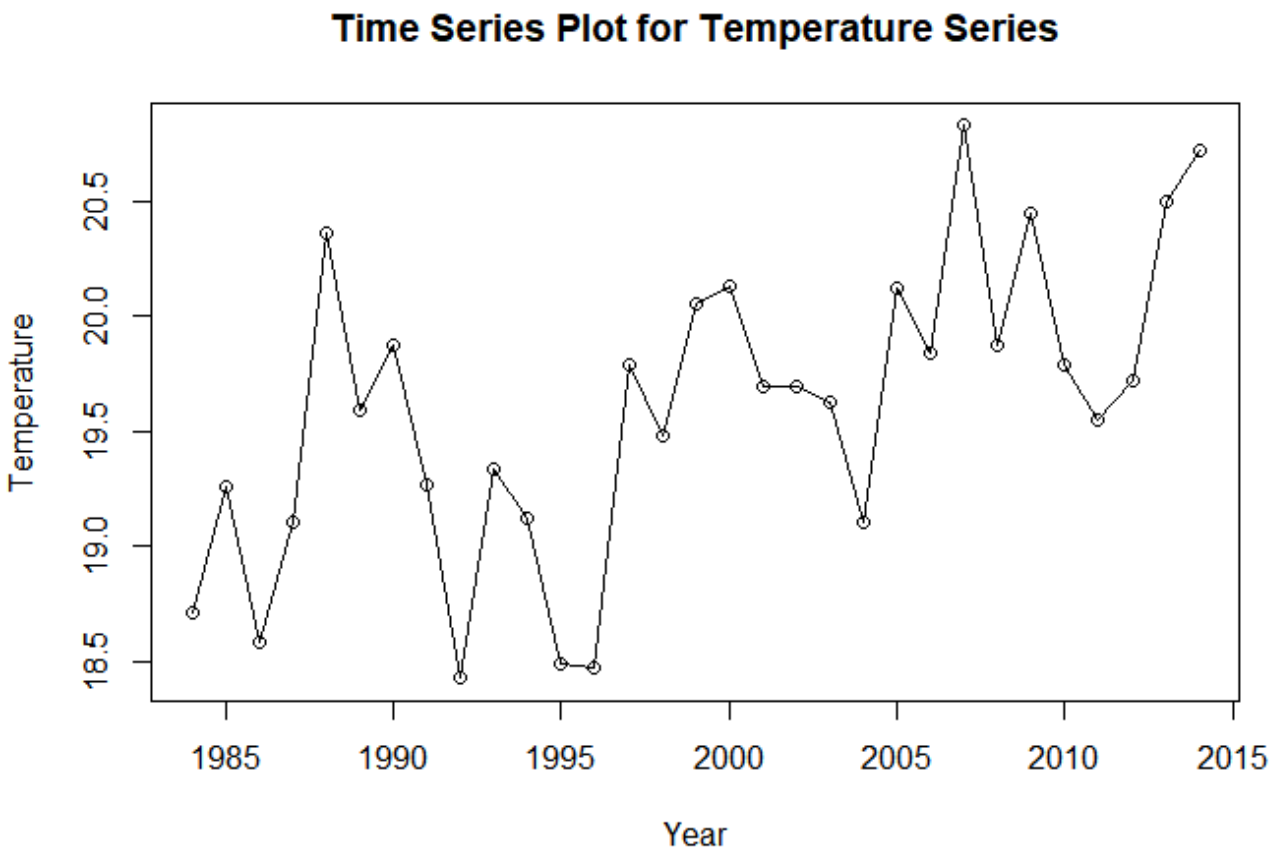

Figure 32

By `Figure 32`, which shows the time series plot of temperature series, we can make the following observations:

- The series has no apparent trends or intervention points
- There are no presence of seasonality.
- Changing variance and moving average behavior can be observed

```
acf(ffd.temp.ts, lag.max = 48, main="Sample ACF Plot for Temperature Series")
```

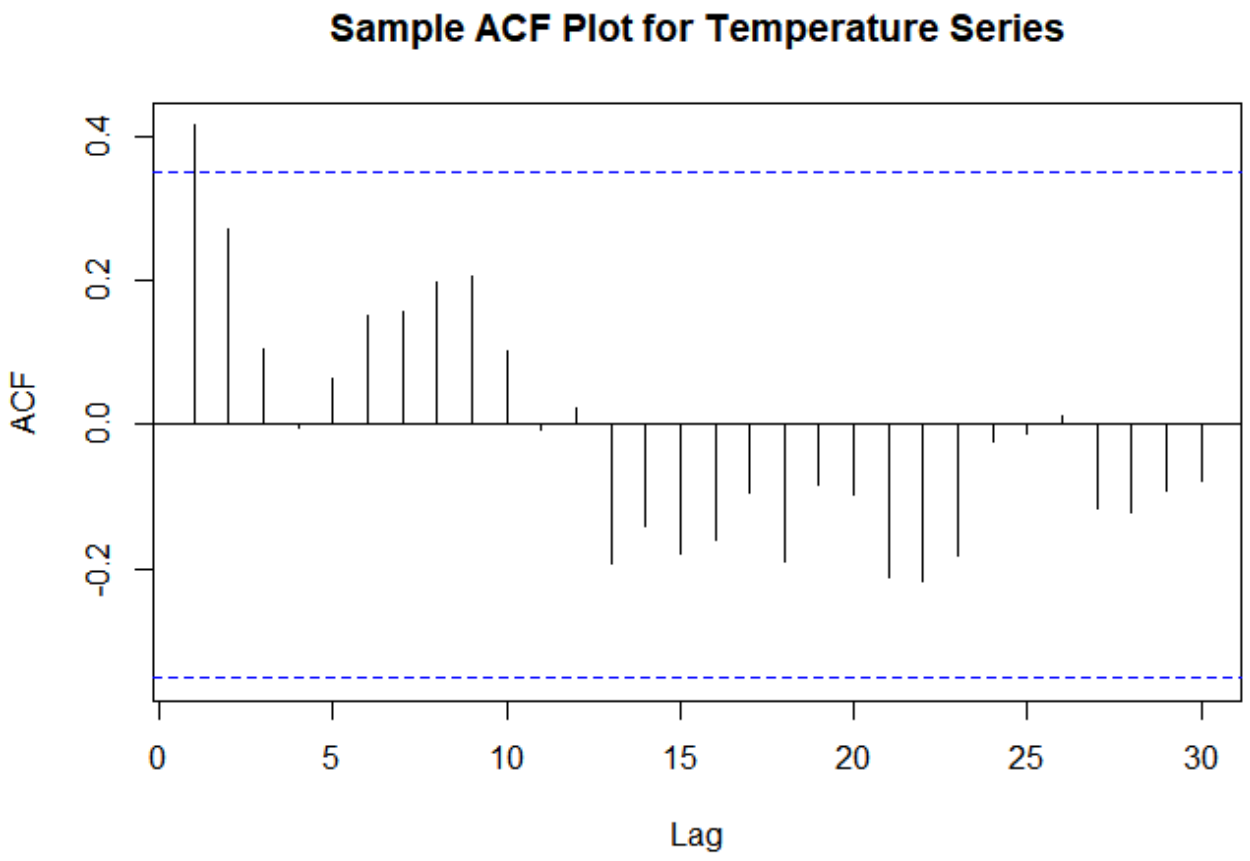

Figure 33

`Figure 33` shows the sample ACF plot for temperature series. Here we can observe that first lag is significant which suggests non-stationary in series.

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

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

In the above output, we used the ADF test for the temperature series. In this approach, we can conclude that the temperature series is non-stationary at 5% level of significance as p value is greater than 0.05.

2.3.2 Rainfall

```
plot(ffd.rain.ts,main="Time Series Plot for Rainfall Series",
     ylab="Rainfall",xlab="Year",type="o")
```

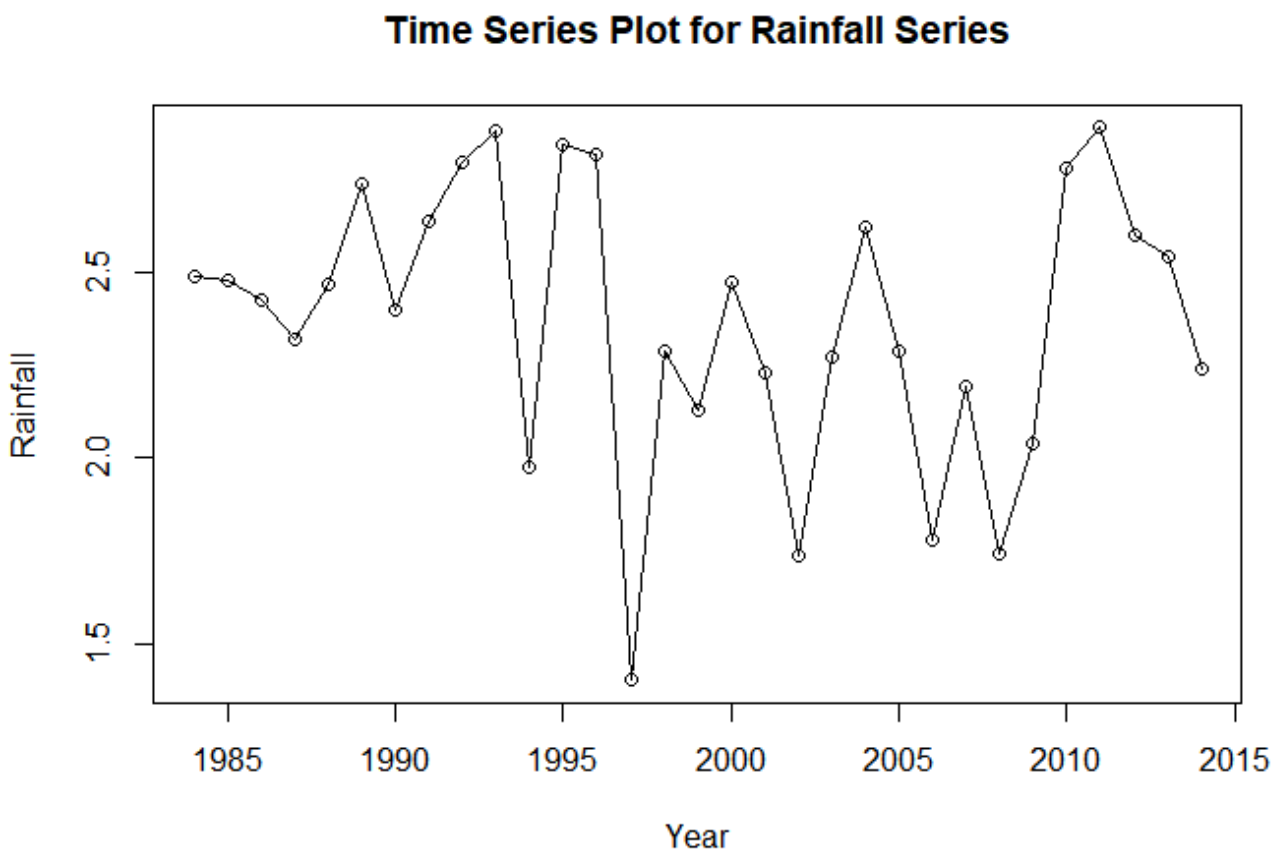

Figure 34

By `Figure 34`, which shows the time series plot of rainfall series, we can make the following observations:

- The series has no apparent trends.
- There are no presence of seasonality.
- Changing variance and moving average behavior can be observed
- There is an apparent intervention point around 1997.

```
acf(ffd.rain.ts, lag.max = 48, main="Sample ACF Plot for Rainfall Series")
```

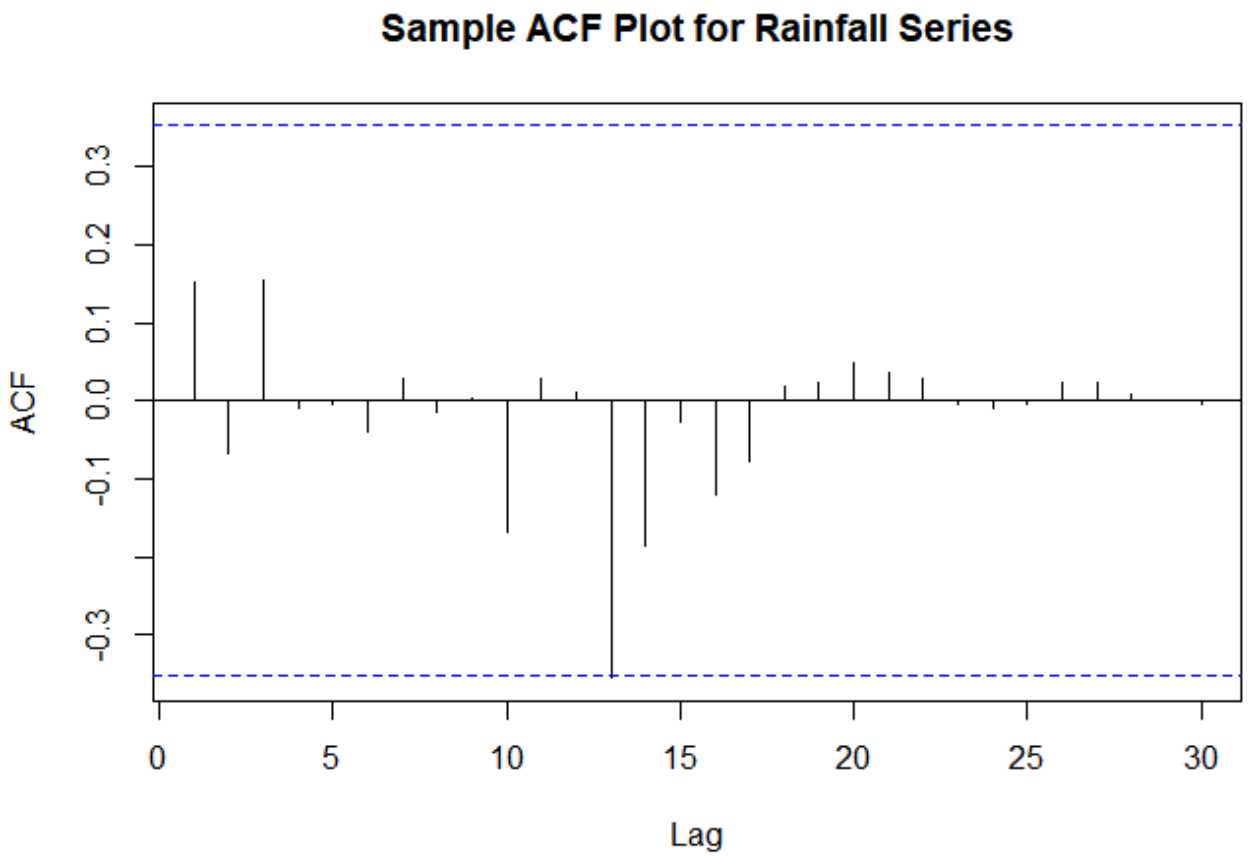

Figure 35

`Figure 35` shows the sample ACF plot for rainfall series. Here we can observe that first lag is not significant which suggests stationary in series.

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

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

In the above output, we used the ADF test for the rainfall series. In this approach, we can conclude that the temperature series is stationary at 5% level of significance as p value is less than 0.05.

2.3.3 Radiation

```
plot(ffd.rad.ts,main="Time Series Plot for Radiation Series",
     ylab="Radiation",xlab="Year",type="o")
```

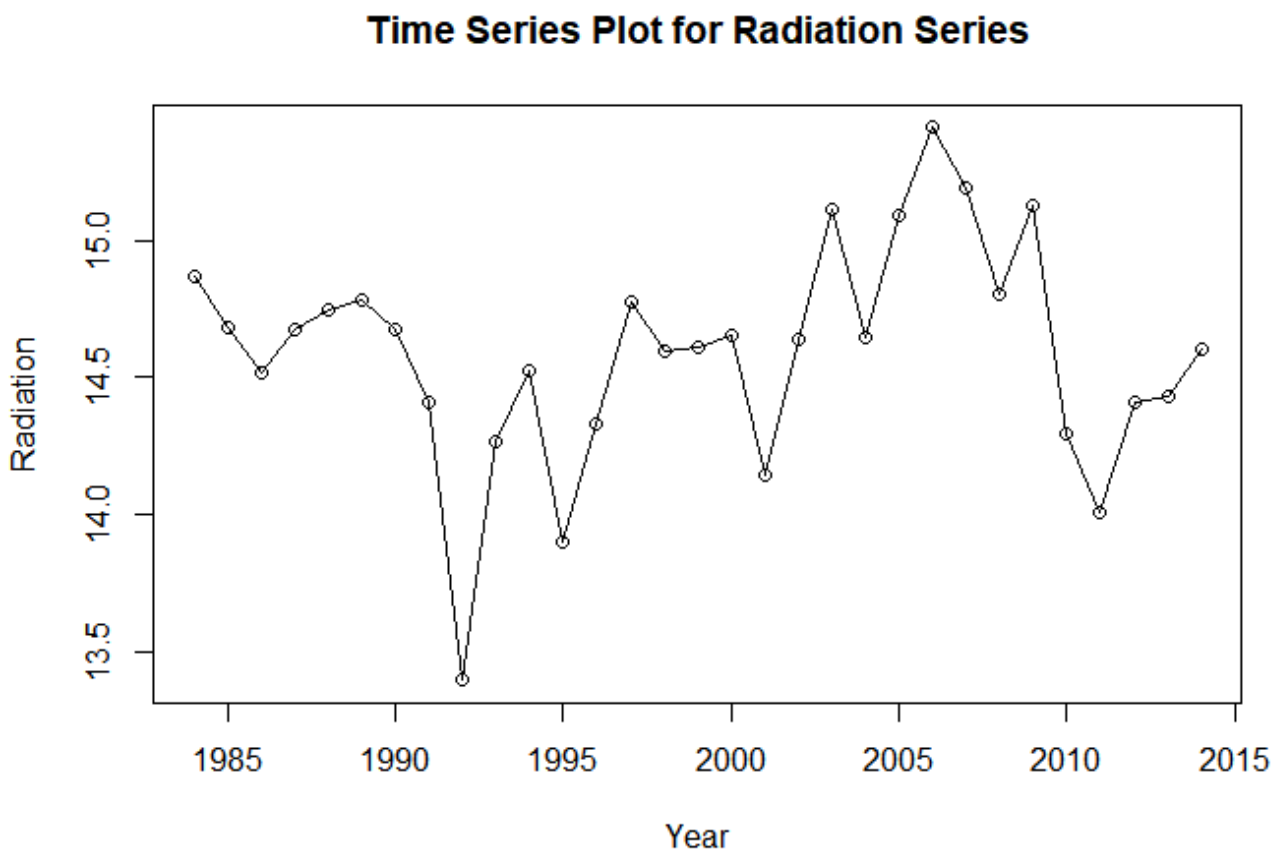

Figure 36

By `Figure 36`, which shows the time series plot of radiation series, we can make the following observations:

- The series has no apparent trends.
- There are no presence of seasonality.
- Changing variance and moving average behavior can be observed
- There is an apparent intervention point around 1992.

```
acf(ffd.rad.ts, lag.max = 48, main="Sample ACF Plot for Radiation Series")
```

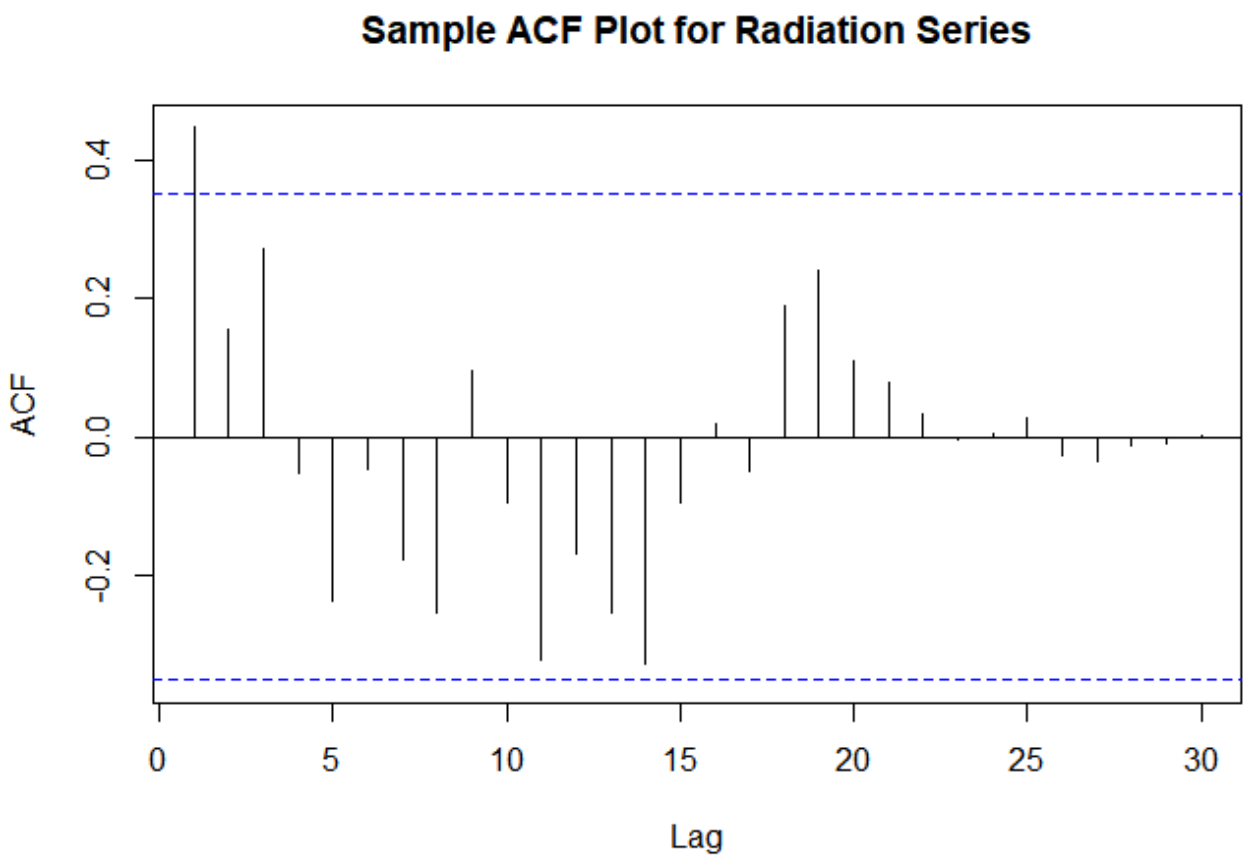

Figure 37

`Figure 37` shows the sample ACF plot for radiation series. Here we can observe that first lag is significant which suggests non-stationary in series.

```
adf.test(ffd.rad.ts, k=ar(ffd.rad.ts)$order)
```

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

In the above output, we used the ADF test for the radiation series. In this approach, we can conclude that the radiation series is non-stationary at 5% level of significance as p value is greater than 0.05.

2.3.4 Relative Humidity

```
plot(ffd.hum.ts,main="Time Series Plot for Relative Humidity Series",
     ylab="Relative Humidity",xlab="Year",type="o")
```

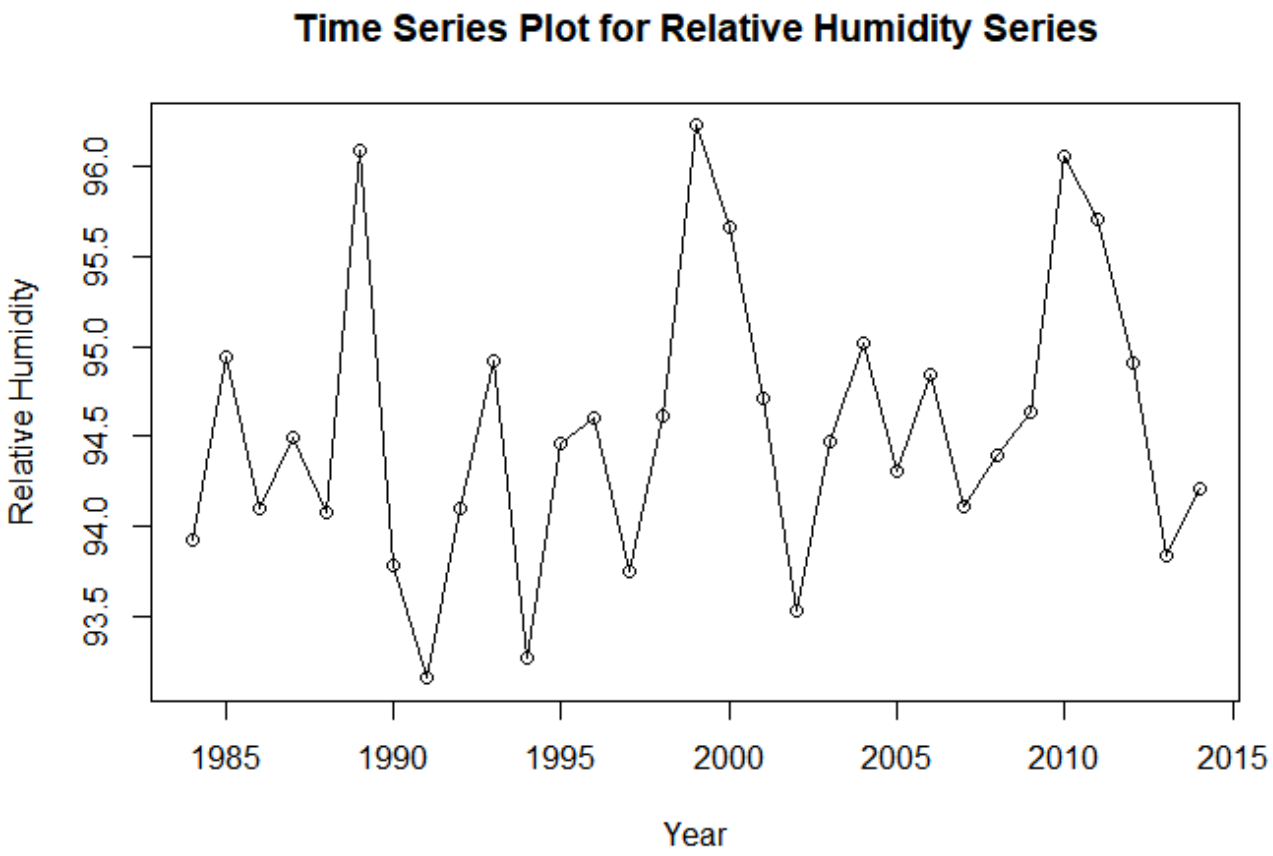

Figure 38

By `Figure 38`, which shows the time series plot of relative humidity series, we can make the following observations:

- The series has no apparent trends.
- There are no presence of seasonality.
- Changing variance and moving average behavior can be observed
- There are multiple intervention points observed.

```
acf(ffd.hum.ts, lag.max = 48, main="Sample ACF Plot for Relative Humidity Series")
```

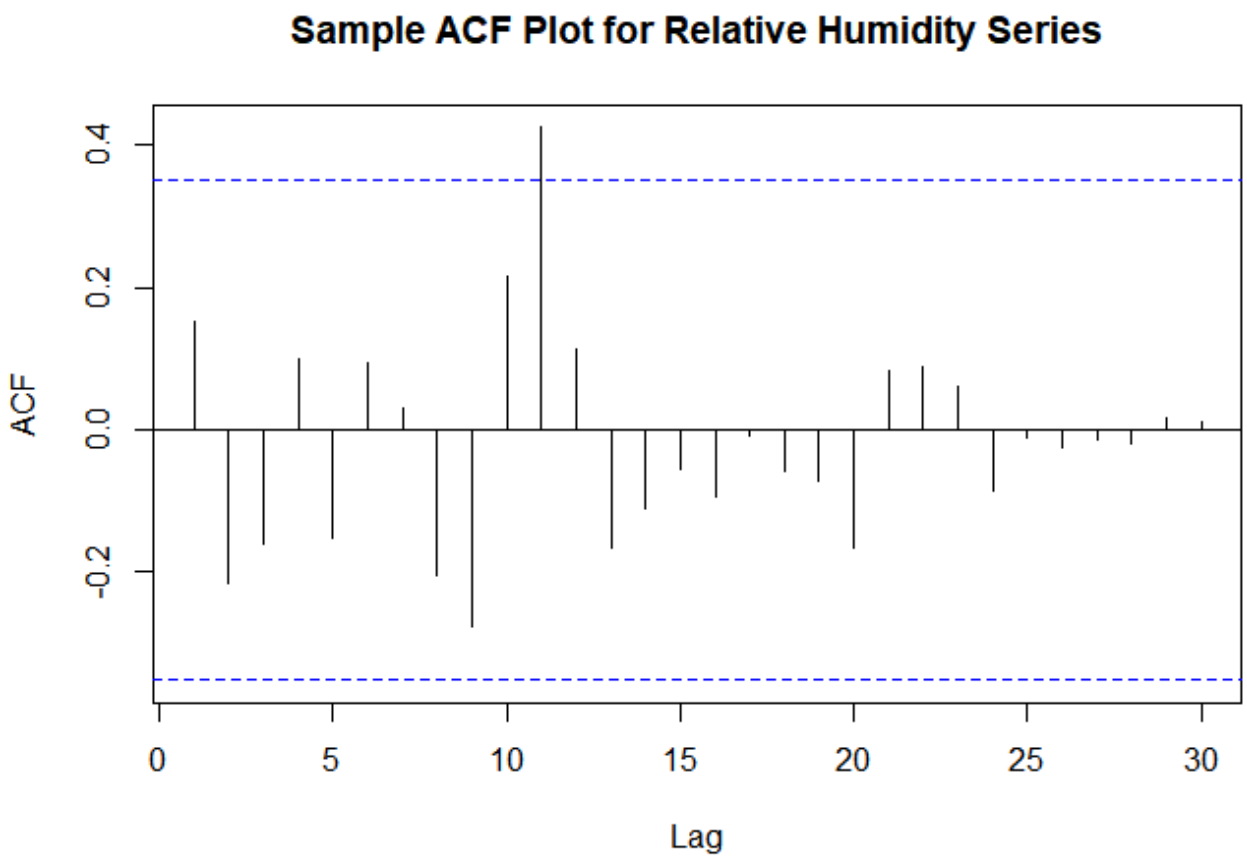

Figure 39

`Figure 39` shows the sample ACF plot for relative humidity series. Here we can observe that first lag is not significant which suggests stationary in series.

```
adf.test(ffd.hum.ts, k=ar(ffd.hum.ts)$order)
```

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

In the above output, we used the ADF test for the radiation series. In this approach, we can conclude that the radiation series is stationary at 5% level of significance as p value is less than 0.05.

2.3.5 FFD

```
plot(ffd.ffd.ts,main="Time Series Plot for FFD Series",
     ylab="FFD",xlab="Year",type="o")
```

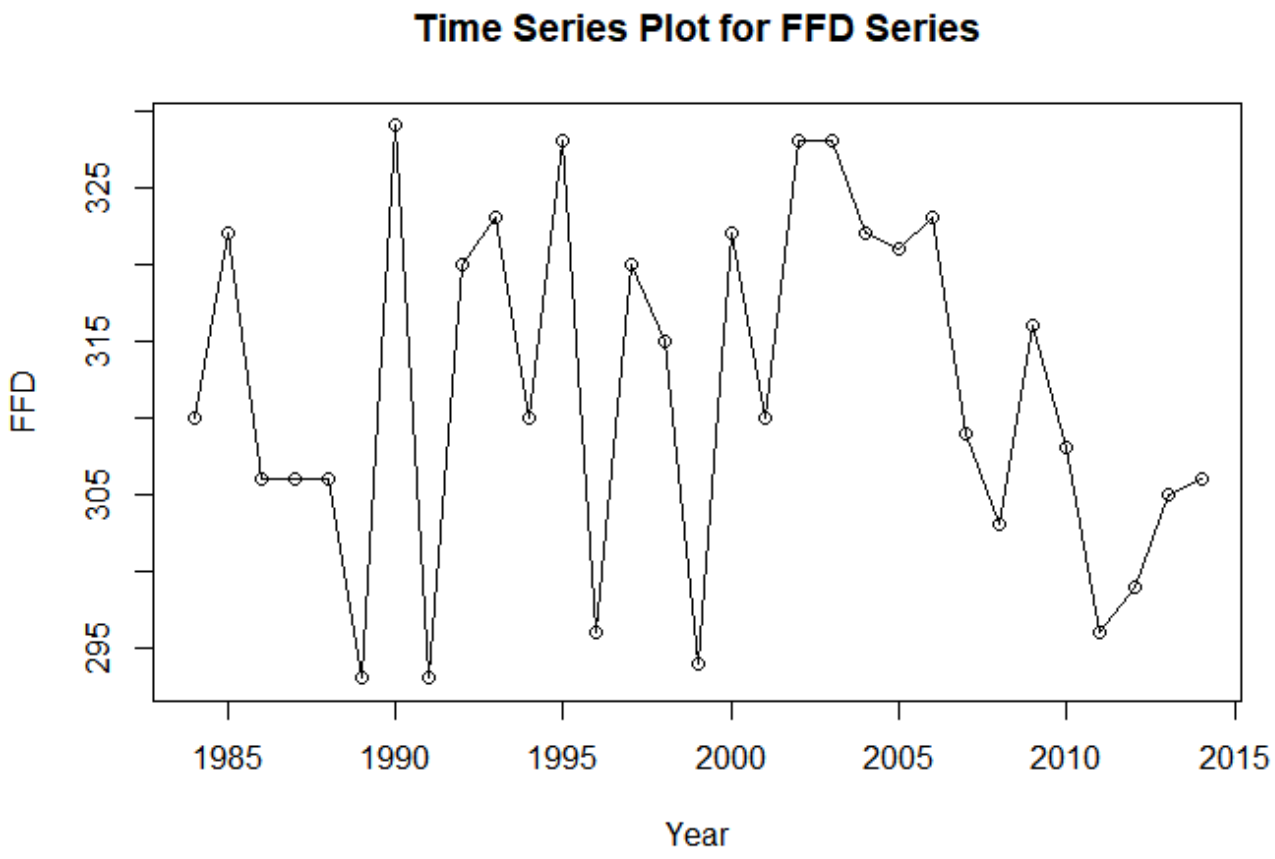

Figure 40

By `Figure 40`, which shows the time series plot of FFD series, we can make the following observations:

- The series has no apparent trends.
- There are no presence of seasonality.
- Changing variance and moving average behavior can be observed
- There are multiple intervention points observed.

```
acf(ffd.ffd.ts, lag.max = 48, main="Sample ACF Plot for FFD Series")
```

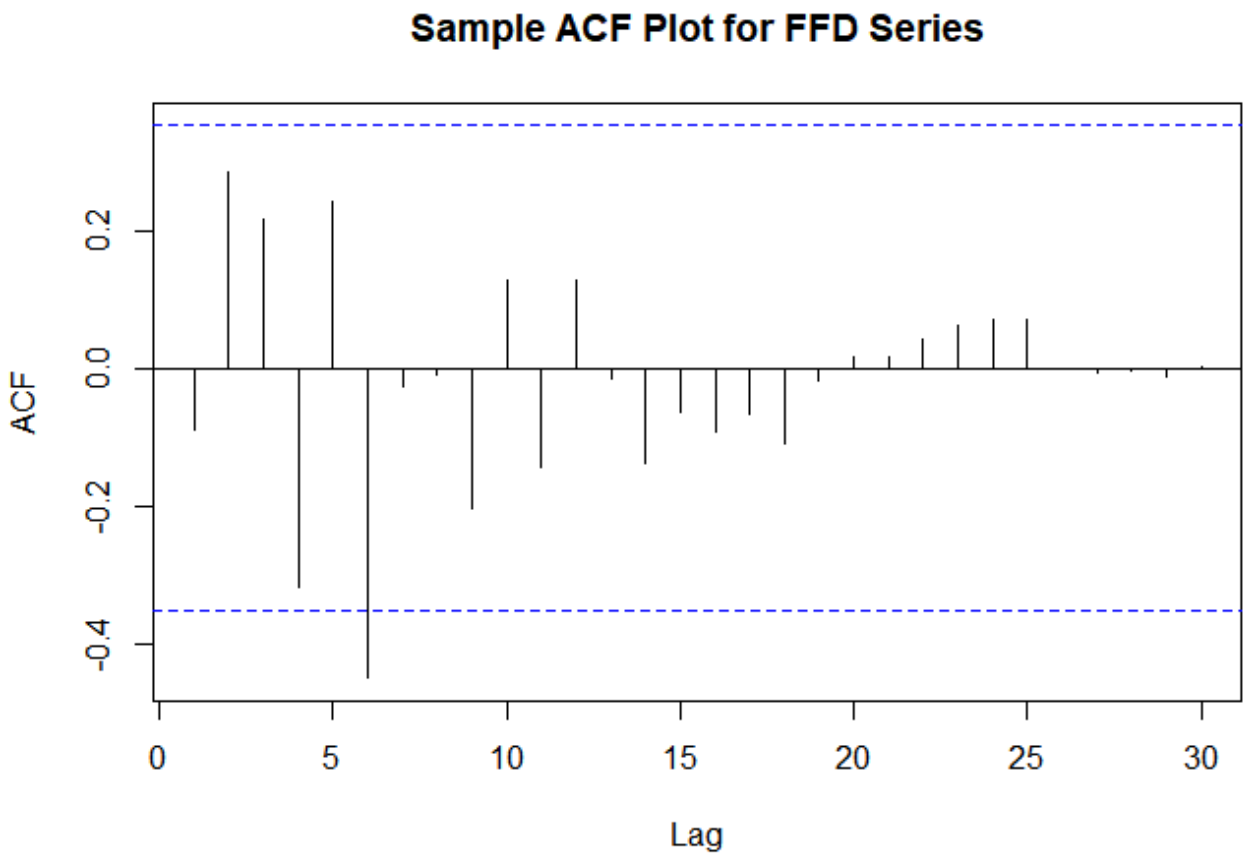

Figure 41

`Figure 41` shows the sample ACF plot for relative humidity series. Here we can observe that first lag is not significant which suggests stationary in series. However, we have to perform the augmented Dickey-Fuller test to be sure.

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

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

In the above output, we used the ADF test for the FFD series. In this approach, we can conclude that the radiation series is non-stationary at 5% level of significance as p value is greater than 0.05.

2.3.6 Scaled Data Series

```
ffd.scaled<-scale(ffd.ts)

ffd.col<-c("darkolivegreen","coral2","blueviolet","deeppink3","aquamarine")
```

```
plot(ffd.scaled,plot.type="s",col=ffd.col,main="Time Series Plot of Scaled Data Series")
legend("bottomleft",cex=0.7,lty=1,col=ffd.col,c("Temperature","Rainfall","Radiation","Rel-
        Humidity","FFD"))
```

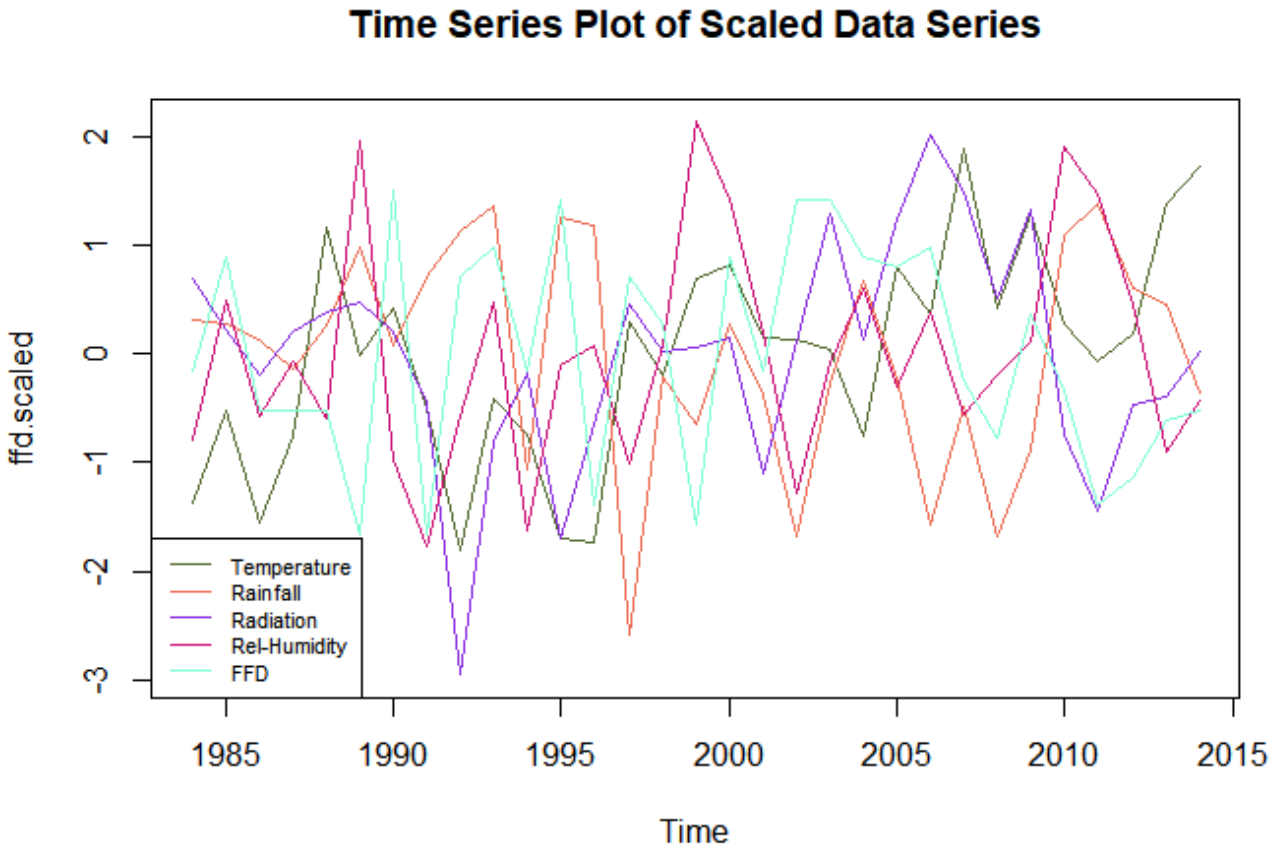

Figure 42

`Figure 42` contains a time series plot of all the data for ffd. To compare these data and fit them in a graph, scaling and centering is done to the data by using the `scale` function.

2.4.0 Dealing with Non-Stationary

From the data visualization section we have identified that there are existence of non-stationary in variables temperature, radiation, and ffd. Hence in this section we will attempt to deal with the non-stationary by using ordinary differencing operation.

2.4.1 Temperature

First we will produce the acf and pacf plot for getting a grasp of the changes made by differencing.

```
par(mfrow=c(1,2))
acf(ffd.temp.ts,lag.max = 48,main="ACF: Temperature")
pacf(ffd.temp.ts,lag.max = 48,main="PACF: Temperature")
```

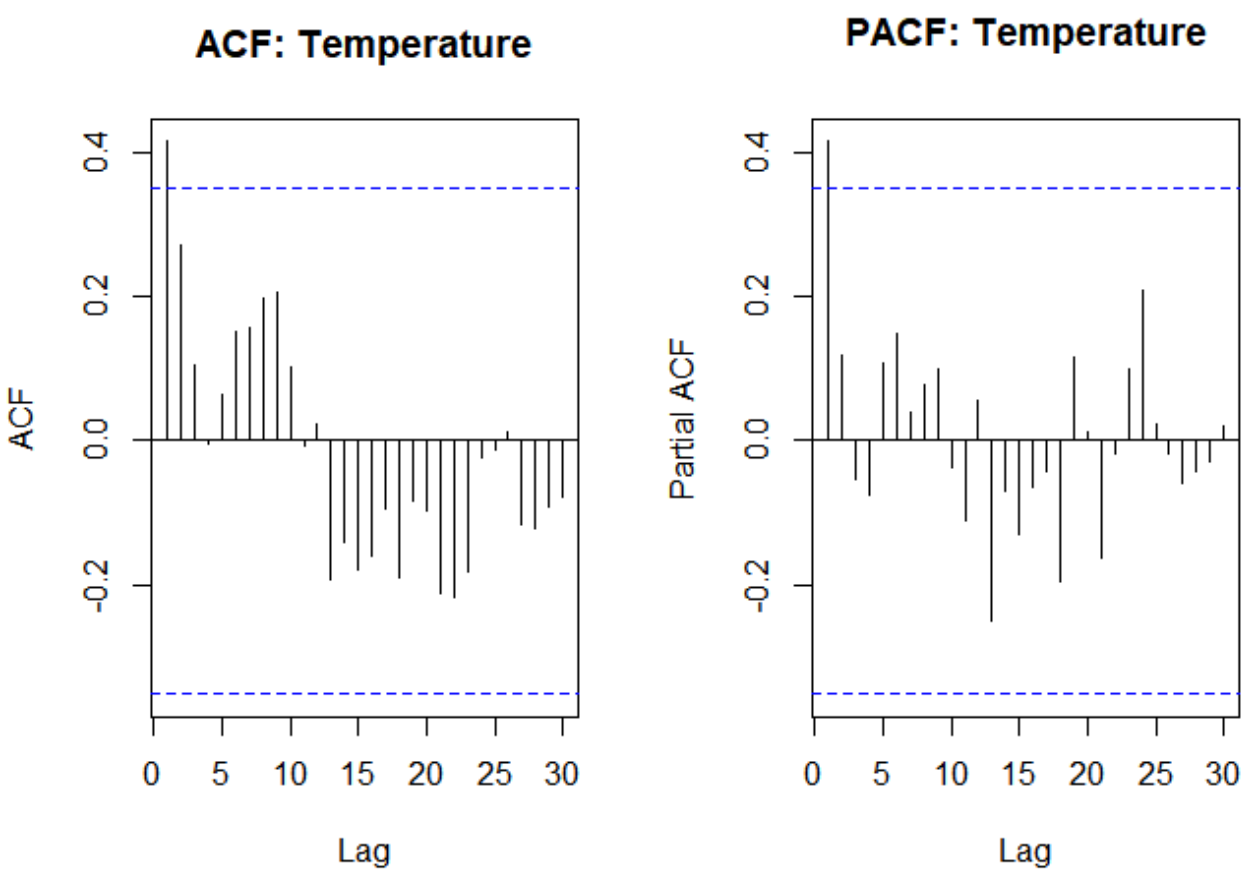

Figure 43

Now we will use the function `diff` to deal with non-stationary data with first difference order.

```
ffd.temp.diff<-diff(ffd.temp.ts,differences=1)
adf.test(ffd.temp.diff, k=ar(ffd.temp.diff)$order)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ffd.temp.diff
## Dickey-Fuller = -4.6021, Lag order = 1, p-value = 0.01
## alternative hypothesis: stationary
```

In the adf output above, we can conclude that the temperature series is now stationary at 5% level of significance as p value is less than 0.05. So we can now plot the acf and pacf plot of the differenced temperature data.

```
par(mfrow=c(1,2))
acf(ffd.temp.diff,lag.max = 48,main="ACF: Temperature Diff")
pacf(ffd.temp.diff,lag.max = 48,main="PACF: Temperature Diff")
```

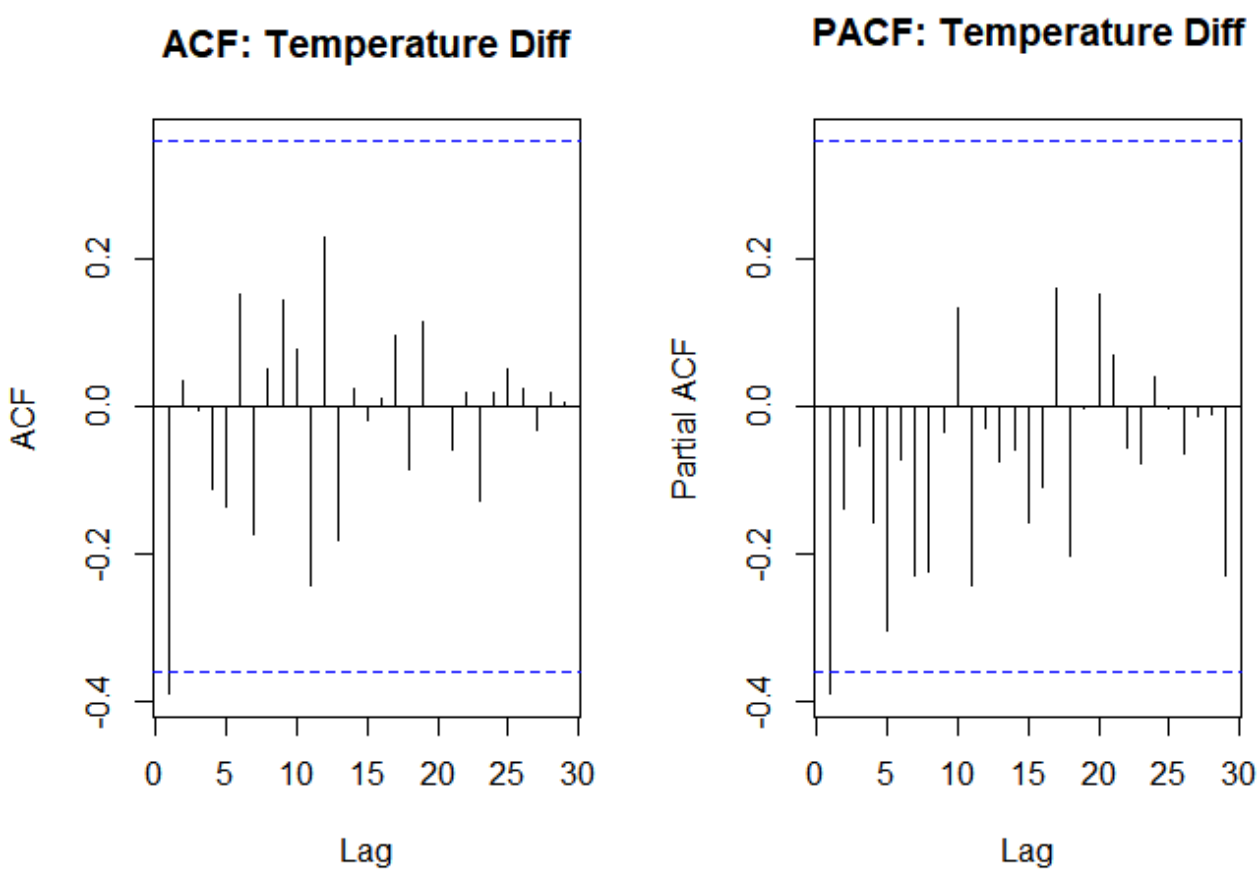

Figure 44

2.4.2 Radiation

First we will produce the acf and pacf plot for getting a grasp of the changes made by differencing.

```
par(mfrow=c(1,2))
acf(ffd.rad.ts,lag.max = 48,main="ACF: Radiation")
pacf(ffd.rad.ts,lag.max = 48,main="PACF: Radiation")
```

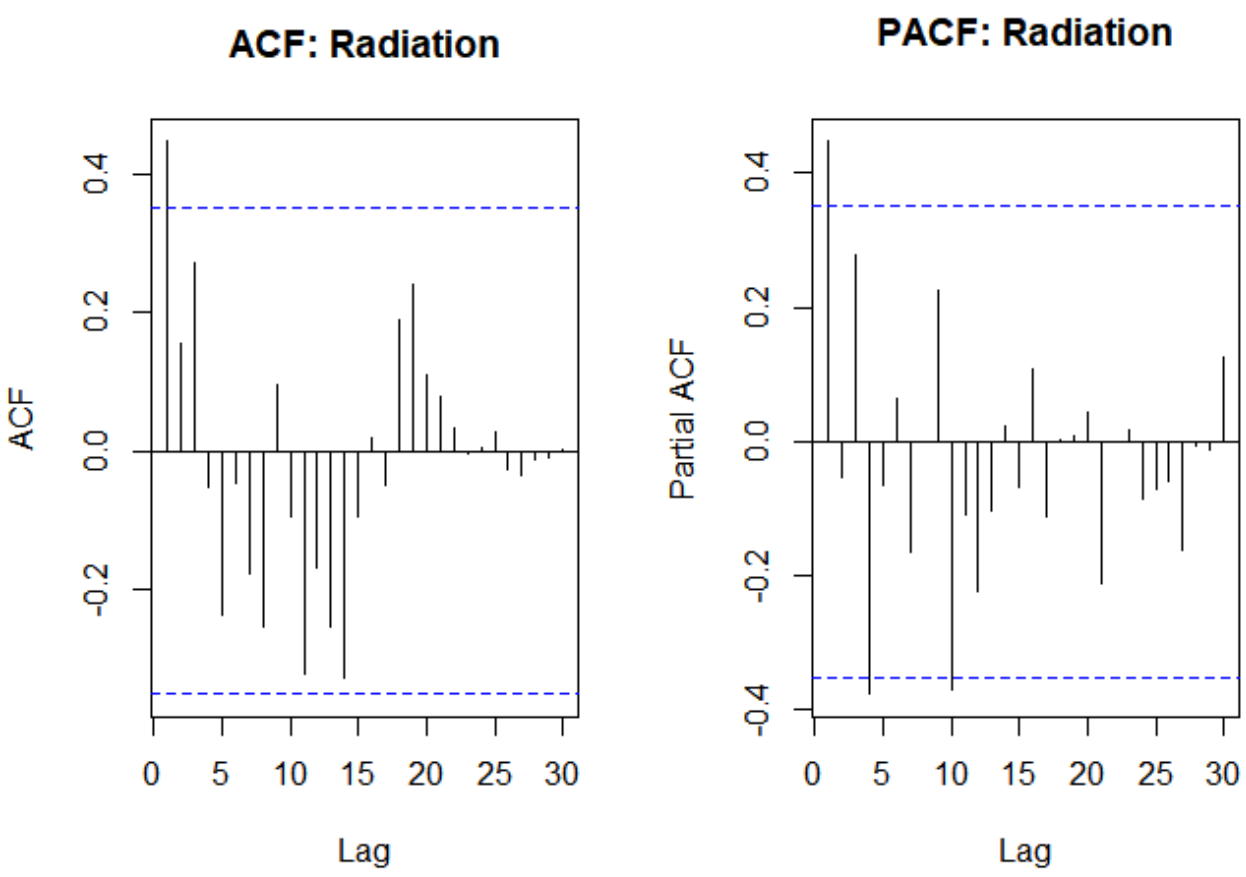

Figure 45

Now we will use the function `diff` to deal with non-stationary data with fourth difference order.

```
ffd.rad.diff<-diff(ffd.rad.ts,differences=4)
adf.test(ffd.rad.diff, k=ar(ffd.rad.diff)$order)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ffd.rad.diff
## Dickey-Fuller = -5.0992, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

In the adf output above, we can conclude that the radiation series is now stationary at 5% level of significance as p value is less than 0.05. So we can now plot the acf and pacf plot of the differenced radiation data.

```
par(mfrow=c(1,2))
acf(ffd.rad.diff,lag.max = 48,main="ACF: Radiation Diff")
pacf(ffd.rad.diff,lag.max = 48,main="PACF: Radiation Diff")
```

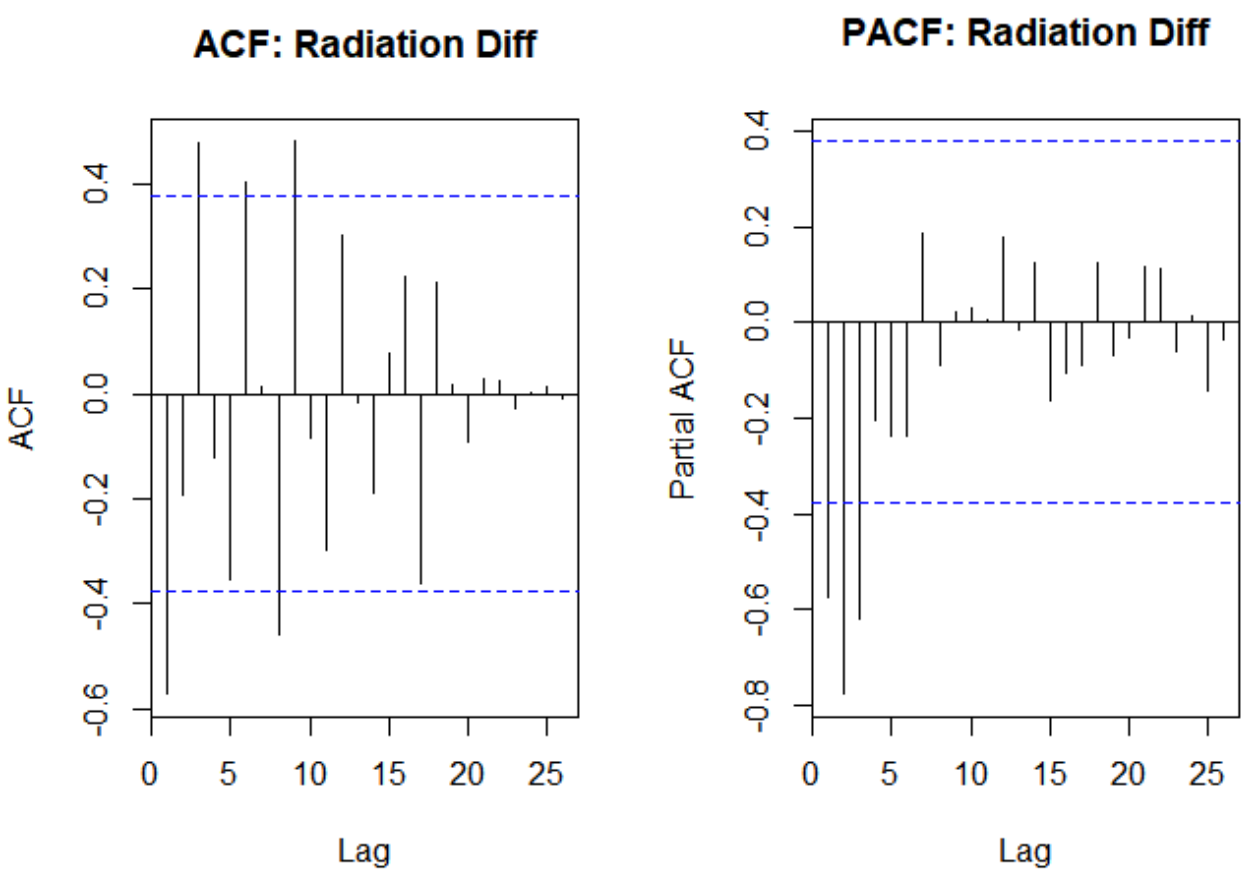

Figure 46

2.4.3 FFD

First we will produce the acf and pacf plot for getting a grasp of the changes made by differencing.

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

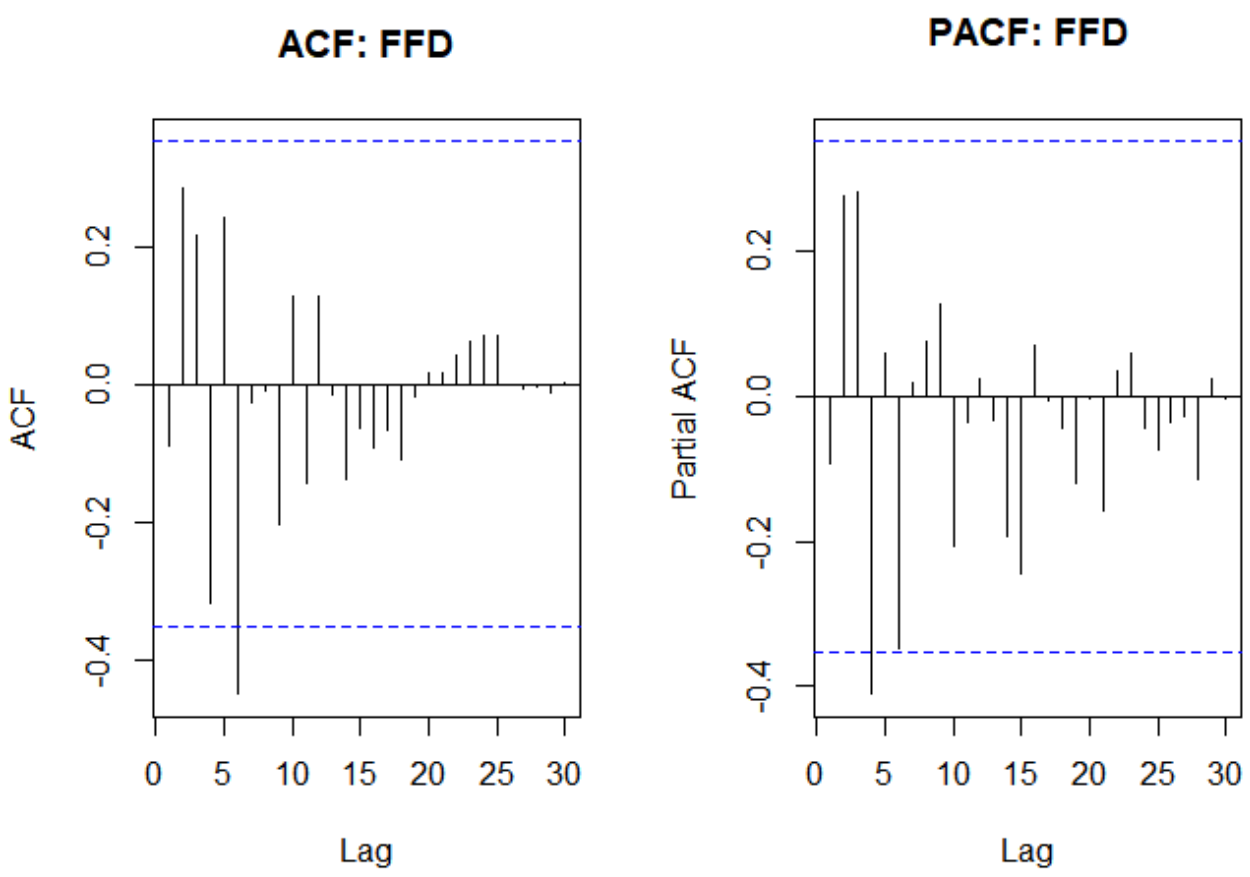

Figure 47

Now we will use the function `diff` to deal with non-stationary data with second difference order.

```
ffd.ffd.diff<-diff(ffd.ffd.ts,differences=2)
adf.test(ffd.ffd.diff, k=ar(ffd.ffd.diff)$order)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ffd.ffd.diff
## Dickey-Fuller = -4.7362, Lag order = 2, p-value = 0.01
## alternative hypothesis: stationary
```

In the adf output above, we can conclude that the ffd series is now stationary at 5% level of significance as p value is less than 0.05. So we can now plot the acf and pacf plot of the differenced ffd data.

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

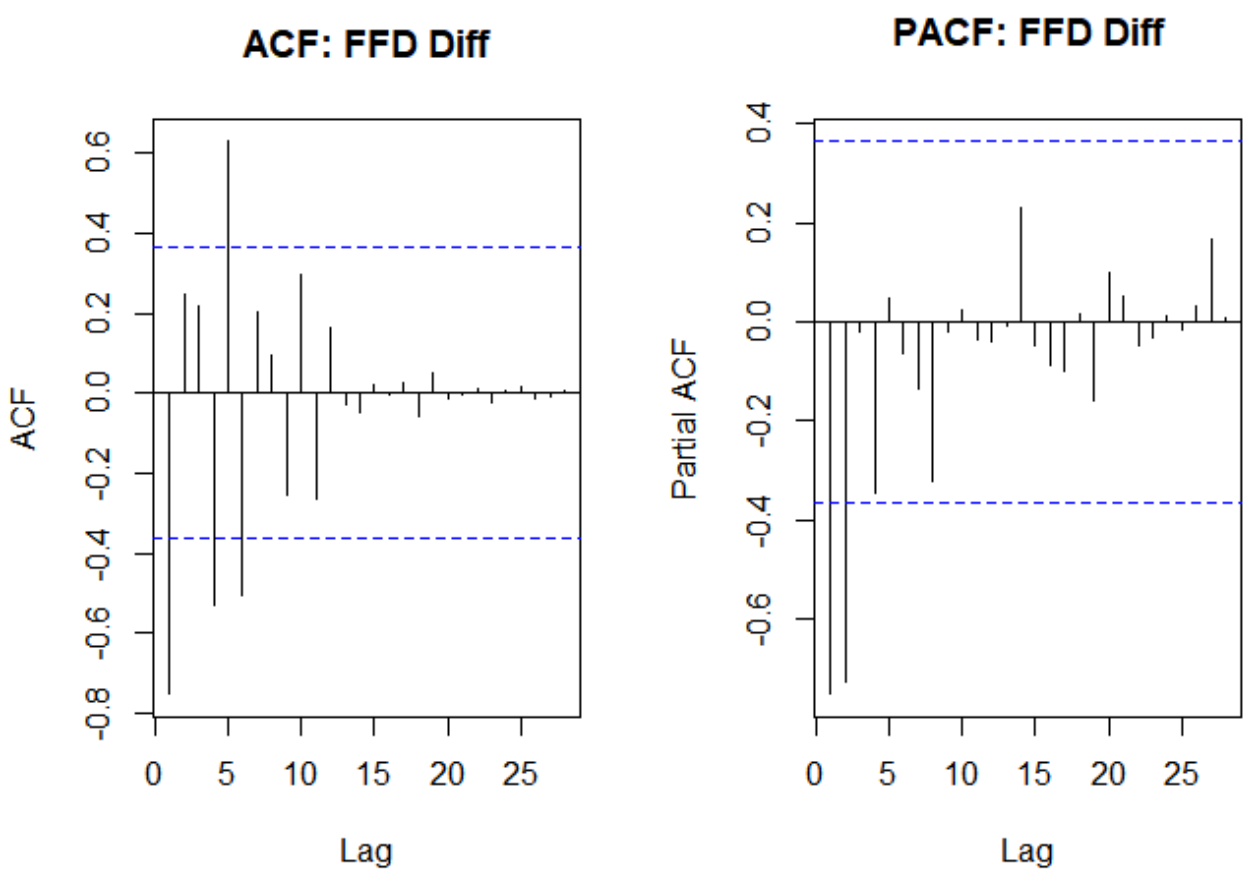

Figure 48

With the differencing operation, we are successful in dealing with the non-stationary data by using order differences 1, 4, and 2 for temperature, radiation, and ffd series.

2.5.0 Correlation

In this section, we will look at the correlation matrix for the data set by using the `cor` function.

```
cor(ffd.ts)
```

```
##             Temperature   Rainfall   Radiation RelHumidity         FFD
## Temperature  1.00000000 -0.3915072  0.51935760  0.09350562 -0.05387003
## Rainfall    -0.39150719  1.0000000 -0.58131610  0.33846101 -0.22125196
## Radiation    0.51935760 -0.5813161  1.00000000 -0.05520965  0.13759174
## RelHumidity  0.09350562  0.3384610 -0.05520965  1.00000000 -0.22660091
## FFD         -0.05387003 -0.2212520  0.13759174 -0.22660091  1.00000000
```

From the correlation matrix output above, we can observe that FFD has a weak negative correlation with Temperature of -0.05387003, week negative correlation with Rainfall of -0.22125196, weak positive correlation with Radiation of 0.13759174, and weak negative correlation with Relative Humidity of -0.22660091. Since we will be analyzing FFD, we shall use FFD as the dependent variable y and the rest of the variables in the series as independent variable x. To further perform univariate analysis of

predictors, we shall test each variable with every models and compare mase values to find the best model.

2.6.0 Time Series Regression Models

2.6.1 Finite Distributed Lag Models

For Finite DLM, we will be using temperature as the independent variable, and the ffd will be the dependent variable. To perform this test, we will first create a loop function and use the dlm function to identify most appropriate number of lags for the DLM.

```
for (i in 10){
model1<-dlm(x=as.vector(ffd.temp.ts), y=as.vector(ffd.ffd.ts), q=i)
cat("q =",i,"AIC =",AIC(model1$model),"BIC =",BIC(model1$model),"MASE=", MASE(model1)$MASE,
"\n")
}
```

```
## q = 10 AIC = 170.8439 BIC = 184.4227 MASE= 0.5495688
```

From the loop function above we will use q=10 as our argument for lag length in the analysis of other variables. Now we will use the `dlm` function to fit the finite dlm for other predictors, and we will compare the aic, bic, and mase values for each predictors.

```
#FFD & Temperature
model1.1<-dlm(x=as.vector(ffd.temp.ts),y=as.vector(ffd.ffd.ts),q=10)
#FFD & Rainfall
model1.2<-dlm(x=as.vector(ffd.rain.ts),y=as.vector(ffd.ffd.ts),q=10)
#FFD & Radiation
model1.3<-dlm(x=as.vector(ffd.rad.ts),y=as.vector(ffd.ffd.ts),q=10)
#FFD & Relative Humidity
model1.4<-dlm(x=as.vector(ffd.hum.ts),y=as.vector(ffd.ffd.ts),q=10)
#Compare
predictor<-c("Temperature","Rainfall","Radiation","Relative Humidity")
fdlm.mase<-MASE(model1.1,model1.2,model1.3,model1.4)
fdlm.aic<-c(AIC(model1.1),AIC(model1.2),AIC(model1.3),AIC(model1.4))
```

```
## [1] 170.8439
## [1] 159.5287
## [1] 134.8487
## [1] 172.2492
```

```
fdlm.bic<-c(BIC(model1.1),BIC(model1.2),BIC(model1.3),BIC(model1.4))
```

```
## [1] 184.4227
## [1] 173.1074
## [1] 148.4275
## [1] 185.828
```

```
fdlm<-data.frame(predictor,fdlm.mase,fdlm.aic,fdlm.bic)
arrange(fdlm,MASE)
```

```
##                  predictor  n      MASE fdlm.aic fdlm.bic
## model1.3         Radiation 21 0.2319041 134.8487 148.4275
## model1.2          Rainfall 21 0.4554174 159.5287 173.1074
## model1.1       Temperature 21 0.5495688 170.8439 184.4227
## model1.4 Relative Humidity 21 0.5702439 172.2492 185.8280
```

From the predictor comparison output above we can see that FFD and Radiation has the lowest AIC, BIC , and MASE values, hence we shall select this predictor for further analysis.

```
summary(model1.3)
```

```
##
## 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
```

By using the `summary` function, we can observe that in `model1.3` the lag weights of the predictors are significant at the 5% statistical level as p-value < 0.05. Furthermore, there is a adjusted R-square value of 0.8021, which accounts for 80.21% of the variability in the data. Here we can also see that for the lags, positive lag coefficients show that the flowering is later, and negative lag coefficients shows earlier flowering. First lag indicates that flowering is later as x.1=17.47280, which is positive value.

```
checkresiduals(model1.3$model$residuals)
```

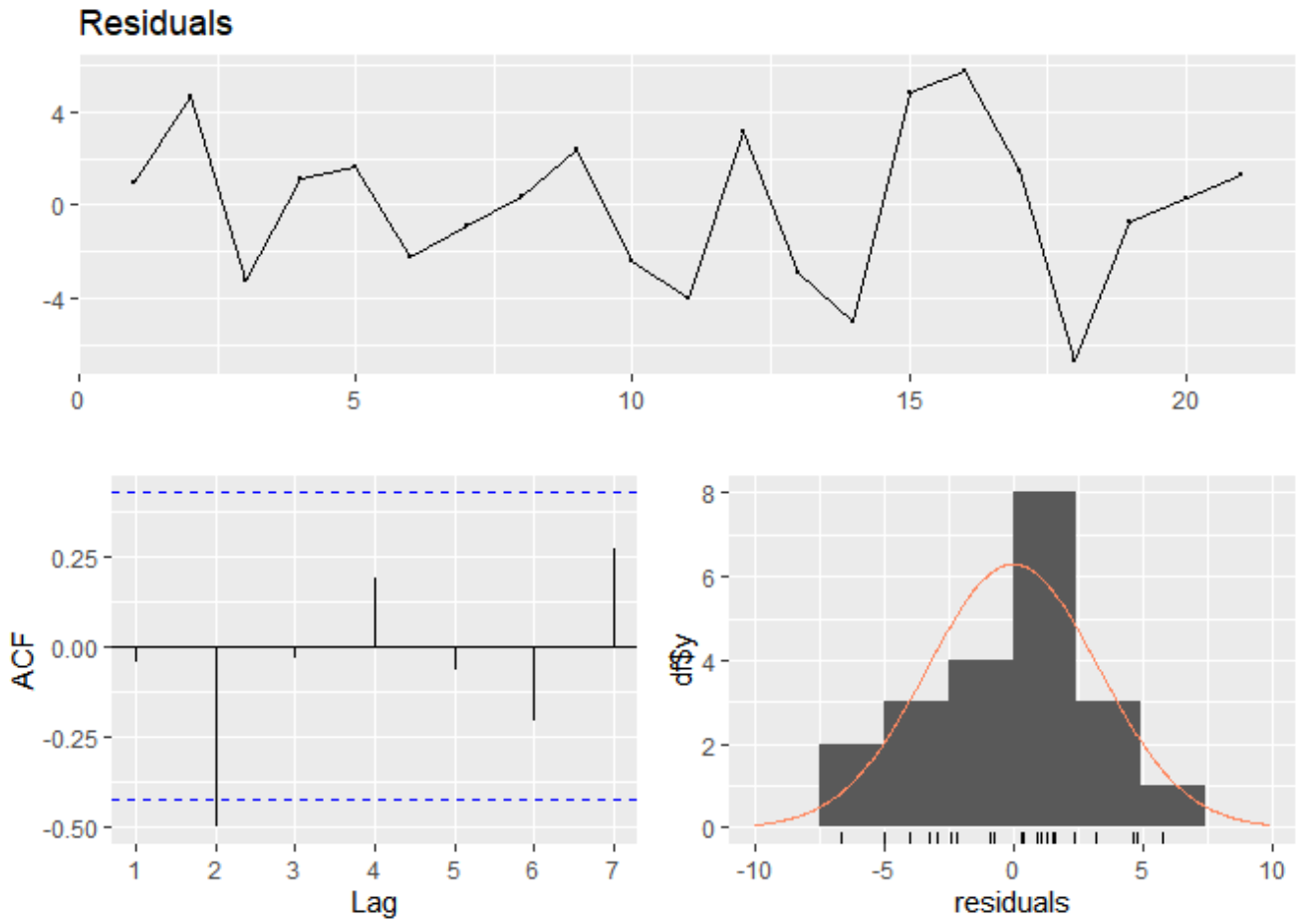

Figure 49

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

```
bgtest(model1.3$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model1.3$model
## LM test = 0.041173, df = 1, p-value = 0.8392
```

```
shapiro.test(model1.3$model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model1.3$model$residuals
## W = 0.98112, p-value = 0.9404
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 49`. The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are not significant. Furthermore, from the Shapiro-Wilk test we can conclude that we do not reject null hypothesis of normality as p-value > 0.05.

```
VIF.model1.3<-vif(model1.3$model)
VIF.model1.3
```

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

```
VIF.model1.3>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
```

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the vif function, we can observe that there is no issue with multicollinearity as all VIF values are less than 10.

2.6.2 Polynomial Distributed Lag Model

Similar to the finite DLM, we shall compare each predictor with the polynomial dlm and select the best with lowest aic, bic, and mase values.

```r
#FFD & Temperature
model2.1<-polyDlm(x=as.vector(ffd.temp.ts),y=as.vector(ffd.ffd.ts),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
```

```r
#FFD & Rainfall
model2.2<-polyDlm(x=as.vector(ffd.rain.ts),y=as.vector(ffd.ffd.ts),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
```

```r
#FFD & Radiation
model2.3<-polyDlm(x=as.vector(ffd.rad.ts),y=as.vector(ffd.ffd.ts),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
```

```
#FFD & Relative Humidity
model2.4<-polyDlm(x=as.vector(ffd.hum.ts),y=as.vector(ffd.ffd.ts),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
```

```
#Compare
predictor<-c("Temperature","Rainfall","Radiation","Relative Humidity")
pdlm.mase<-MASE(model2.1,model2.2,model2.3,model2.4)
pdlm.aic<-c(AIC(model2.1),AIC(model2.2),AIC(model2.3),AIC(model2.4))
```

```
## [1] 163.7328
## [1] 164.9582
## [1] 163.2937
## [1] 163.5124
```

```
pdlm.bic<-c(BIC(model2.1),BIC(model2.2),BIC(model2.3),BIC(model2.4))
```

```
## [1] 168.9554
## [1] 170.1808
## [1] 168.5163
## [1] 168.735
```

```
pdlm<-data.frame(predictor,pdlm.mase,pdlm.aic,pdlm.bic)
arrange(pdlm,MASE)
```

```
##                  predictor  n      MASE pdlm.aic pdlm.bic
## model2.3         Radiation 21 0.6355363 163.2937 168.5163
## model2.2          Rainfall 21 0.6551896 164.9582 170.1808
## model2.1       Temperature 21 0.6796679 163.7328 168.9554
## model2.4 Relative Humidity 21 0.7312748 163.5124 168.7350
```

From the output above, we can observe that similar to Finite DLM predictor radiation has the lowest AIC, BIC, and MASE scores. Hence, we shall use this predictor for further analysis.

```
summary(model2.3)
```

```
##
## 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
```

By using the `summary` function, we can observe that in `model2.3` the lag weights of the predictors are not significant at the 5% statistical level as p-value > 0.05. Furthermore, there is a adjusted R-square value of 0.1301, which accounts for 13.01% of the variability in the data. Here we can also see that for the lags, positive lag coefficients show that the flowering is later, and negative lag coefficients shows earlier flowering. First lag indicates that flowering is early as z.t1=-0.8977, which is negative value.

```
checkresiduals(model2.3$model$residuals)
```

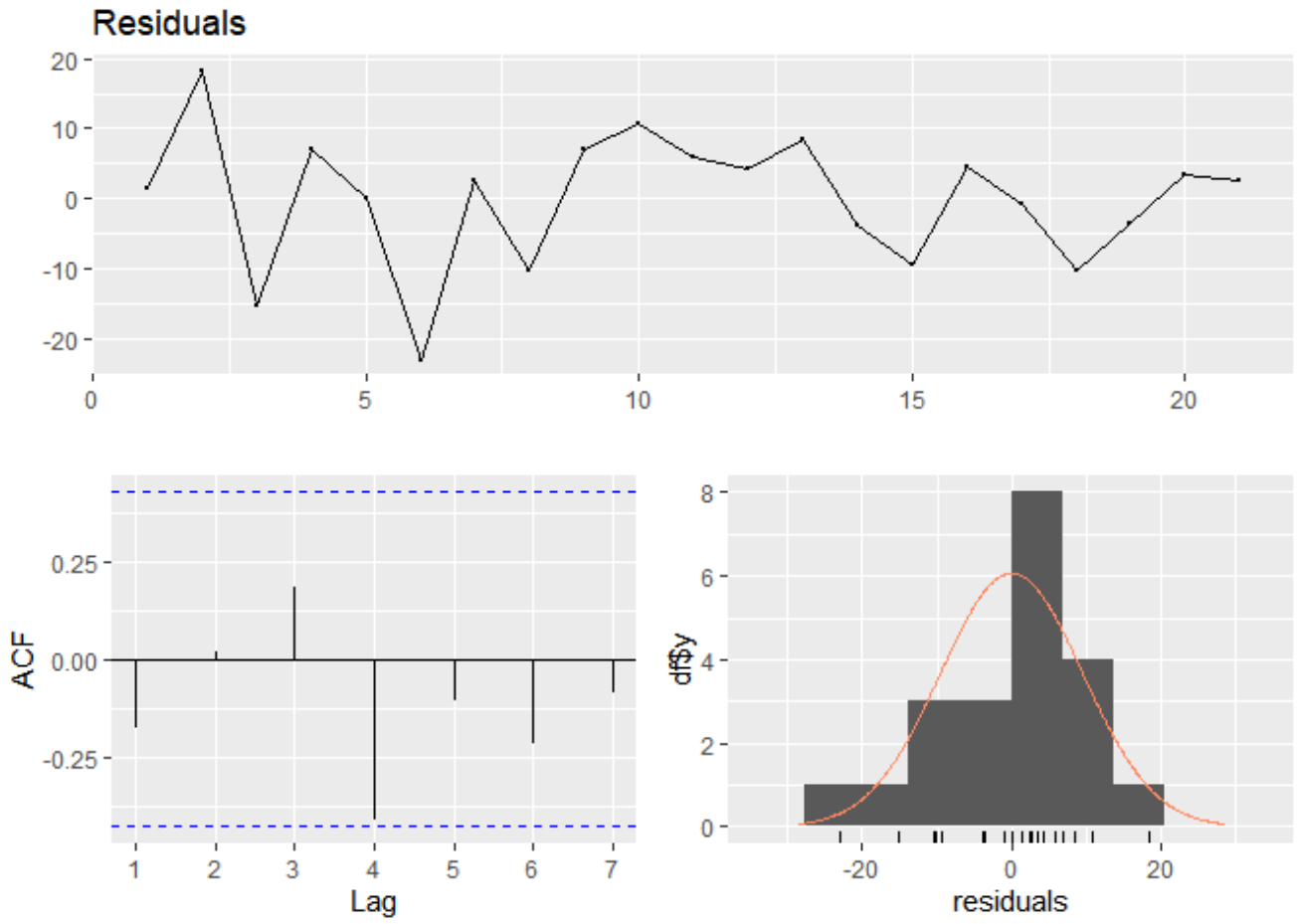

Figure 50

```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 6.4595, df = 4, p-value = 0.1674
##
## Model df: 0.   Total lags used: 4
```

```
bgtest(model2.3$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model2.3$model
## LM test = 0.6448, df = 1, p-value = 0.422
```

```
shapiro.test(model2.3$model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
```

```
## data:  model2.3$model$residuals
## W = 0.9602, p-value = 0.5202
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 50`. The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are not significant. Furthermore, from the Shapiro-Wilk test we can conclude that we do not reject null hypothesis of normality as p-value > 0.05.

```
VIF.model2.3<-vif(model2.3$model)
VIF.model2.3
```

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

```
VIF.model2.3>10
```

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

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the vif function, we can observe that there is an issue with multicollinearity as VIF values for z.t1 and z.t2 are greater than 10.

2.6.3 Koyck Distributed Lag Model

Similar to the previous models, we shall compare each predictor with the Koyck dlm and select the best with lowest aic, bic, and mase values.

```
#FFD & Temperature
model3.1<-koyckDlm(x=as.vector(ffd.temp.ts),y=as.vector(ffd.ffd.ts))
#FFD & Rainfall
model3.2<-koyckDlm(x=as.vector(ffd.rain.ts),y=as.vector(ffd.ffd.ts))
#FFD & Radiation
model3.3<-koyckDlm(x=as.vector(ffd.rad.ts),y=as.vector(ffd.ffd.ts))
#FFD & Relative Humidity
model3.4<-koyckDlm(x=as.vector(ffd.hum.ts),y=as.vector(ffd.ffd.ts))
#Compare
predictor<-c("Temperature","Rainfall","Radiation","Relative Humidity")
kdlm.mase<-MASE(model3.1,model3.2,model3.3,model3.4)
kdlm.aic<-c(AIC(model3.1),AIC(model3.2),AIC(model3.3),AIC(model3.4))
```

```
## [1] 242.2256
## [1] 269.7062
## [1] 237.873
## [1] 248.0706
```

```
kdlm.bic<-c(BIC(model3.1),BIC(model3.2),BIC(model3.3),BIC(model3.4))
```

```
## [1] 247.8304
## [1] 275.3109
## [1] 243.4778
## [1] 253.6754
```

```
kdlm<-data.frame(predictor,kdlm.mase,kdlm.aic,kdlm.bic)
arrange(kdlm,MASE)
```

```
##                  predictor  n      MASE kdlm.aic kdlm.bic
## model3.3         Radiation 30 0.7540611 237.8730 243.4778
## model3.1       Temperature 30 0.7968756 242.2256 247.8304
## model3.4 Relative Humidity 30 0.8390184 248.0706 253.6754
## model3.2          Rainfall 30 1.1546848 269.7062 275.3109
```

From the output above, we can observe that similar to previous models radiation has the lowest AIC, BIC, and MASE scores. Hence, we shall use this predictor for further analysis.

```
summary(model3.3)
```

```
##
## 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
```

By using the `summary` function, we can observe that in `model3.3` the lag weights of the predictors are not significant at the 5% statistical level as p-value > 0.05. Furthermore, there is a adjusted R-square value of -0.03775, which accounts for -3.77% of the variability in the data. Here we can also see that for the lags, positive lag coefficients show that the flowering is later, and negative lag coefficients shows earlier flowering. First lag indicates that flowering is early as Y.1=-0.118, which is negative value.

```
checkresiduals(model3.3$model$residuals)
```

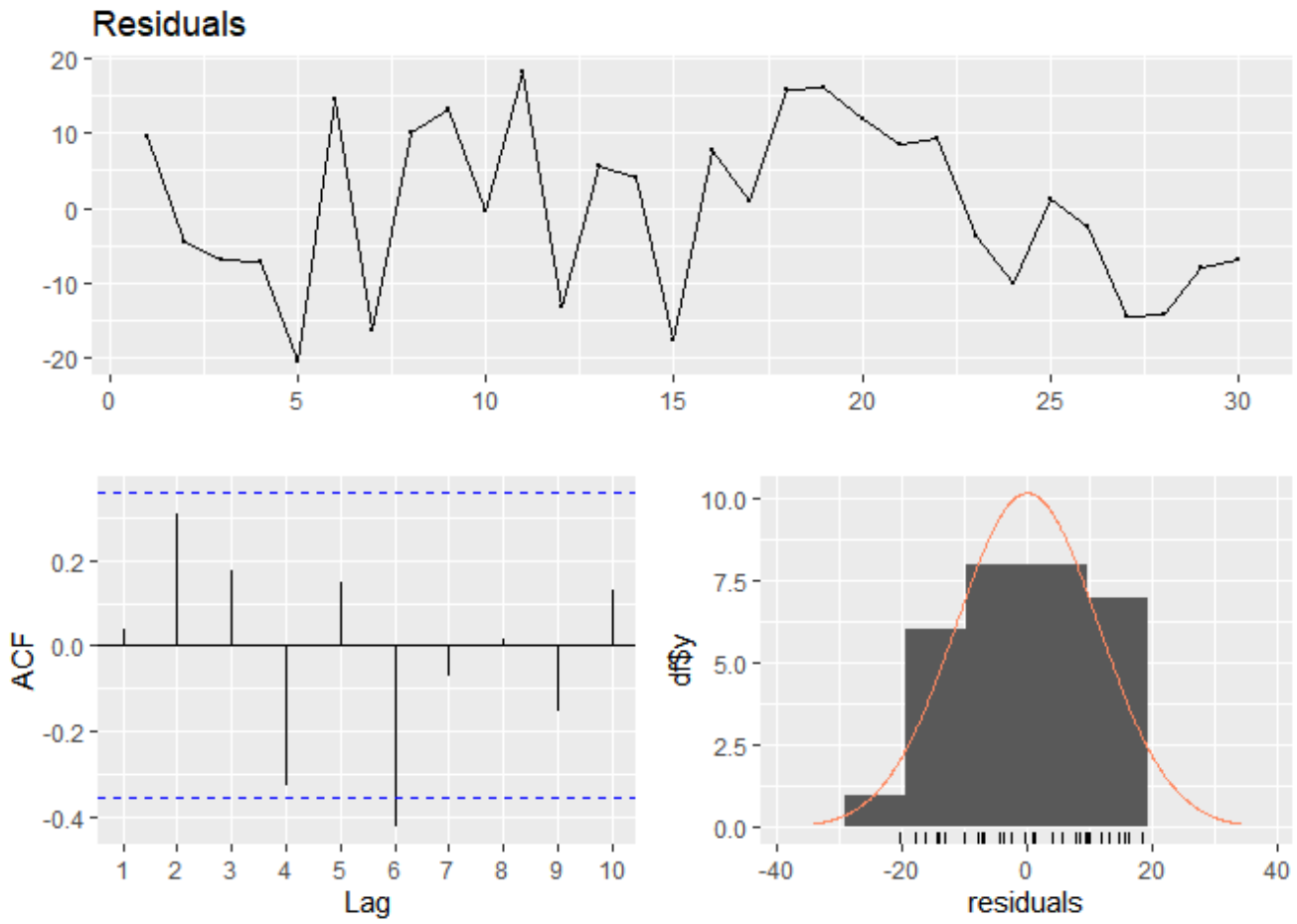

Figure 51

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

```
bgtest(model3.3$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model3.3$model
## LM test = 1.7914, df = 1, p-value = 0.1808
```

```
shapiro.test(model3.3$model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model3.3$model$residuals
## W = 0.95518, p-value = 0.2321
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 51`. The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are not significant. Furthermore, from the Shapiro-Wilk test we can conclude that we do not reject null hypothesis of normality as p-value > 0.05.

```
VIF.model3.3<-vif(model3.3$model)
VIF.model3.3
```

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

```
VIF.model3.3>10
```

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

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the vif function, we can observe that there is no issue with multicollinearity as VIF values for Y.1 and X.t are less than 10.

2.6.4 Autoregressive Distributed Lag Model

Similar to previous models, we will compare and find the best fit and predictors based on the aic, bic, and mase values. To perform the Autoregressive DLM, we will first create a loop function for p and q and use the `ardlDLM` function to identify the most appropriate number of lags for the DLM.

```
for (i in 1:5){
for(j in 1:5){
model4.1 = ardlDlm(x = as.vector(ffd.temp.ts), y = as.vector(ffd.ffd.ts), p = i , q = j)
cat("p =", i, "q =", j, "AIC =", AIC(model4.1$model), "BIC =", BIC(model4.1$model), "MASE =",
MASE(model4.1)$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
```

From the output above, we can observe that P=1 Q=5 has the lowest AIC values and BIC values, P=5 Q=5 has the lowest MASE values. Therefore we will compare the predictors with these 2 arguments.

```
#P=1 Q=5
#FFD & Temperature
model4.2.1<-ardlDlm(x=as.vector(ffd.temp.ts),y=as.vector(ffd.ffd.ts),p=1,q=5)
#FFD & Rainfall
model4.2.2<-ardlDlm(x=as.vector(ffd.rain.ts),y=as.vector(ffd.ffd.ts),p=1,q=5)
#FFD & Radiation
model4.2.3<-ardlDlm(x=as.vector(ffd.rad.ts),y=as.vector(ffd.ffd.ts),p=1,q=5)
#FFD & Relative Humidity
```

```
model4.2.4<-ardlDlm(x=as.vector(ffd.hum.ts),y=as.vector(ffd.ffd.ts),p=1,q=5)

#P=5 Q=5
#FFD & Temperature
model4.3.1<-ardlDlm(x=as.vector(ffd.temp.ts),y=as.vector(ffd.ffd.ts),p=5,q=5)
#FFD & Rainfall
model4.3.2<-ardlDlm(x=as.vector(ffd.rain.ts),y=as.vector(ffd.ffd.ts),p=5,q=5)
#FFD & Radiation
model4.3.3<-ardlDlm(x=as.vector(ffd.rad.ts),y=as.vector(ffd.ffd.ts),p=5,q=5)
#FFD & Relative Humidity
model4.3.4<-ardlDlm(x=as.vector(ffd.hum.ts),y=as.vector(ffd.ffd.ts),p=5,q=5)

predictor<-c("Temperature","Rainfall","Radiation","Relative Humidity")
adlm.mase1<-MASE(model4.2.1,model4.2.2,model4.2.3,model4.2.4)
adlm.aic1<-c(AIC(model4.2.1),AIC(model4.2.2),AIC(model4.2.3),AIC(model4.2.4))
```

```
## [1] 208.9515
## [1] 201.1726
## [1] 207.439
## [1] 206.1062
```

```
adlm.bic1<-c(BIC(model4.2.1),BIC(model4.2.2),BIC(model4.2.3),BIC(model4.2.4))
```

```
## [1] 220.2744
## [1] 212.4955
## [1] 218.7618
## [1] 217.429
```

```
adlm1<-data.frame(predictor,adlm.mase1,adlm.aic1,adlm.bic1)
colnames(adlm1)<-c("predictor","n","MASE","AIC","BIC")

adlm.mase2<-MASE(model4.3.1,model4.3.2,model4.3.3,model4.3.4)
adlm.aic2<-c(AIC(model4.3.1),AIC(model4.3.2),AIC(model4.3.3),AIC(model4.3.4))
```

```
## [1] 216.3587
## [1] 207.6224
## [1] 206.4543
## [1] 208.052
```

```
adlm.bic2<-c(BIC(model4.3.1),BIC(model4.3.2),BIC(model4.3.3),BIC(model4.3.4))
```

```
## [1] 232.7139
## [1] 223.9776
```

```
## [1] 222.8096
## [1] 224.4073
```

```
adlm2<-data.frame(predictor,adlm.mase2,adlm.aic2,adlm.bic2)
colnames(adlm2)<-c("predictor","n","MASE","AIC","BIC")

adlm3<-rbind(adlm1,adlm2)
arrange(adlm3,MASE)
```

```
##                    predictor  n      MASE      AIC      BIC
## model4.3.2          Rainfall 26 0.4143799 207.6224 223.9776
## model4.3.3         Radiation 26 0.4516374 206.4543 222.8096
## model4.2.2          Rainfall 26 0.4629303 201.1726 212.4955
## model4.3.4 Relative Humidity 26 0.4860361 208.0520 224.4073
## model4.2.4 Relative Humidity 26 0.5336178 206.1062 217.4290
## model4.3.1       Temperature 26 0.5540506 216.3587 232.7139
## model4.2.3         Radiation 26 0.5750332 207.4390 218.7618
## model4.2.1       Temperature 26 0.5812989 208.9515 220.2744
```

In the output above, we can observe that predictor rainfall (p5, q5) has the lowest mase value compared to the other predictors. However, predictor rainfall (p1, q5) has the lowest aic value and bic value. In our case for analysis, we shall select predictor rainfall (p5, q5) as our predictor for further analysis.

```
summary(model4.3.2)
```

```
##
## 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
```

By using the `summary` function, we can observe that in `model4.3.2` the lag weights of the predictors are not significant at the 5% statistical level as p-value > 0.05. Furthermore, there is a adjusted R-square value of 0.1968, which accounts for 19.68% of the variability in the data. Here we can also see that for the lags, positive lag coefficients show that the flowering is later, and negative lag coefficients shows earlier flowering. First lag indicates that flowering is early as X.1=-9.97784, which is negative value.

```
checkresiduals(model4.3.2$model$residuals)
```

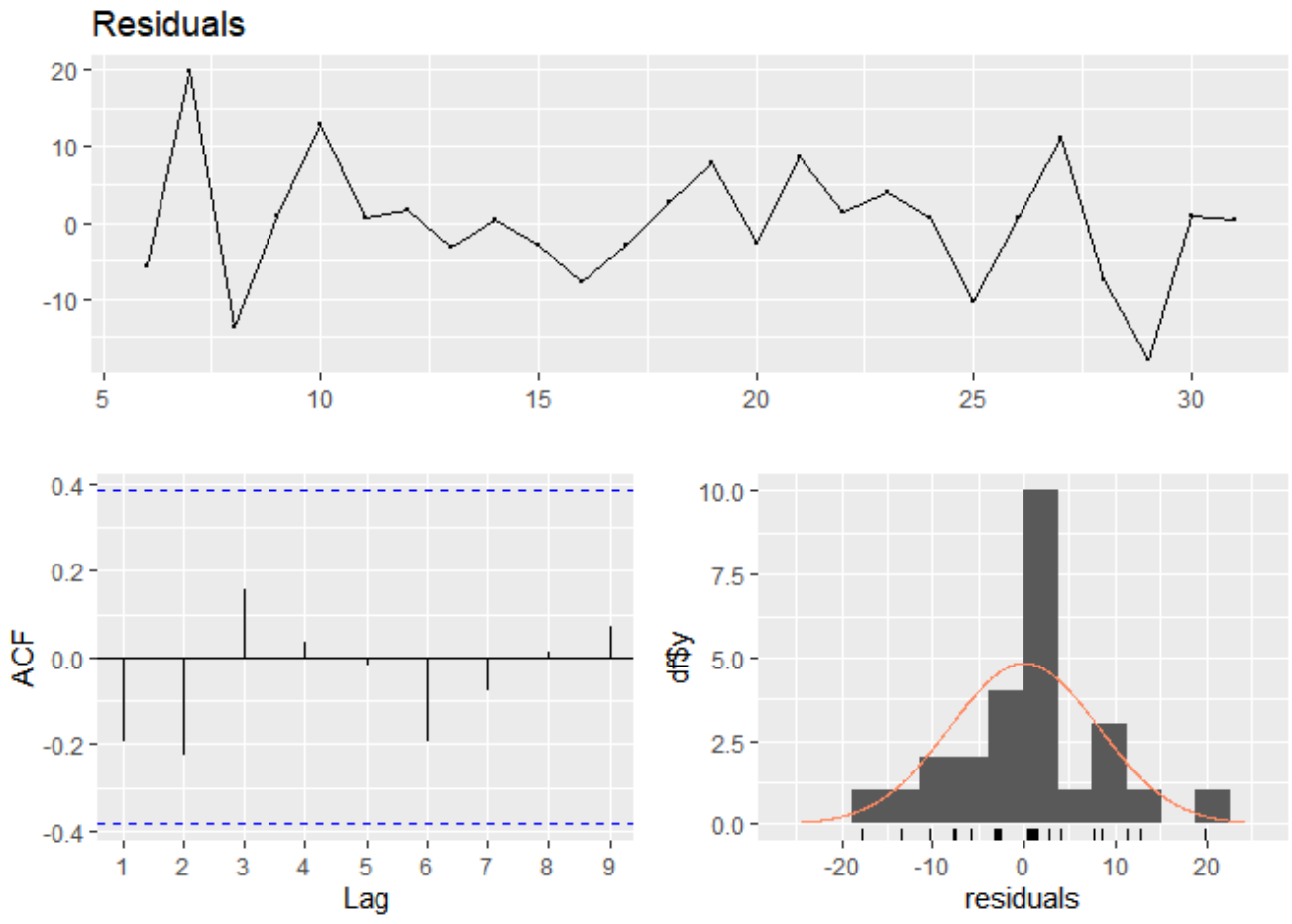

Figure 52

```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 3.4505, df = 5, p-value = 0.6309
```

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

```
bgtest(model4.3.2$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model4.3.2$model
## LM test = 11.927, df = 1, p-value = 0.0005534
```

```
shapiro.test(model4.3.2$model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model4.3.2$model$residuals
## W = 0.96736, p-value = 0.5561
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 52`. The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are not significant. Furthermore, from the Shapiro-Wilk test we can conclude that we do not reject null hypothesis of normality as p-value > 0.05.

```
VIF.model4.3.2<-vif(model4.3.2$model)
VIF.model4.3.2
```

```
##       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
```

```
VIF.model4.3.2>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
```

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the vif function, we can observe that there is no issue with multicollinearity as all VIF values are less than 10.

2.7.0 Dynamic Lag Models

We will fit the dynamic linear models as the model is for assessing the effect of the intervention on the time series. We will compute the following dynamic lag models and choose the best one with the lowest mase value for further analysis.

```
Y.t=log(ffd.ffd.ts)
T=7
P.t=1*(seq(ffd.ffd.ts)==T)
P.t.1=Lag(P.t,+1)

model5.1 = dynlm(Y.t ~ L(Y.t , k = 1 ) + P.t.1 + P.t + trend(Y.t))
model5.2 = dynlm(Y.t ~ L(Y.t , k = 1 ) + P.t.1 + P.t + L(Y.t , k = 2 )
                 + trend(Y.t))

model5.mase<-MASE(lm(model5.1),lm(model5.2))
arrange(model5.mase,MASE)
```

```
##                n      MASE
## lm(model5.2) 29 0.6361339
## lm(model5.1) 30 0.6757785
```

Here in this output we can see that `model5.2` has the best MASE value so we will use this model for further analysis.

```
summary(model5.2)
```

```
##
## Time series regression with "ts" data:
## Start = 1986, End = 2014
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, k = 1) + P.t.1 + P.t + L(Y.t, k = 2) +
##     trend(Y.t))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.06413 -0.01826  0.00000  0.02176  0.05063
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.649e+00  1.659e+00   2.200   0.0381 *
```

```
## L(Y.t, k = 1)  1.109e-01  1.992e-01   0.557   0.5831
## P.t.1         -4.988e-02  4.092e-02  -1.219   0.2352
## P.t            6.788e-02  4.042e-02   1.679   0.1066
## L(Y.t, k = 2)  2.532e-01  1.949e-01   1.299   0.2067
## trend(Y.t)     2.008e-05  8.469e-04   0.024   0.9813
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03594 on 23 degrees of freedom
## Multiple R-squared:  0.237,  Adjusted R-squared:  0.07111
## F-statistic: 1.429 on 5 and 23 DF,  p-value: 0.2515
```

By using the `summary` function, we can observe that in `model5.2` the lag weights of the predictors are not significant at the 5% statistical level as p-value > 0.05. Furthermore, there is a adjusted R-square value of 0.0711, which accounts for 7.11% of the variability in the data. Here we can also see that for the lags, positive lag coefficients show that the flowering is later, and negative lag coefficients shows earlier flowering. First lag indicates that flowering is early as there is positive value.

```
checkresiduals(model5.2)
```

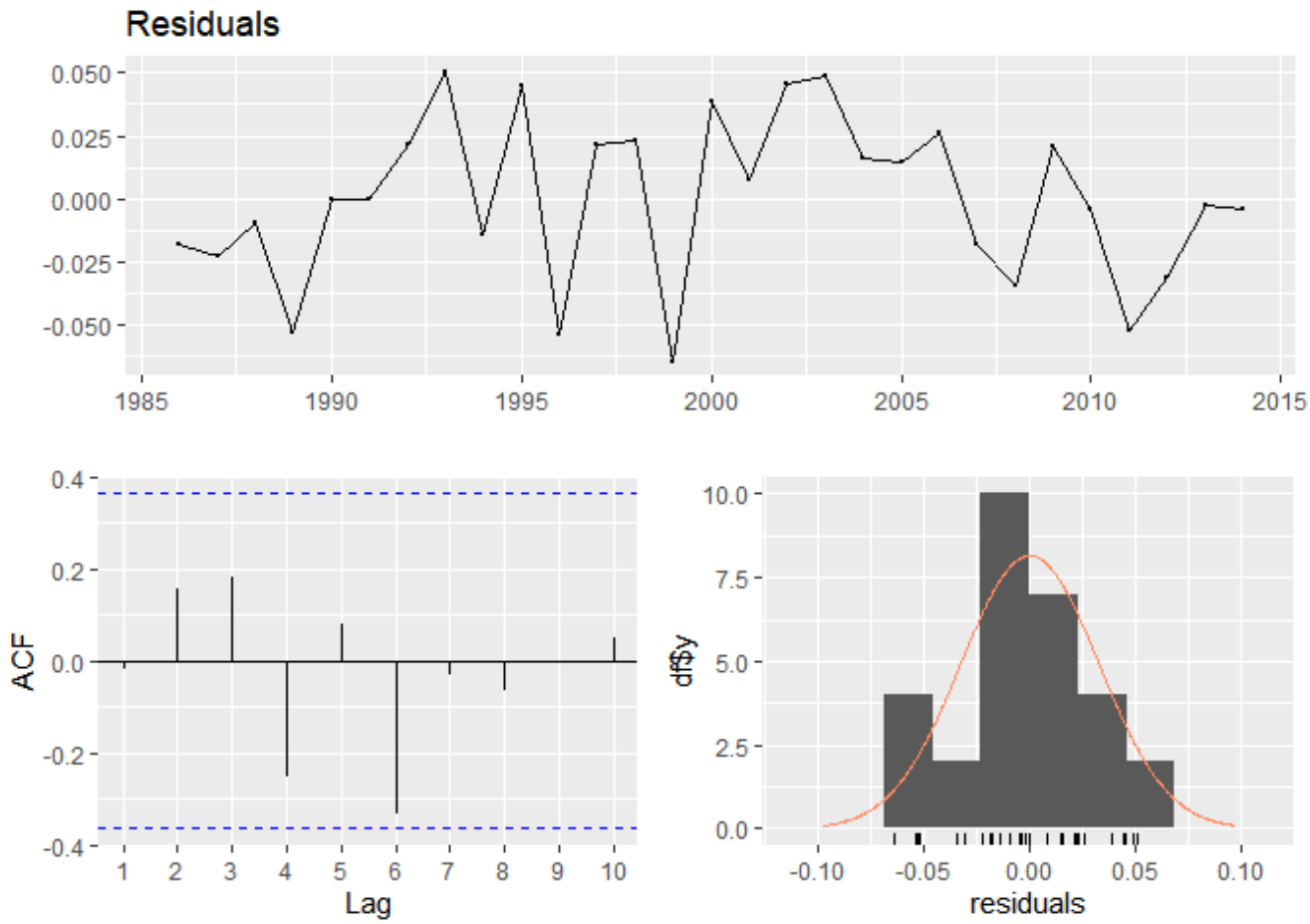

Figure 53

```
##
##  Breusch-Godfrey test for serial correlation of order up to 9
##
```

```
## data:  Residuals
## LM test = 10.583, df = 9, p-value = 0.3054
```

```
bgtest(model5.2$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model5.2$model
## LM test = 0.056977, df = 1, p-value = 0.8113
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 52`. The Breush-Godfrey test is also conducted here using the `bgtest` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are not significant.

```
VIF.model5.2<-vif(model5.2)
VIF.model5.2
```

```
## L(Y.t, k = 1)         P.t.1           P.t L(Y.t, k = 2)    trend(Y.t)
##      1.219052      1.251748      1.221734      1.152660      1.127447
```

```
VIF.model5.2>10
```

```
## L(Y.t, k = 1)         P.t.1           P.t L(Y.t, k = 2)    trend(Y.t)
##         FALSE         FALSE         FALSE         FALSE         FALSE
```

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the vif function, we can observe that there is no issue with multicollinearity as all VIF values are less than 10.

2.8.0 Exponential Smoothing Models

We will now test the time series data by fitting holt's linear trend methd. We will select models with lowest RMSE values.

```
#Software Estimated Alpha and Beta
model6.1<-holt(ffd.ffd.ts,initial="simple",h=4)
#Brown's double exponential smoothing
model6.2<-holt(ffd.ffd.ts,alpha=0.4826,beta=0.6284,initial="simple",h=4)
#Simple exponential smoothing with drift
```

```
model6.3<-holt(ffd.ffd.ts,beta=0,initial="simple",h=4)
#Exponential trend
model6.4<-holt(ffd.ffd.ts,initial="simple",exponential=TRUE,h=4)

esm.model<-c("model6.1","model6.2","model6.3","model6.4")
esm.rmse<-c(accuracy(model6.1)[,2],accuracy(model6.2)[,2],accuracy(model6.3)[,2],
            accuracy(model6.4)[,2])
esm.mase<-c(accuracy(model6.1)[,6],accuracy(model6.2)[,6],accuracy(model6.3)[,6],
            accuracy(model6.4)[,6])
esm.compare<-data.frame(esm.model,esm.rmse,esm.mase)
arrange(esm.compare,esm.rmse)
```

```
##   esm.model esm.rmse  esm.mase
## 1  model6.1 14.90577 0.9774001
## 2  model6.4 14.97084 0.9805558
## 3  model6.2 15.21833 0.9615677
## 4  model6.3 20.40488 1.3374803
```

The smallest RMSE `model6.1` was with Holt's linear trend model when constants are estimated from the data. Even though `model6.2` has the smallest MASE values, we will select the lowest rmse value for this analysis.

```
summary(model6.1)
```

```
##
## Forecast method: Holt's method
##
## Model Information:
## Holt's method
##
## Call:
##  holt(y = ffd.ffd.ts, h = 4, initial = "simple")
##
##   Smoothing parameters:
##     alpha = 0.4826
##     beta  = 0.3142
##
##   Initial states:
##     l = 310
##     b = 12
##
##   sigma:  14.9058
## Error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -2.705196 14.90577 12.57588 -0.9914293 4.060386 0.9774001
##                    ACF1
## Training set -0.2147934
##
## Forecasts:
```

```
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2015       301.0218 281.9193 320.1243 271.8071 330.2366
## 2016       300.3078 275.8834 324.7321 262.9540 337.6615
## 2017       299.5937 267.2381 331.9493 250.1101 349.0773
## 2018       298.8796 256.5955 341.1637 234.2116 363.5476
```

Above is the summary output for `model6.1` . We will now check for residuals in the `model6.1` .

```
checkresiduals(model6.1)
```

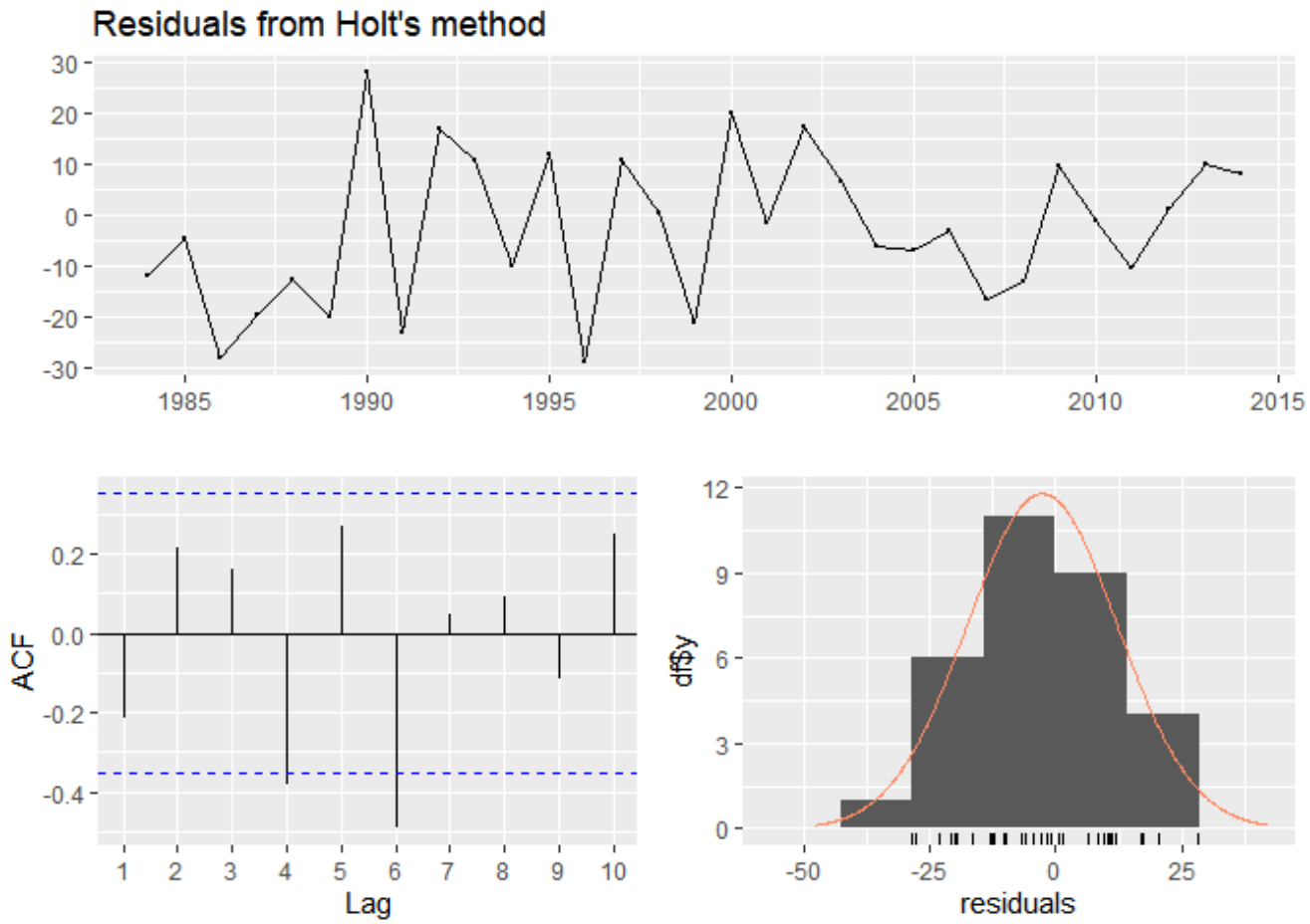

Figure 54

```
##
##  Ljung-Box test
##
## data:  Residuals from Holt's method
## Q* = 22.456, df = 6, p-value = 0.001001
##
## Model df: 0.   Total lags used: 6
```

```
shapiro.test(residuals(model6.1$model))
```

```
##
##  Shapiro-Wilk normality test
##
```

```
## data:  residuals(model6.1$model)
## W = 0.97704, p-value = 0.7264
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 54`. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are not significant. Furthermore, from the Shapiro-Wilk test we can conclude that we do not reject null hypothesis of normality as p-value > 0.05.

2.9.0 State-Space Models

In this section we will first use `autoets` function to determine the best state-space models for FFD as given by the software.

```
autoets<-ets(ffd.ffd.ts)
autoets$method
```

```
## [1] "ETS(A,N,N)"
```

In the output above, we can see that it is recommended that we use the additive error, no trend, and no seasonality for analysis.

2.9.1 A,N,N

```
model7.1<-ets(ffd.ffd.ts,model="ANN")
summary(model7.1)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = ffd.ffd.ts, 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
```

The model shows the summary output for `model7.1` which is an additive error model. Here we can observe that this model has a MASE value of 0.7584965 and AIC score of 262.0960.

```
checkresiduals(model7.1)
```

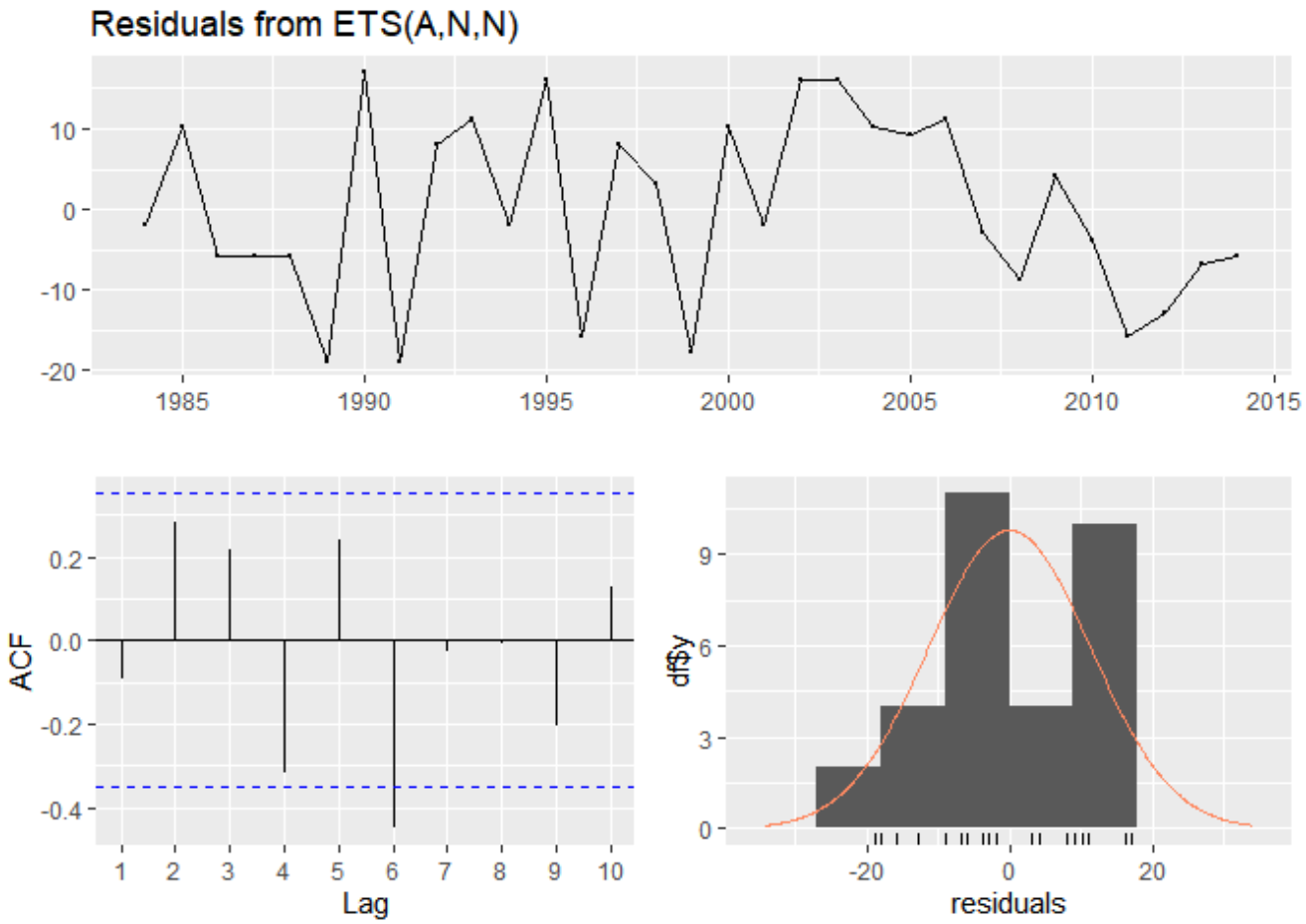

Figure 55

```
##
##  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
```

```
shapiro.test(model7.1$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
```

```
## data:  model7.1$residuals
## W = 0.93492, p-value = 0.05978
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 55`. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are not significant. Furthermore, from the Shapiro-Wilk test we can conclude that we do not reject null hypothesis of normality as p-value > 0.05.

2.10.0 Model Comparison and Selection

With our models selected from each section, we will now compare each models and select the best for further analysis.

```
comp.tsrm.models<-c("model1.3","model2.3","model3.3","model4.3.2")
comp.tsrm.mase<-MASE(model1.3,model2.3,model3.3,model4.3.2)
comp.tsrm<-data.frame(comp.tsrm.models,comp.tsrm.mase)
comp.tsrm<-comp.tsrm[,-2]
colnames(comp.tsrm)<-c("Model","MASE")

comp.ss.models<-c("model5.2","model6.1","model7.1")
comp.ss.mase<-c(accuracy(model5.2)[,6],accuracy(model6.1)[,6],
                accuracy(model7.1)[,6])
comp.ss<-data.frame(comp.ss.models,comp.ss.mase)
colnames(comp.ss)<-c("Model","MASE")

comp<-rbind(comp.tsrm,comp.ss)
rownames(comp)<-NULL
arrange(comp,MASE)
```

```
##        Model      MASE
## 1   model1.3 0.2319041
## 2 model4.3.2 0.4143799
## 3   model2.3 0.6355363
## 4   model5.2 0.6361339
## 5   model3.3 0.7540611
## 6   model7.1 0.7584965
## 7   model6.1 0.9774001
```

Based on the lowest MASE values, we can now choose `model1.3` and `model4.3.2` for further analysis in the next section.

2.11.0 Forecasting

We will first forecast the data for `model1.3` which is a finite DLM with radiation predictor. Here we will also use the covariate values provided for point forecasting.

```
ffd.covar[,4]
```

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

```
frc.model1.3<-forecast(model=model1.3,
                       x=c(14.60,14.56,14.79,14.79),
                       h=4)$forecast
frc.model1.3
```

```
## [1] 302.7963 296.9306 281.4089 312.2780
```

In the output above, we are able to see the point forecast for FFD based on `model1.3.` However, due to errors in the package, we are unable to output the 95% confidence intervals for `model1.3` . We will now plot the four year ahead forecast plots.

```
y.extended<-c(ffd$FFD,frc.model1.3)
plot(ts(y.extended,start=1984),type="o",col="red",ylim=c(250,400),
     xlab="Year",ylab="FFD",main="FFD 4 Year Forecast (Finite DLM)")
lines(ts(as.vector(ffd$FFD),start=1984),col="black",type="o")
legend("topleft",lty=1,pch=1,cex=0.7,col=c("black","red"),
       c("FFD Data","Finite DLM Forecast"))
```

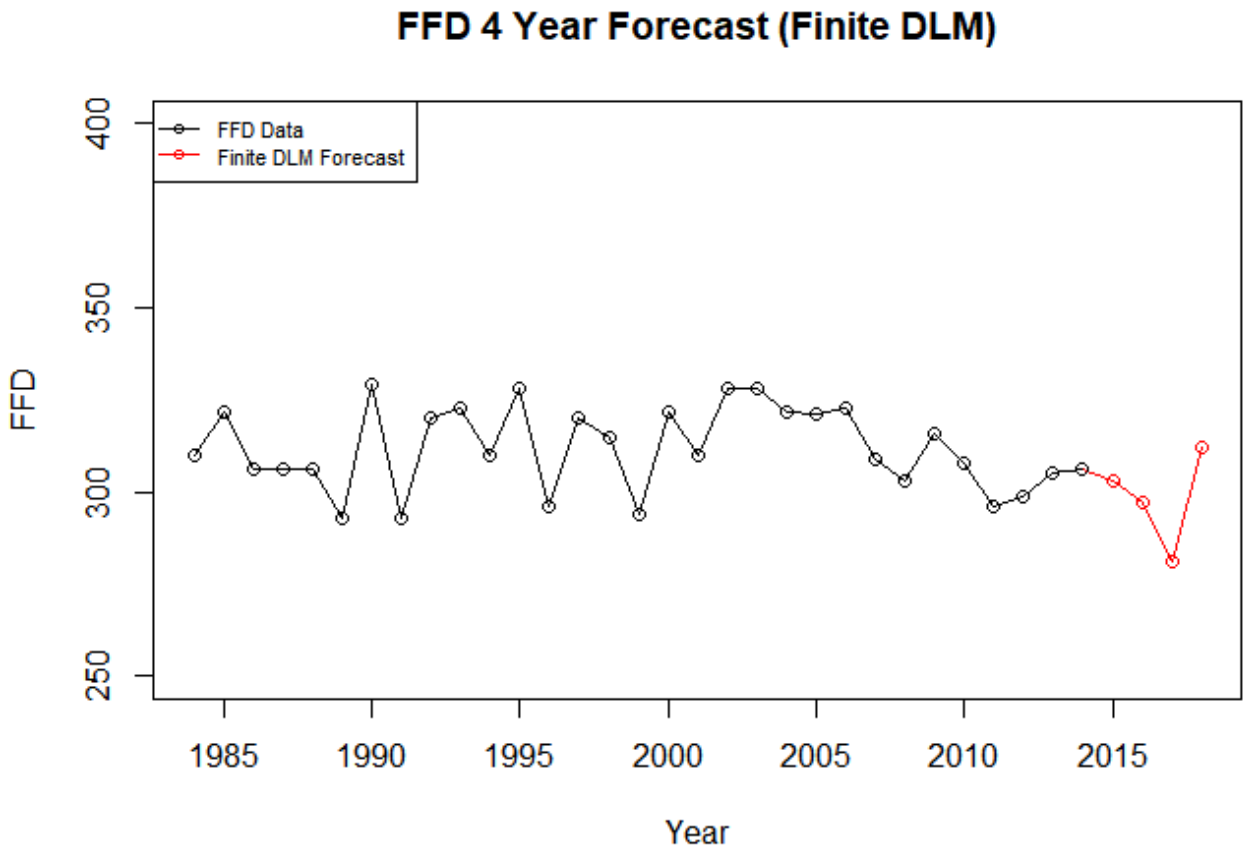

Figure 56

From `Figure 56` above, we can see that finite dlm fit and the plots for the next four years prediction (2015-2018) shows that there will be a intervention around 2017. Next we will forecast the data for `model4.3.2` which is a Autoregressive DLM with rainfall as predictor for (P5 Q5).

```
ffd.covar[,3]
```

```
## # A tibble: 4 × 1
##   Rainfall
##      <dbl>
## 1     2.27
## 2     2.38
## 3     2.26
## 4     2.27
```

```
frc.model4.3.2<-forecast(model=model4.3.2,
                         x=c(2.27,2.38,2.26,2.27),
                         h=4,
                         interval=TRUE)$forecast
frc.model4.3.2
```

```
##     95% LB Forecast   95% UB
## 1 302.0332 322.7660 344.6669
## 2 302.4627 322.4705 344.2289
```

```
## 3 298.8548 326.2953 349.6524
## 4 305.0863 327.3587 351.5964
```

In the forecast output above, we can see the 95% lower bound, point forecast and 95% upper bound forecast data for FFD. We will now plot the model against the original data.

```
y.extended2<-c(ffd$FFD,frc.model4.3.2$Forecast)
plot(ts(y.extended2,start=1984),type="o",col="red",ylim=c(250,400),
     xlab="Year",ylab="FFD",main="FFD 4 Year Forecast (Autoregressive DLM)")
lines(ts(as.vector(ffd$FFD),start=1984),col="black",type="o")
legend("topleft",lty=1,pch=1,cex=0.7,col=c("black","red"),
       c("FFD Data","Autoregressive DLM Forecast"))
```

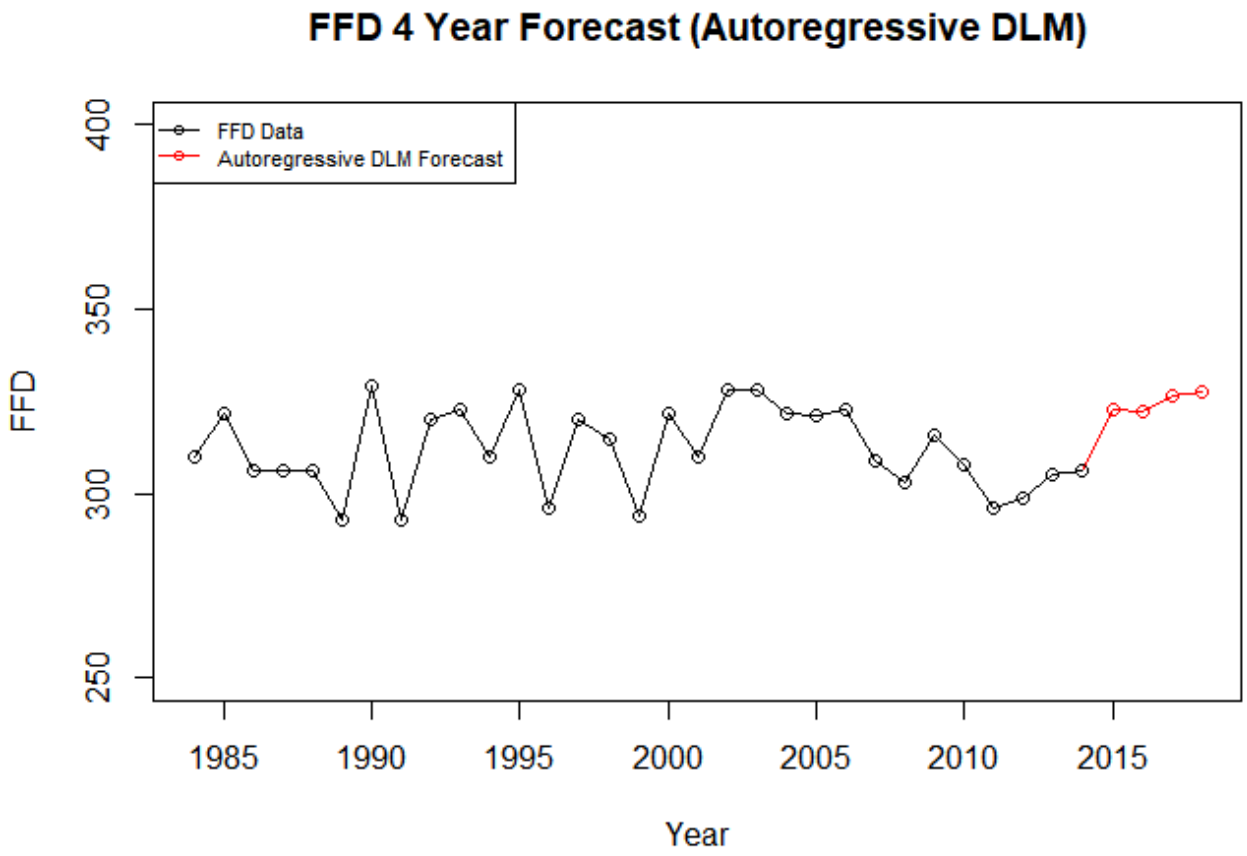

Figure 57

From `Figure 57` above, we can see that autoregressive dlm fit and the plots for the next four years prediction (2015-2018). Interestingly, compared to Finite DLM forecast, the point forecast shows that the data would increase over the next four years.

2.12.0 Conclusion

In conclusion in task 2 we have successfully created forecast plots for FFD for the best 4 years ahead forecast. We have found that the model with the lowest MASE values are Finite DLM and Autoregressive DLM.

3.0.0 Task 3 (A)

3.1.0 Introduction

The aim of this investigation for task 3 is to carry out analysis based on univariate climate regressors and forecast RBO three years ahead for the best model selected. The data set provided `rbo.csv` contains observation data for variables year, rbo, temperature, rainfall, radiation, and relative humidity from 1984 to 2014. To perform these investigations, we will first prep the data for analysis and do further testing through various analysis methods.

3.2.0 Data Description and Preprocessing

The dataset `RBO.csv` is used for analysis in task 1. Hence, we will import the data set using the `read_csv` function.

```
rbo<-read_csv("RBO .csv")
```

```
## New names:
## Rows: 36 Columns: 14
## ── Column specification
## ──────────────────────────────────────────────────────── Delimiter: "," dbl
## (6): Year, RBO, Temperature, Rainfall, Radiation, RelHumidity lgl (8): ...7,
## ...8, ...9, ...10, ...11, ...12, ...13, ...14
## ℹ Use `spec()` to retrieve the full column specification for this data. ℹ
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## • `` -> `...7`
## • `` -> `...8`
## • `` -> `...9`
## • `` -> `...10`
## • `` -> `...11`
## • `` -> `...12`
## • `` -> `...13`
## • `` -> `...14`
```

```
rbo.covar<-read_csv("Covariate x-values for Task 3 .csv")
```

```
## Rows: 6 Columns: 5
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## dbl (5): Year, Temperature, Rainfall, Radiation, RelHumidity
##
```

```
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(rbo)
```

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

```
head(rbo.covar)
```

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

```
rbo<-rbo[-(32:36),-(7:14)]
rbo.covar<-rbo.covar[-(5:6),]

class(rbo)
```

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

```
str(rbo)
```

```
## tibble [31 × 6] (S3: tbl_df/tbl/data.frame)
##  $ Year       : num [1:31] 1984 1985 1986 1987 1988 ...
##  $ RBO        : num [1:31] 0.755 0.741 0.842 0.748 0.798 ...
##  $ Temperature: num [1:31] 18.7 19.3 18.6 19.1 20.4 ...
##  $ Rainfall   : num [1:31] 2.49 2.48 2.42 2.32 2.47 ...
```

```
##  $ Radiation  : num [1:31] 14.9 14.7 14.5 14.7 14.7 ...
##  $ RelHumidity: num [1:31] 54.6 55 55 53.9 53.1 ...
```

```
head(rbo)
```

```
## # A tibble: 6 × 6
##    Year   RBO Temperature Rainfall Radiation RelHumidity
##   <dbl> <dbl>       <dbl>    <dbl>     <dbl>       <dbl>
## 1  1984 0.755        18.7     2.49      14.9        54.6
## 2  1985 0.741        19.3     2.48      14.7        55.0
## 3  1986 0.842        18.6     2.42      14.5        55.0
## 4  1987 0.748        19.1     2.32      14.7        53.9
## 5  1988 0.798        20.4     2.47      14.7        53.1
## 6  1989 0.794        19.6     2.74      14.8        55.4
```

```
class(rbo.covar)
```

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

```
str(rbo.covar)
```

```
## tibble [4 × 5] (S3: tbl_df/tbl/data.frame)
##  $ Year       : num [1:4] 2015 2016 2017 2018
##  $ Temperature: num [1:4] 20.7 20.5 20.5 20.6
##  $ Rainfall   : num [1:4] 2.27 2.38 2.26 2.27
##  $ Radiation  : num [1:4] 14.6 14.6 14.8 14.8
##  $ RelHumidity: num [1:4] 52.2 52.9 52.6 52.5
```

```
head(rbo.covar)
```

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

Here we can see that the class, structure, and head of data is what we want. However, the dataset contains many na values which then is dealt by subsetting the data. Next, using the `ts` function we will convert the data into a time series data, with each variable in it's own time series vector.

```
rbo.ts<-ts(rbo[,2:6],start=c(1984,1),frequency=1)
rbo.rbo.ts<-ts(rbo$RBO,start=c(1984,1),frequency=1)
rbo.temp.ts<-ts(rbo$Temperature,start=c(1984,1),frequency=1)
rbo.rain.ts<-ts(rbo$Rainfall,start=c(1984,1),frequency=1)
rbo.rad.ts<-ts(rbo$Radiation,start=c(1984,1),frequency=1)
rbo.hum.ts<-ts(rbo$RelHumidity,start=c(1984,1),frequency=1)
rbo.covar.ts<-ts(rbo.covar[,2:5],start=c(2015,1),frequency=1)
```

3.3.0 Data Visualisation

Here we will plot each time series objects as a graph to gain a further understanding of the data through visual inspection. We will use the `plot` function to plot the graphs.

3.3.1 RBO

```
plot(rbo.rbo.ts,main="Time Series Plot for RBO Series",
     ylab="RBO",xlab="Year",type="o")
```

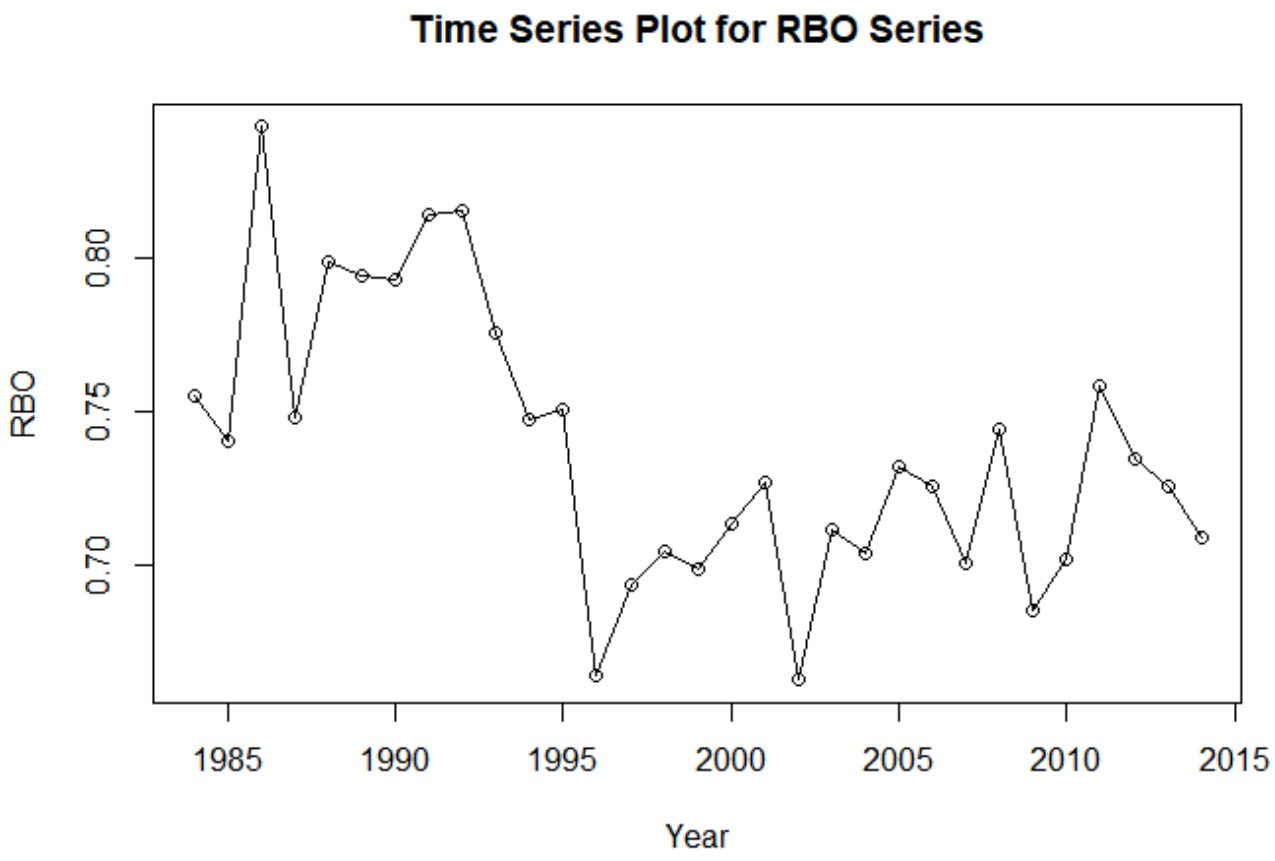

Figure 58

By `Figure 58`, which shows the time series plot of rbo series, we can make the following observations:

- The series has apparent downward trend.

- There are no presence of seasonality.
- Changing variance and moving average behavior can be observed
- Multiple intervention points can be observed in the plot.

```
acf(rbo.rbo.ts, lag.max = 48, main="Sample ACF Plot for RBO Series")
```

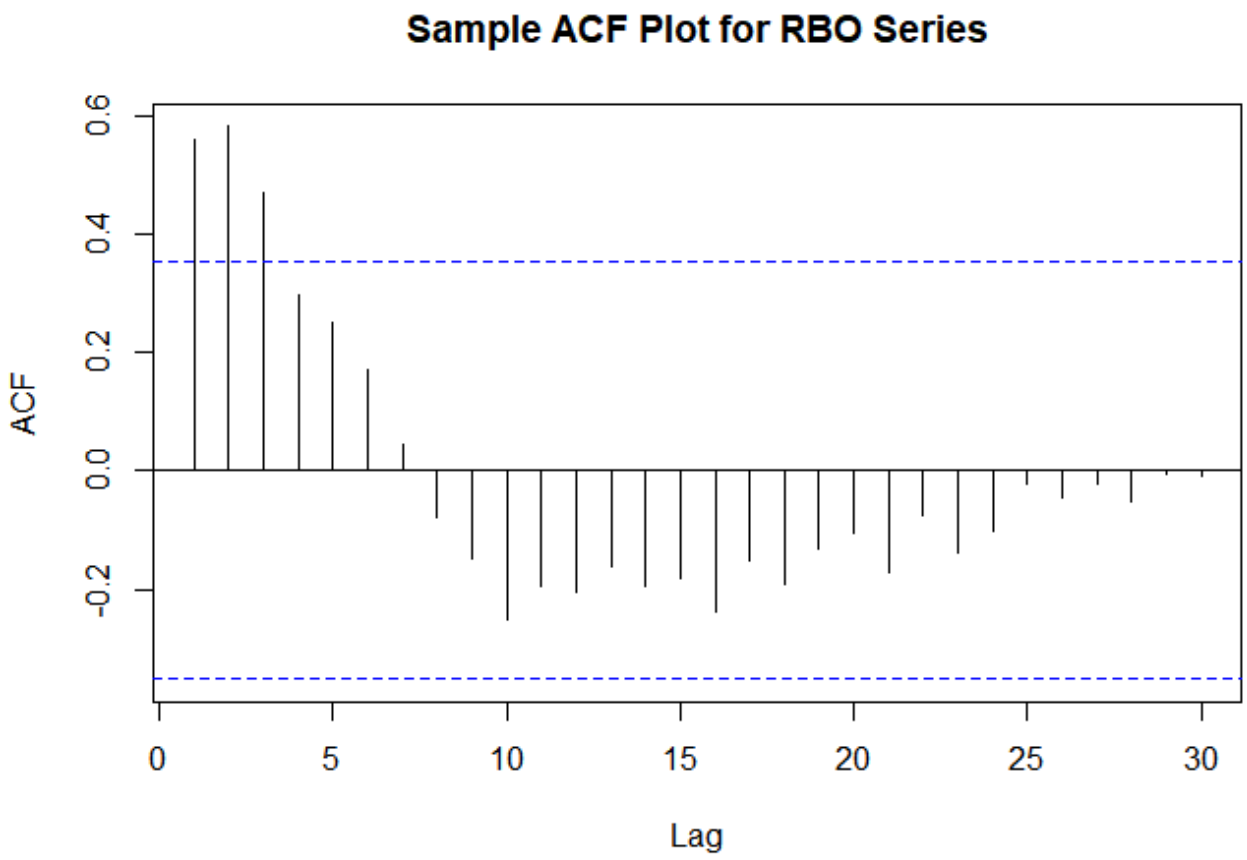

Figure 59

`Figure 59` shows the sample ACF plot for RBO series. Here we can observe that first lag is significant which suggests non-stationary in series.

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

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

In the above output, we used the ADF test for the radiation series. In this approach, we can conclude that the radiation series is non-stationary at 5% level of significance as p value is greater than 0.05.

3.3.2 Temperature

```
plot(rbo.temp.ts,main="Time Series Plot for Temperature Series",
     ylab="Temperature",xlab="Year",type="o")
```

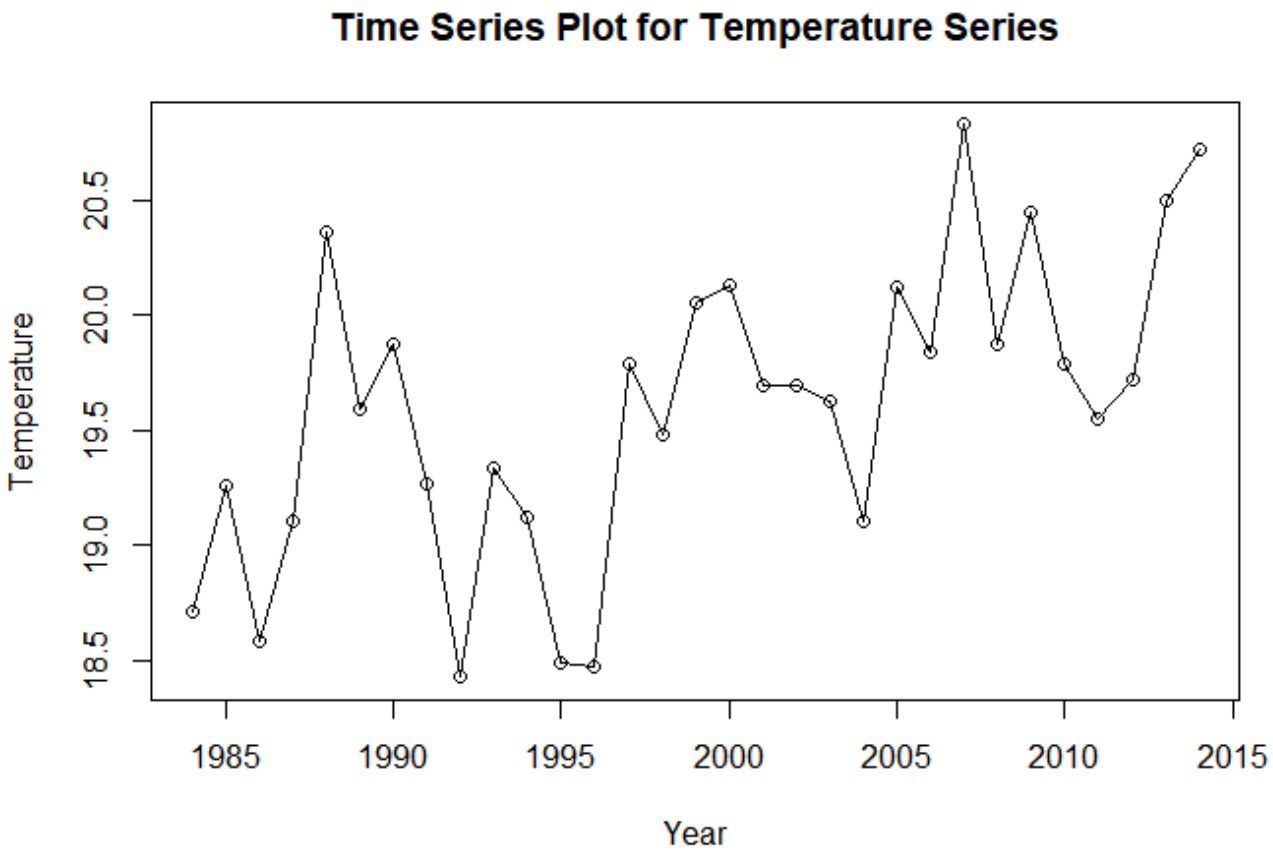

Figure 60

By `Figure 60`, which shows the time series plot of temperature series, we can make the following observations:

- The series has apparent upward trend.
- There are no presence of seasonality.
- Changing variance and moving average behavior can be observed
- No intervention points can be observed in the plot.

```
acf(rbo.temp.ts, lag.max = 48, main="Sample ACF Plot for Temperature Series")
```

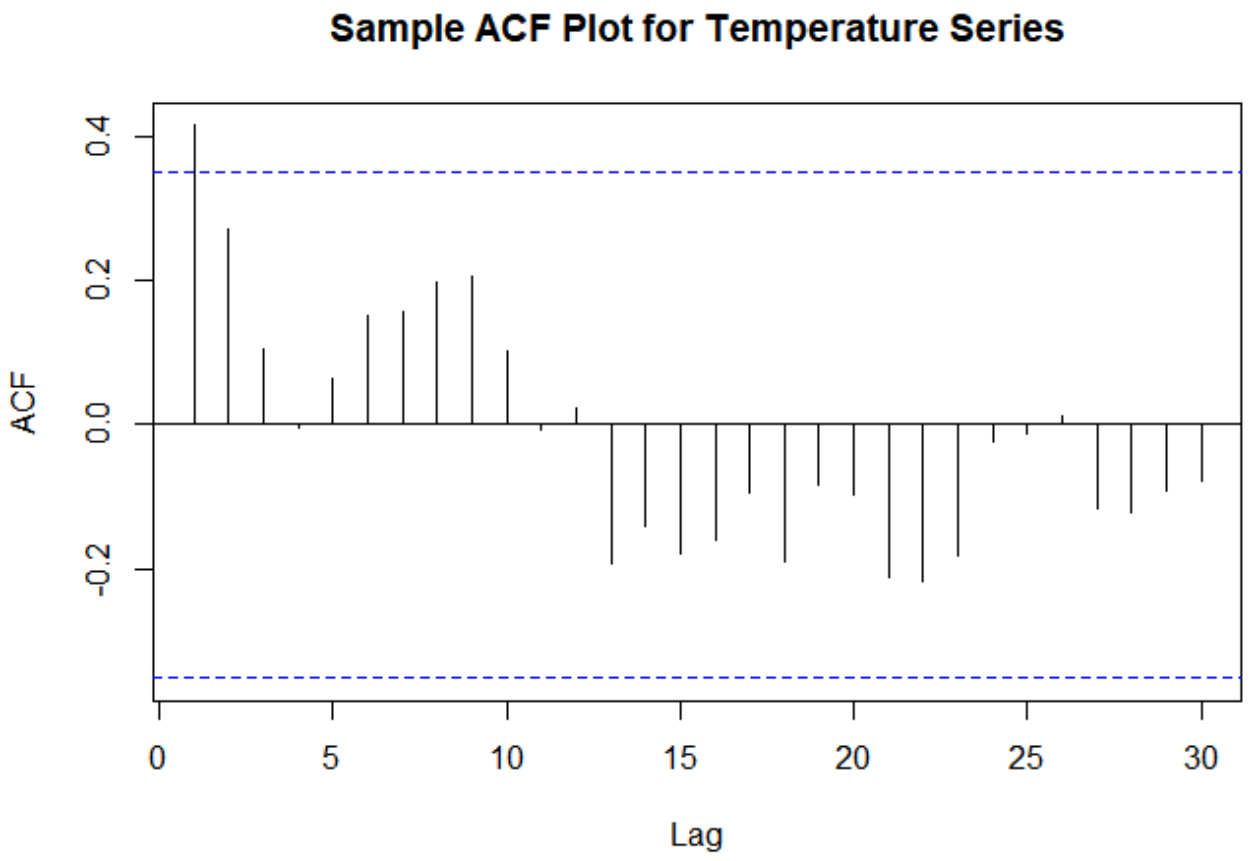

Figure 61

`Figure 61` shows the sample ACF plot for temperature series. Here we can observe that first lag is significant which suggests non-stationary in series.

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

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

In the above output, we used the ADF test for the radiation series. In this approach, we can conclude that the radiation series is non-stationary at 5% level of significance as p value is greater than 0.05.

3.3.3 Rainfall

```
plot(rbo.rain.ts,main="Time Series Plot for Rainfall Series",
     ylab="Rainfall",xlab="Year",type="o")
```

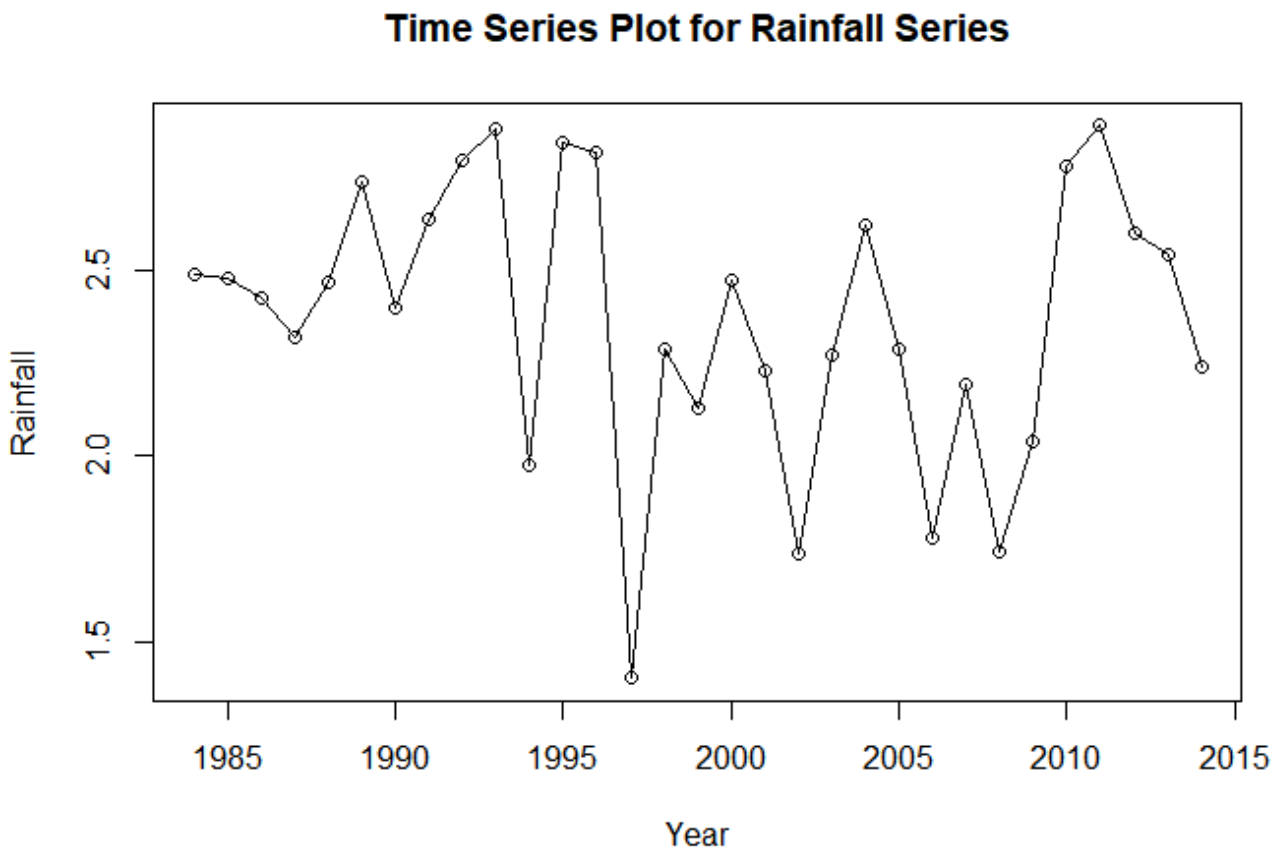

Figure 62

By `Figure 62`, which shows the time series plot of temperature series, we can make the following observations:

- The series has no apparent trends.
- There are no presence of seasonality.
- Changing variance and moving average behavior can be observed
- Intervention points can be observed in the plot at 1994 and 1997.

```
acf(rbo.rain.ts, lag.max = 48, main="Sample ACF Plot for Rainfall Series")
```

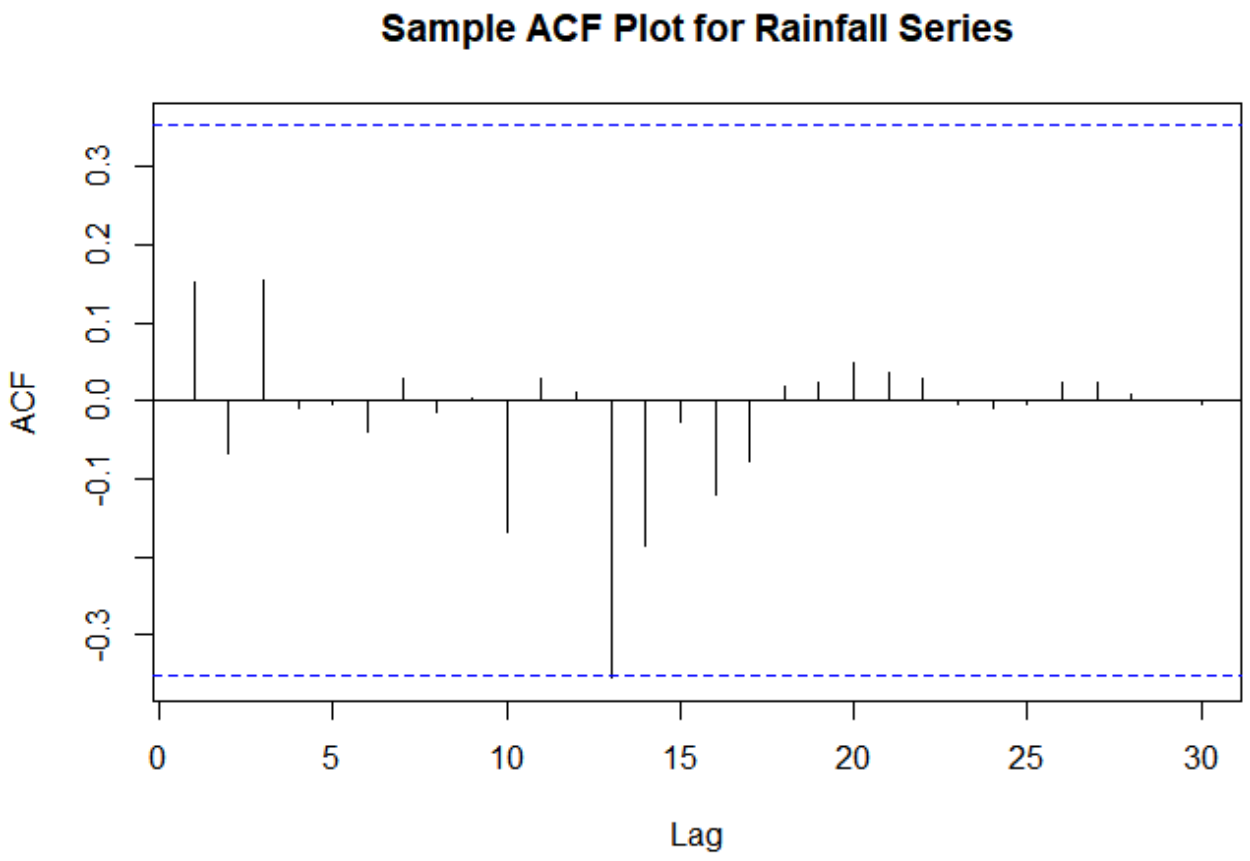

Figure 63

`Figure 63` shows the sample ACF plot for rainfall series. Here we can observe that first lag is not significant which suggests stationary in series.

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

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

In the above output, we used the ADF test for the radiation series. In this approach, we can conclude that the radiation series is stationary at 5% level of significance as p value is less than 0.05.

3.3.4 Radiation

```
plot(rbo.rad.ts,main="Time Series Plot for Radiation Series",
     ylab="Radiation",xlab="Year",type="o")
```

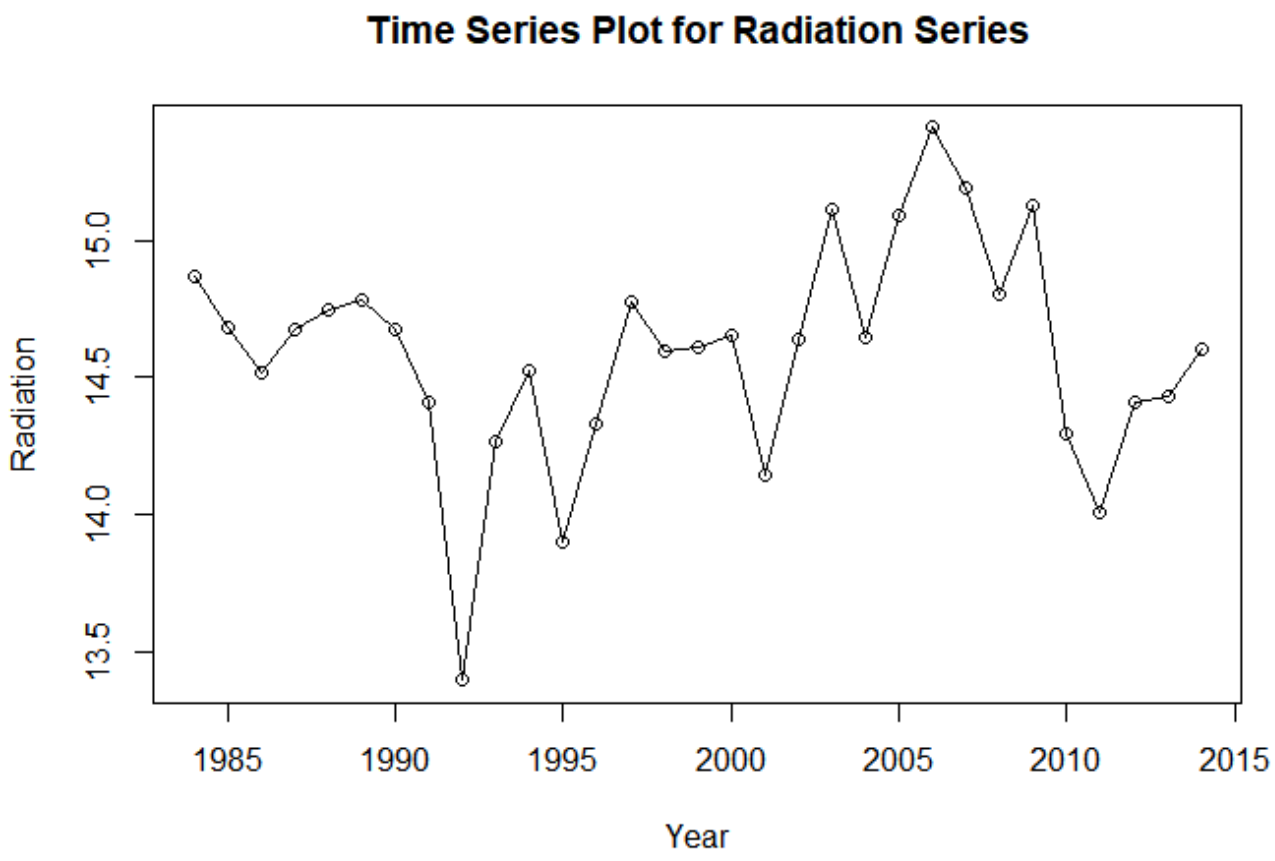

Figure 64

By `Figure 64`, which shows the time series plot of temperature series, we can make the following observations:

- The series has no apparent trends.
- There are no presence of seasonality.
- Changing variance and moving average behavior can be observed
- Multiple Intervention points can be observed in the plot.

```
acf(rbo.rad.ts, lag.max = 48, main="Sample ACF Plot for Radiation Series")
```

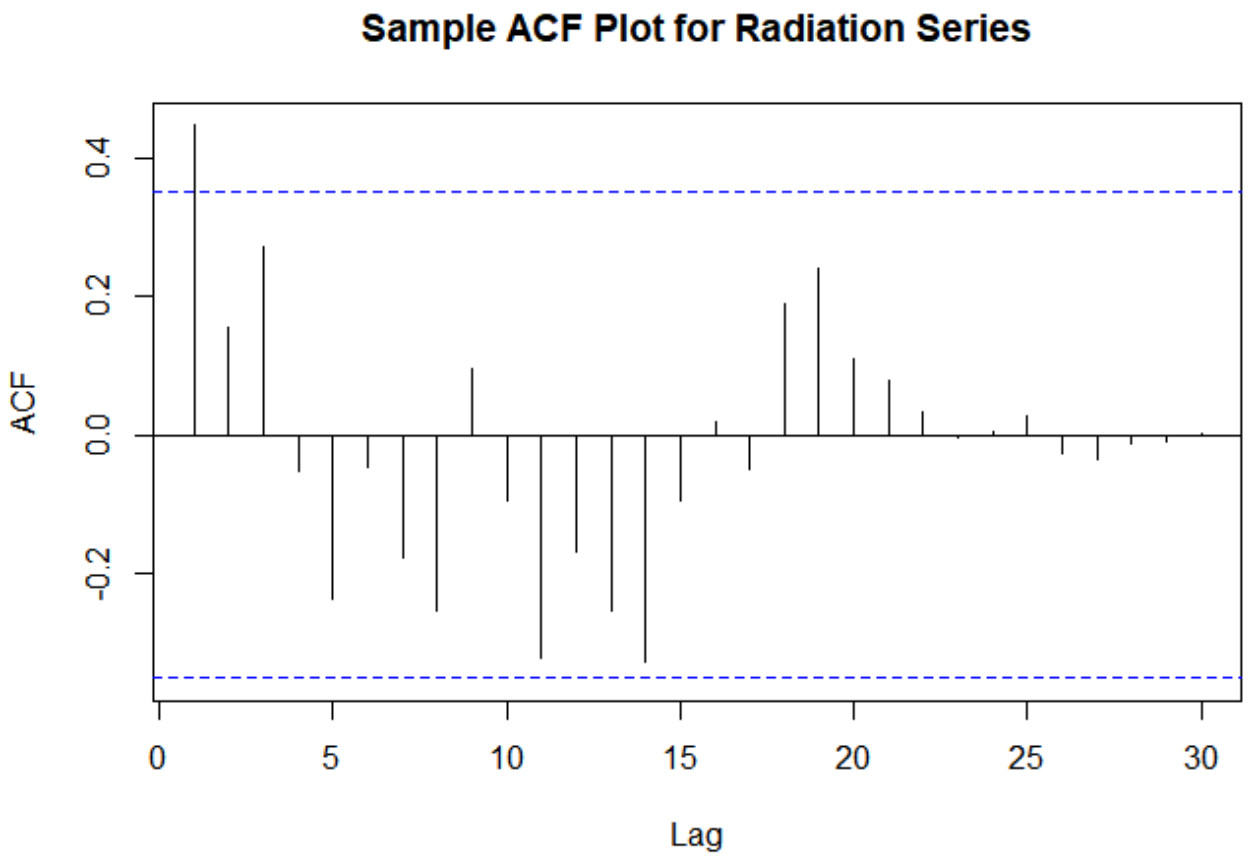

Figure 65

`Figure 65` shows the sample ACF plot for radiation series. Here we can observe that first lag is significant which suggests non-stationary in series.

```
adf.test(rbo.rad.ts, k=ar(rbo.rad.ts)$order)
```

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

In the above output, we used the ADF test for the radiation series. In this approach, we can conclude that the radiation series is non-stationary at 5% level of significance as p value is greater than 0.05.

3.3.5 Relative Humidity

```
plot(rbo.hum.ts,main="Time Series Plot for Relative Humidity Series",
     ylab="Relative Humidity",xlab="Year",type="o")
```

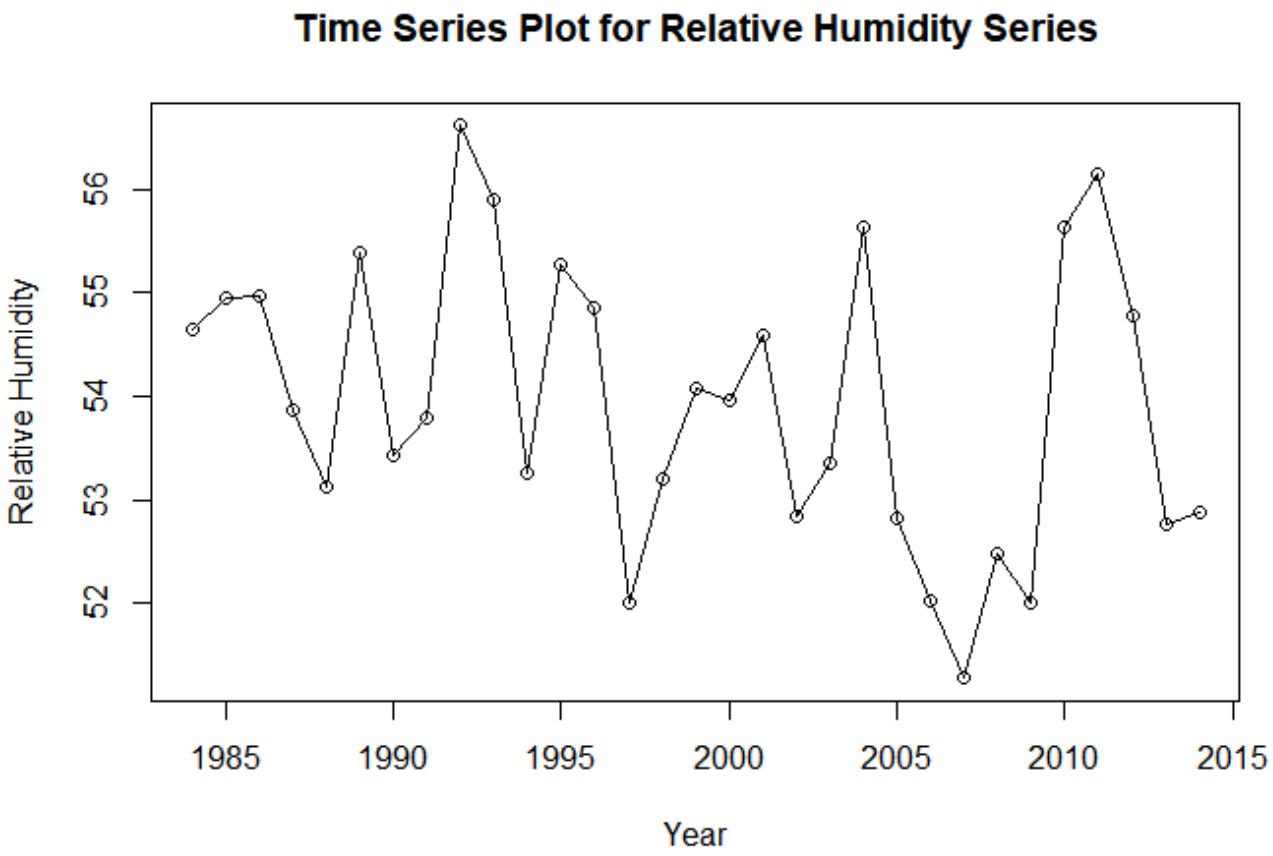

Figure 66

By `Figure 66`, which shows the time series plot of temperature series, we can make the following observations:

- The series has no apparent trends.
- There are no presence of seasonality.
- Changing variance and moving average behavior can be observed
- No intervention points can be observed in the plot.

```
acf(rbo.hum.ts, lag.max = 48, main="Sample ACF Plot for Relative Humidity Series")
```

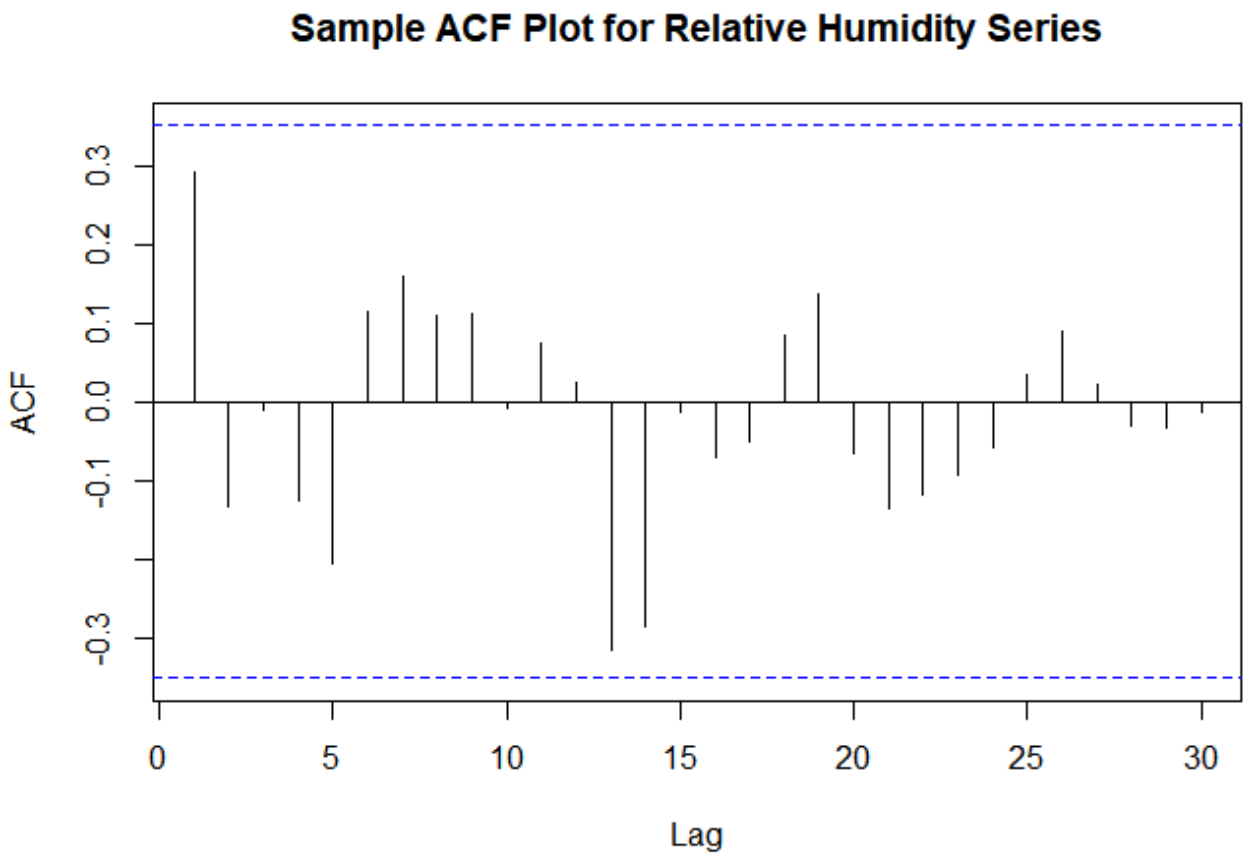

Figure 67

`Figure 67` shows the sample ACF plot for relative humidity series. Here we can observe that first lag is not significant which suggests stationary in series.

```
adf.test(rbo.hum.ts, k=ar(rbo.hum.ts)$order)
```

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

In the above output, we used the ADF test for the radiation series. In this approach, we can conclude that the radiation series is stationary at 5% level of significance as p value is less than 0.05.

3.3.6 Scaled Data Series

```
rbo.scaled<-scale(rbo.ts)

rbo.col<-c("darkolivegreen","coral2","blueviolet","deeppink3","aquamarine")
```

```
plot(rbo.scaled,plot.type="s",col=rbo.col,main="Time Series Plot of Scaled Data Series")
legend("bottomleft",cex=0.7,lty=1,col=rbo.col,c("RBO","Temperature","Rainfall","Radiation","Rel-
        Humidity"))
```

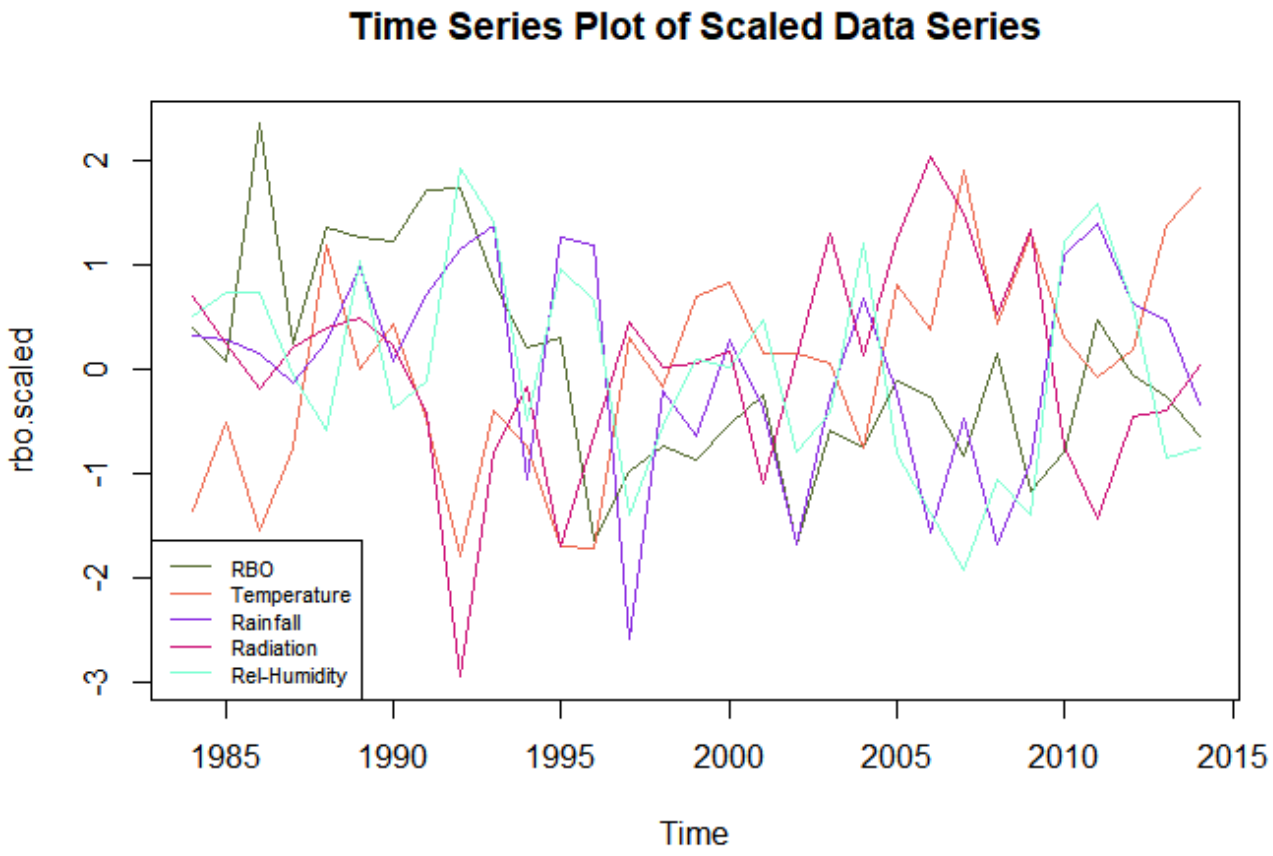

Figure 68

`Figure 68` contains a time series plot of all the data for RBO. To compare these data and fit them in a graph, scaling and centering is done to the data by using the `scale` function.

3.4.0 Dealing with Non-Stationary

From the data visualization section we have identified that there are existence o

---

f non-stationary in variables RBO, Temperature, and Radiation. Hence in this section we will attempt to deal with the non-stationary by using ordinary differencing operation.

3.4.1 RBO

First we will produce the acf and pacf plot for getting a grasp of the changes made by differencing.

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

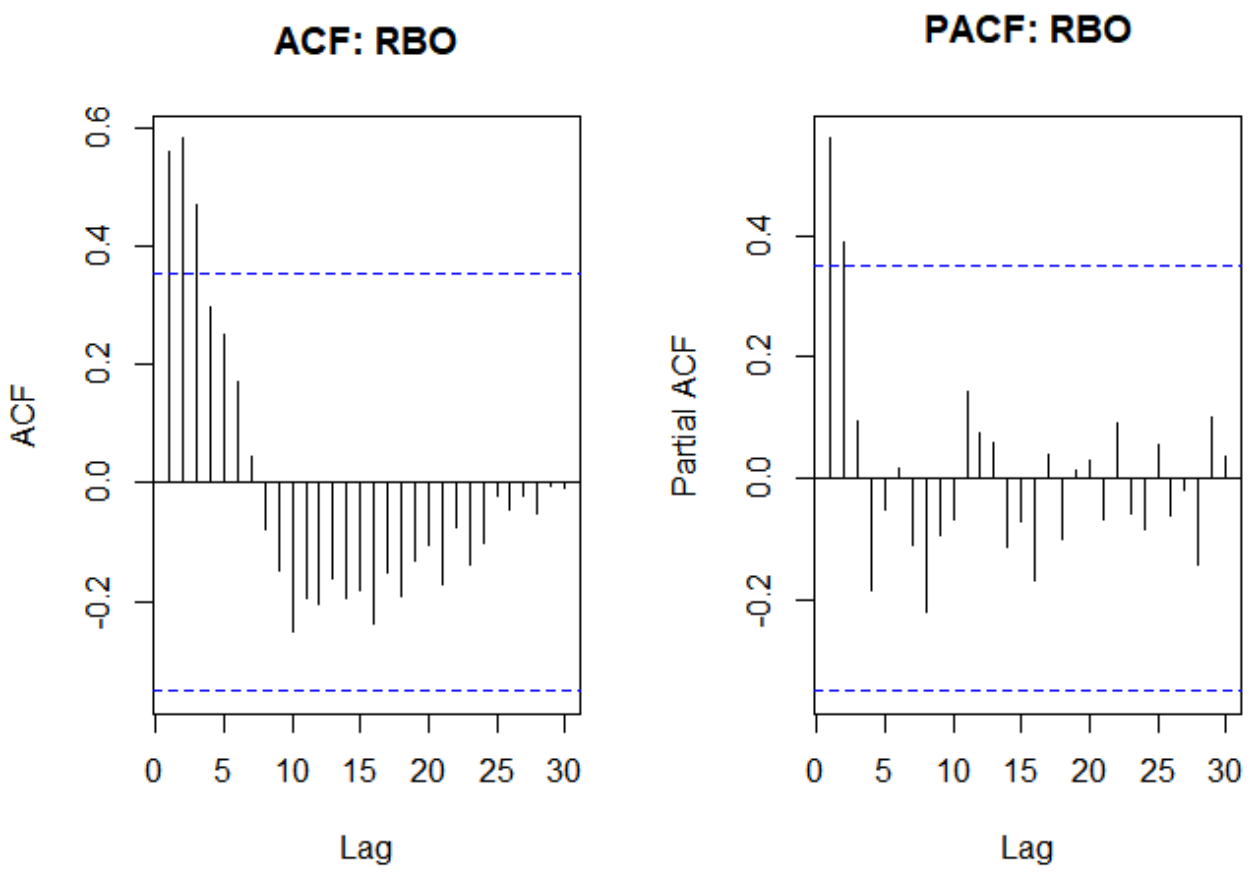

Figure 69

Now we will use the function `diff` to deal with non-stationary data with first difference order.

```
rbo.rbo.diff<-diff(rbo.rbo.ts,differences=1)
adf.test(rbo.rbo.diff, k=ar(rbo.rbo.diff)$order)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  rbo.rbo.diff
## Dickey-Fuller = -6.0473, Lag order = 1, p-value = 0.01
## alternative hypothesis: stationary
```

In the adf output above, we can conclude that the RBO series is now stationary at 5% level of significance as p value is less than 0.05. So we can now plot the acf and pacf plot of the differenced RBO data.

```
par(mfrow=c(1,2))
acf(rbo.rbo.diff,lag.max = 48,main="ACF: RBO Diff")
pacf(rbo.rbo.diff,lag.max = 48,main="PACF: RBO Diff")
```

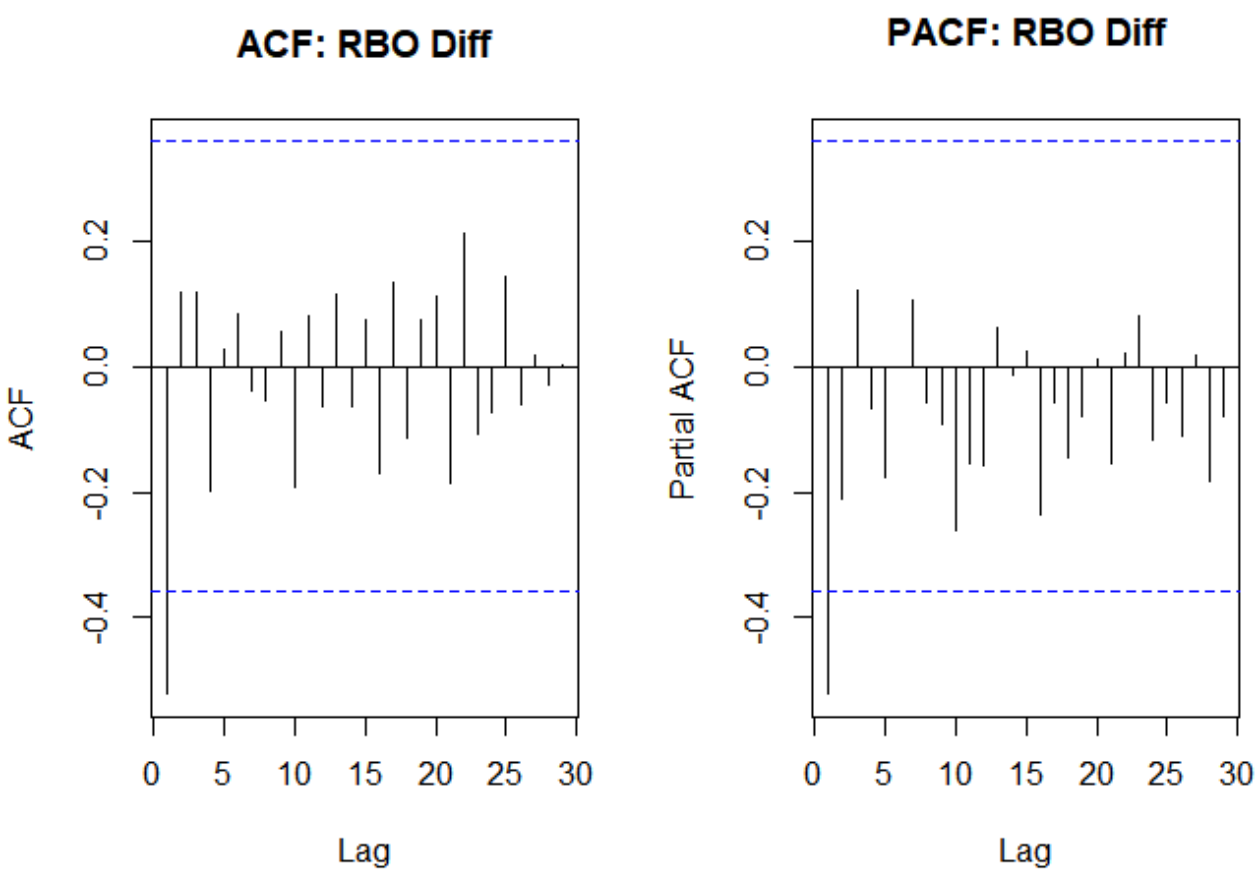

Figure 70

3.4.2 Temperature

First we will produce the acf and pacf plot for getting a grasp of the changes made by differencing.

```
par(mfrow=c(1,2))
acf(rbo.temp.ts,lag.max = 48,main="ACF: Temperature")
pacf(rbo.temp.ts,lag.max = 48,main="PACF: Temperature")
```

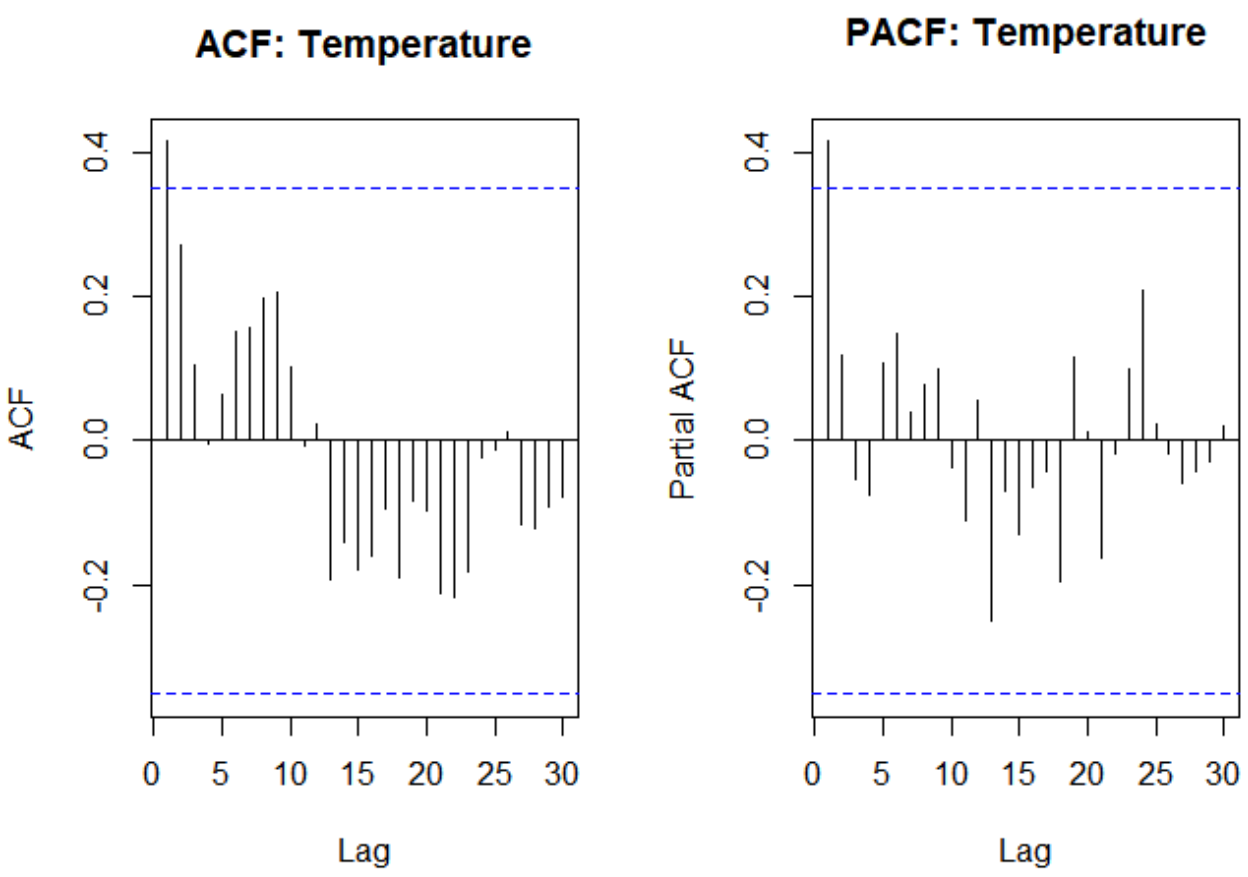

Figure 71

Now we will use the function `diff` to deal with non-stationary data with first difference order.

```
rbo.temp.diff<-diff(rbo.temp.ts,differences=1)
adf.test(rbo.temp.diff, k=ar(rbo.temp.diff)$order)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  rbo.temp.diff
## Dickey-Fuller = -4.6021, Lag order = 1, p-value = 0.01
## alternative hypothesis: stationary
```

In the adf output above, we can conclude that the Temperature series is now stationary at 5% level of significance as p value is less than 0.05. So we can now plot the acf and pacf plot of the differenced Temperature data.

```
par(mfrow=c(1,2))
acf(rbo.temp.diff,lag.max = 48,main="ACF: Temperature Diff")
pacf(rbo.temp.diff,lag.max = 48,main="PACF: Temperature Diff")
```

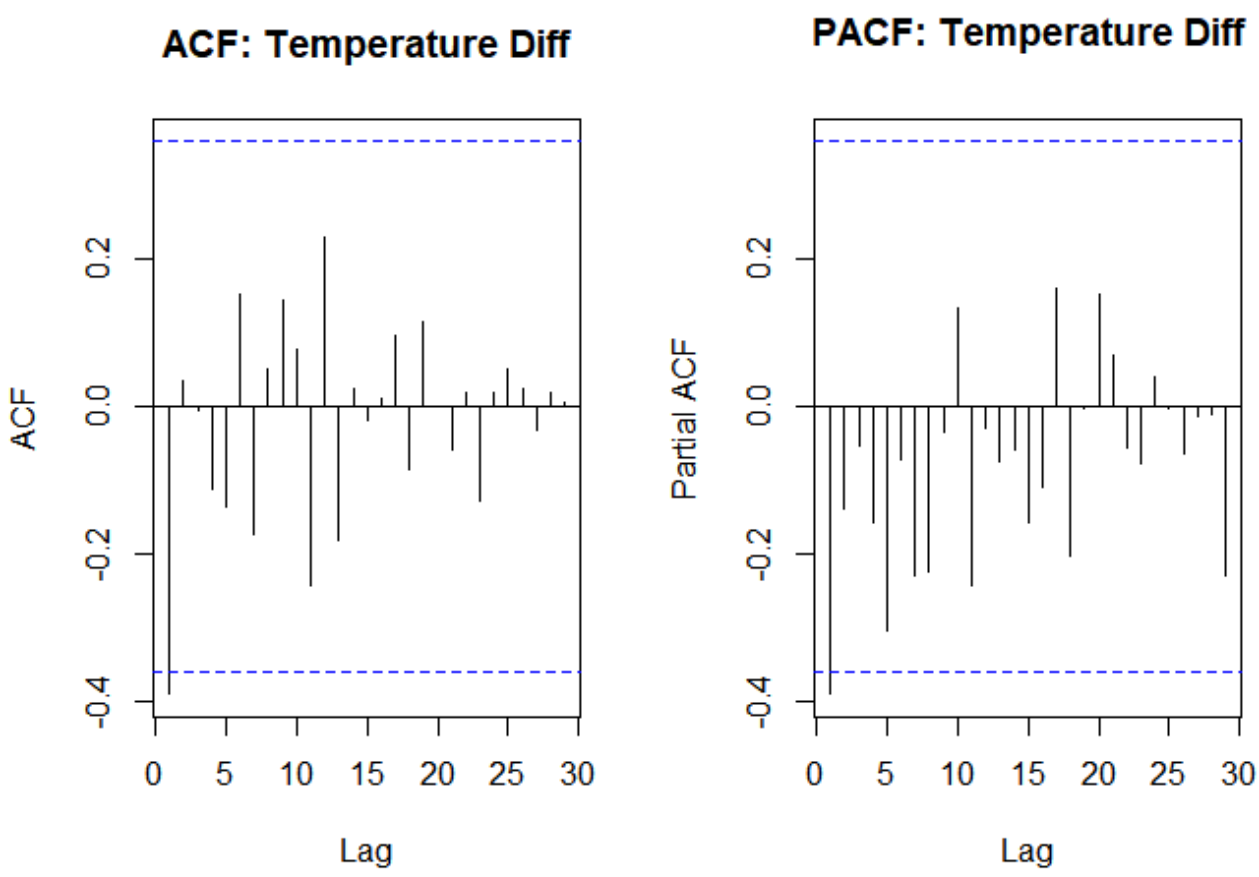

Figure 72

3.4.3 Radiation

First we will produce the acf and pacf plot for getting a grasp of the changes made by differencing.

```
par(mfrow=c(1,2))
acf(rbo.rad.ts,lag.max = 48,main="ACF: Radiation")
pacf(rbo.rad.ts,lag.max = 48,main="PACF: Radiation")
```

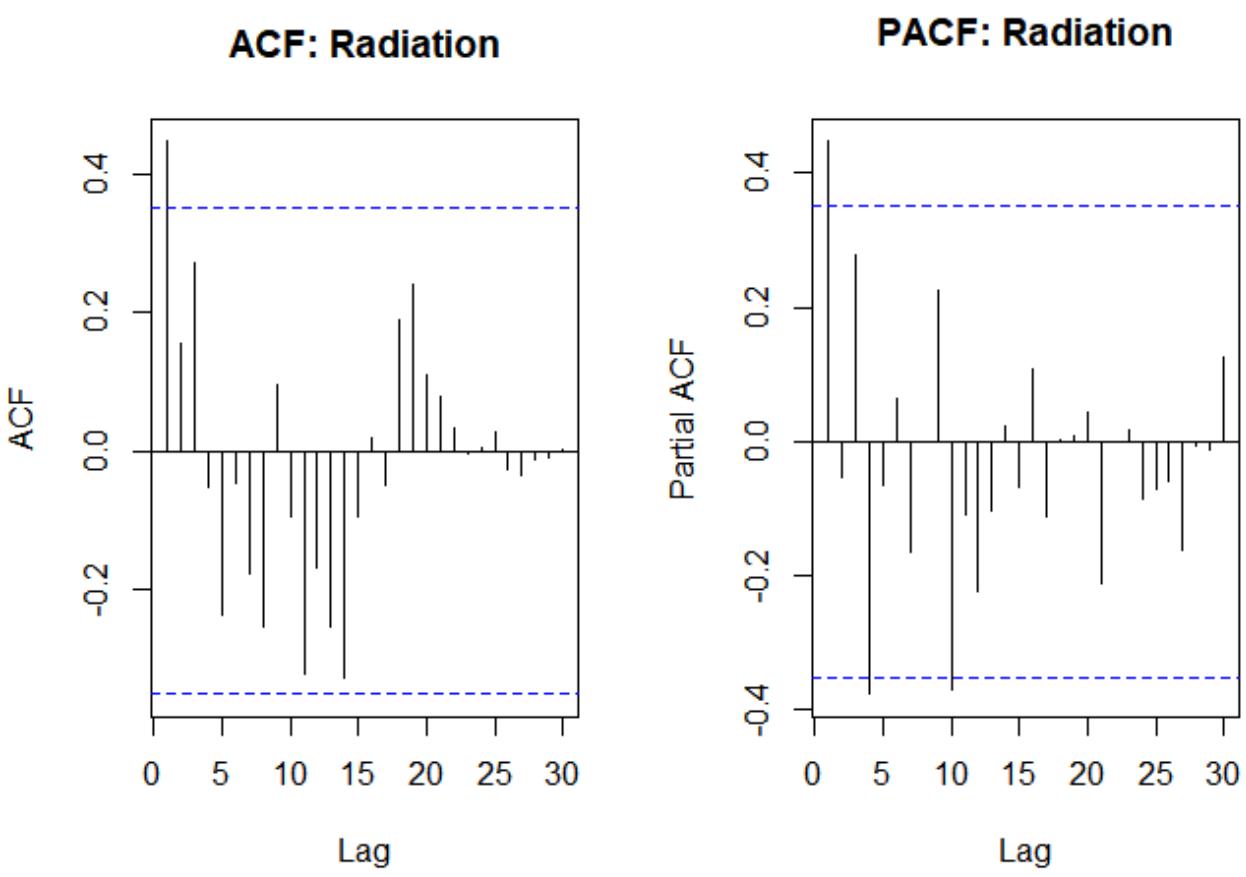

Figure 73

Now we will use the function `diff` to deal with non-stationary data with second difference order.

```
rbo.rad.diff<-diff(rbo.rad.ts,differences=2)
adf.test(rbo.rad.diff, k=ar(rbo.rad.diff)$order)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  rbo.rad.diff
## Dickey-Fuller = -4.7561, Lag order = 2, p-value = 0.01
## alternative hypothesis: stationary
```

In the adf output above, we can conclude that the Radiation series is now stationary at 5% level of significance as p value is less than 0.05. So we can now plot the acf and pacf plot of the differenced Radiation data.

```
par(mfrow=c(1,2))
acf(rbo.rad.diff,lag.max = 48,main="ACF: Radiation Diff")
pacf(rbo.rad.diff,lag.max = 48,main="PACF: Radiation Diff")
```

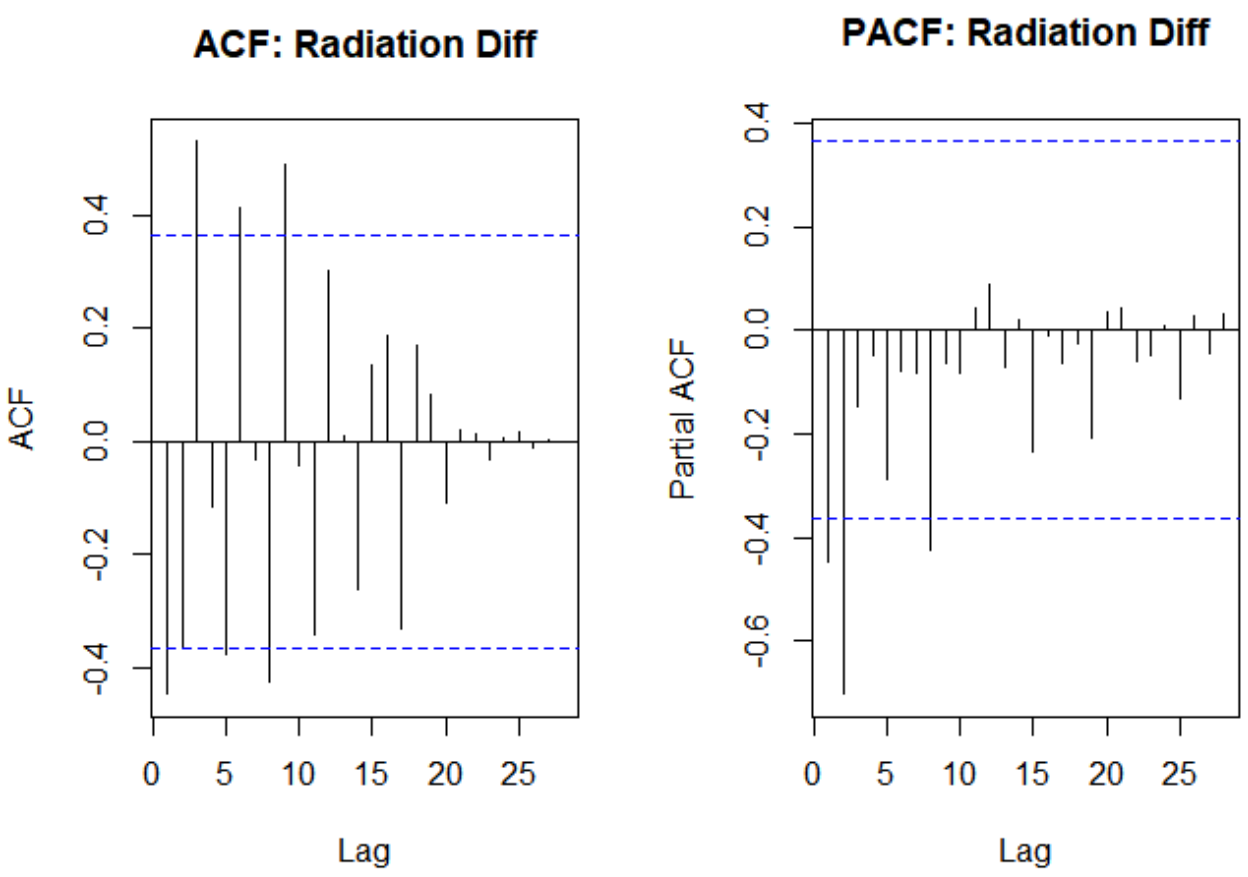

Figure 74

With the differencing operation, we are successful in dealing with the non-stationary data by using order differences 1, 1, and 2 for RBO, temperature, and radiation series.

3.5.0 Correlation

In this section, we will look at the correlation matrix for the data set by using the `cor` function.

```
cor(rbo.ts)
```

```
##                    RBO Temperature   Rainfall  Radiation RelHumidity
## RBO          1.0000000  -0.3452053  0.3932282 -0.3173602   0.3885540
## Temperature -0.3452053   1.0000000 -0.3915072  0.5193576  -0.6621980
## Rainfall     0.3932282  -0.3915072  1.0000000 -0.5813161   0.7911079
## Radiation   -0.3173602   0.5193576 -0.5813161  1.0000000  -0.7354087
## RelHumidity  0.3885540  -0.6621980  0.7911079 -0.7354087   1.0000000
```

From the correlation output above, we can see that for RBO, there is a low negative correlation with Temperature of -0.3452053, low positive correlation with Rainfall of 0.3932282, low negative correlation with Radiation of -0.3173602, and low positive correlation with Relative Humidity of 0.3885540. Since we will be analyzing RBO, we shall use RBO as the dependent variable y and the rest of the variables in the series as independent variable x. To further perform univariate analysis of predictors, we shall test each variable with every models and compare mase values to find the best model.

3.6.0 Time Series Regression Models

3.6.1 Finite Distributed Lag Model

For Finite DLM, we will be using temperature as the independent variable, and the RBO will be the dependent variable. To perform this test, we will first create a loop function and use the `dlm` function to identify most appropriate number of lags for the DLM.

```
for (i in 10){
model1<-dlm(x=as.vector(rbo.temp.ts), y=as.vector(rbo.rbo.ts), q=i)
cat("q =",i,"AIC =",AIC(model1$model),"BIC =",BIC(model1$model),"MASE=", MASE(model1)$MASE,
"\n")
}
```

```
## q = 10 AIC = -85.18937 BIC = -71.61058 MASE= 0.523527
```

From the loop function above we will use q=10 as our argument for lag length in the analysis of other variables. Now we will use the `dlm` function to fit the finite dlm for other predictors, and we will compare the aic, bic, and mase values for each predictors.

```
#RBO & Temperature
model1.1<-dlm(x=as.vector(rbo.temp.ts),y=as.vector(rbo.rbo.ts),q=10)
#RBO & Rainfall
model1.2<-dlm(x=as.vector(rbo.rain.ts),y=as.vector(rbo.rbo.ts),q=10)
#RBO & Radiation
model1.3<-dlm(x=as.vector(rbo.rad.ts),y=as.vector(rbo.rbo.ts),q=10)
#RBO & Relative Humidity
model1.4<-dlm(x=as.vector(rbo.hum.ts),y=as.vector(rbo.rbo.ts),q=10)
#Compare
predictor<-c("Temperature","Rainfall","Radiation","Relative Humidity")
fdlm.mase<-MASE(model1.1,model1.2,model1.3,model1.4)
fdlm.aic<-c(AIC(model1.1),AIC(model1.2),AIC(model1.3),AIC(model1.4))
```

```
## [1] -85.18937
## [1] -76.93255
## [1] -78.06879
## [1] -77.84443
```

```
fdlm.bic<-c(BIC(model1.1),BIC(model1.2),BIC(model1.3),BIC(model1.4))
```

```
## [1] -71.61058
## [1] -63.35376
```

```
## [1] -64.49
## [1] -64.26564
```

```
fdlm<-data.frame(predictor,fdlm.mase,fdlm.aic,fdlm.bic)
arrange(fdlm,MASE)
```

```
##                  predictor  n      MASE  fdlm.aic  fdlm.bic
## model1.1       Temperature 21 0.5235270 -85.18937 -71.61058
## model1.2          Rainfall 21 0.5856170 -76.93255 -63.35376
## model1.4 Relative Humidity 21 0.5947606 -77.84443 -64.26564
## model1.3         Radiation 21 0.6037290 -78.06879 -64.49000
```

From the predictor comparison output above we can see that `model1.1` for predictor RBO and Temperature has the lowest AIC, BIC, and MASE values, hence we shall select this predictor for further analysis.

```
summary(model1.1)
```

```
##
## Call:
## lm(formula = model.formula, data = design)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.02910 -0.01175 -0.00495  0.01596  0.02903
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.301149   0.398691   0.755    0.469
## x.t         -0.011599   0.012936  -0.897    0.393
## x.1          0.024259   0.013847   1.752    0.114
## x.2         -0.007121   0.012630  -0.564    0.587
## x.3         -0.010736   0.012412  -0.865    0.410
## x.4          0.010868   0.012314   0.883    0.400
## x.5         -0.001881   0.012441  -0.151    0.883
## x.6          0.022337   0.012282   1.819    0.102
## x.7          0.001619   0.011577   0.140    0.892
## x.8         -0.006598   0.011773  -0.560    0.589
## x.9         -0.007787   0.012477  -0.624    0.548
## x.10         0.007677   0.012192   0.630    0.545
##
## Residual standard error: 0.02618 on 9 degrees of freedom
## Multiple R-squared:  0.5452, Adjusted R-squared:  -0.0107
## F-statistic: 0.9808 on 11 and 9 DF,  p-value: 0.5205
##
## AIC and BIC values for the model:
```

```
##         AIC       BIC
## 1 -85.18937 -71.61058
```

By using the `summary` function, we can observe that in `model1.1` the lag weights of the predictors are not significant at the 5% statistical level as p-value > 0.05. Furthermore, there is a adjusted R-square value of -0.0107, which accounts for -1.07% of the variability in the data. Here we can also see that for the lags, positive lag coefficients show that higher RBO values indicate similarity of the order of the first flowering occurence. First lag indicates that flowering is not similar as x.1=0.024259, which is small value.

```
checkresiduals(model1.1$model$residuals)
```

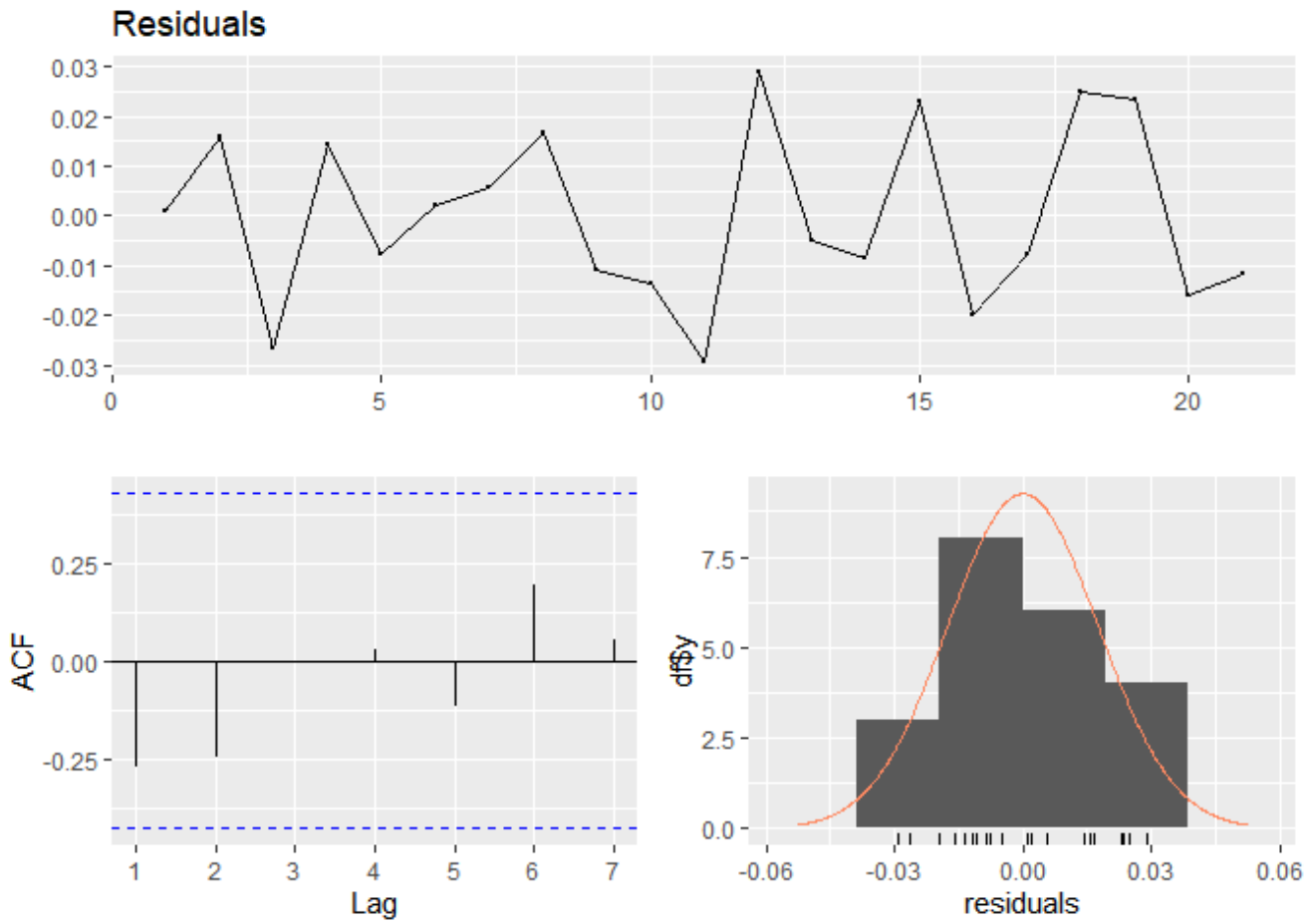

Figure 75

```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 3.358, df = 4, p-value = 0.4998
##
## Model df: 0.   Total lags used: 4
```

```
bgtest(model1.1$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model1.1$model
## LM test = 1.677, df = 1, p-value = 0.1953
```

```
shapiro.test(model1.1$model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model1.1$model$residuals
## W = 0.94816, p-value = 0.3143
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 75`. The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are not significant. Furthermore, from the Shapiro-Wilk test we can conclude that we do not reject null hypothesis of normality as p-value > 0.05.

```
VIF.model1.1<-vif(model1.1$model)
VIF.model1.1
```

```
##      x.t      x.1      x.2      x.3      x.4      x.5      x.6      x.7
## 1.882817 1.925625 1.781877 1.739787 1.730817 1.757883 1.682573 1.517274
##      x.8      x.9     x.10
## 1.373584 1.508769 1.414382
```

```
VIF.model1.1>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
```

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the vif function, we can observe that there is no issue with multicollinearity as VIF values are all greater than 10.

3.6.2 Polynomial Distributed Lag Models

Similar to the finite DLM, we shall compare each predictor with the polynomial dlm and select the best with lowest aic, bic, and mase values.

```r
#RBO & Temperature
model2.1<-polyDlm(x=as.vector(rbo.temp.ts),y=as.vector(rbo.rbo.ts),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
```

```r
#RBO & Rainfall
model2.2<-polyDlm(x=as.vector(rbo.rain.ts),y=as.vector(rbo.rbo.ts),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
```

```r
#RBO & Radiation
model2.3<-polyDlm(x=as.vector(rbo.rad.ts),y=as.vector(rbo.rbo.ts),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
```

```
#RBO & Relative Humidity
model2.4<-polyDlm(x=as.vector(rbo.hum.ts),y=as.vector(rbo.rbo.ts),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
```

```
#Compare
predictor<-c("Temperature","Rainfall","Radiation","Relative Humidity")
pdlm.mase<-MASE(model2.1,model2.2,model2.3,model2.4)
pdlm.aic<-c(AIC(model2.1),AIC(model2.2),AIC(model2.3),AIC(model2.4))
```

```
## [1] -87.29374
## [1] -88.26525
## [1] -88.15084
## [1] -88.49321
```

```
pdlm.bic<-c(BIC(model2.1),BIC(model2.2),BIC(model2.3),BIC(model2.4))
```

```
## [1] -82.07113
## [1] -83.04264
## [1] -82.92823
## [1] -83.2706
```

```
pdlm<-data.frame(predictor,pdlm.mase,pdlm.aic,pdlm.bic)
arrange(pdlm,MASE)
```

```
##                  predictor  n      MASE  pdlm.aic  pdlm.bic
## model2.2          Rainfall 21 0.6451804 -88.26525 -83.04264
## model2.4 Relative Humidity 21 0.6523137 -88.49321 -83.27060
## model2.3         Radiation 21 0.6711964 -88.15084 -82.92823
## model2.1       Temperature 21 0.6714874 -87.29374 -82.07113
```

From the output above, we can observe that predictor rainfall has the lowest MASE scores. However, predictor relative humidity has the lowest aic score, and predictor radiation has the lowest bic score. We shall use MASE as a better selection and select predictor rainfall for further analysis.

```
summary(model2.2)
```

```
##
## 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
```

By using the `summary` function, we can observe that in `model2.2` the lag weights of the predictors are not significant at the 5% statistical level as p-value > 0.05. Furthermore, there is a adjusted R-square value of 0.009865, which accounts for 0.98% of the variability in the data. Here we can also see that for the lags, positive lag coefficients show that higher RBO values indicate similarity of the order of the first flowering occurence. First lag indicates that flowering is similar as z.t0=0.7166742, which is large value.

```
checkresiduals(model2.2$model$residuals)
```

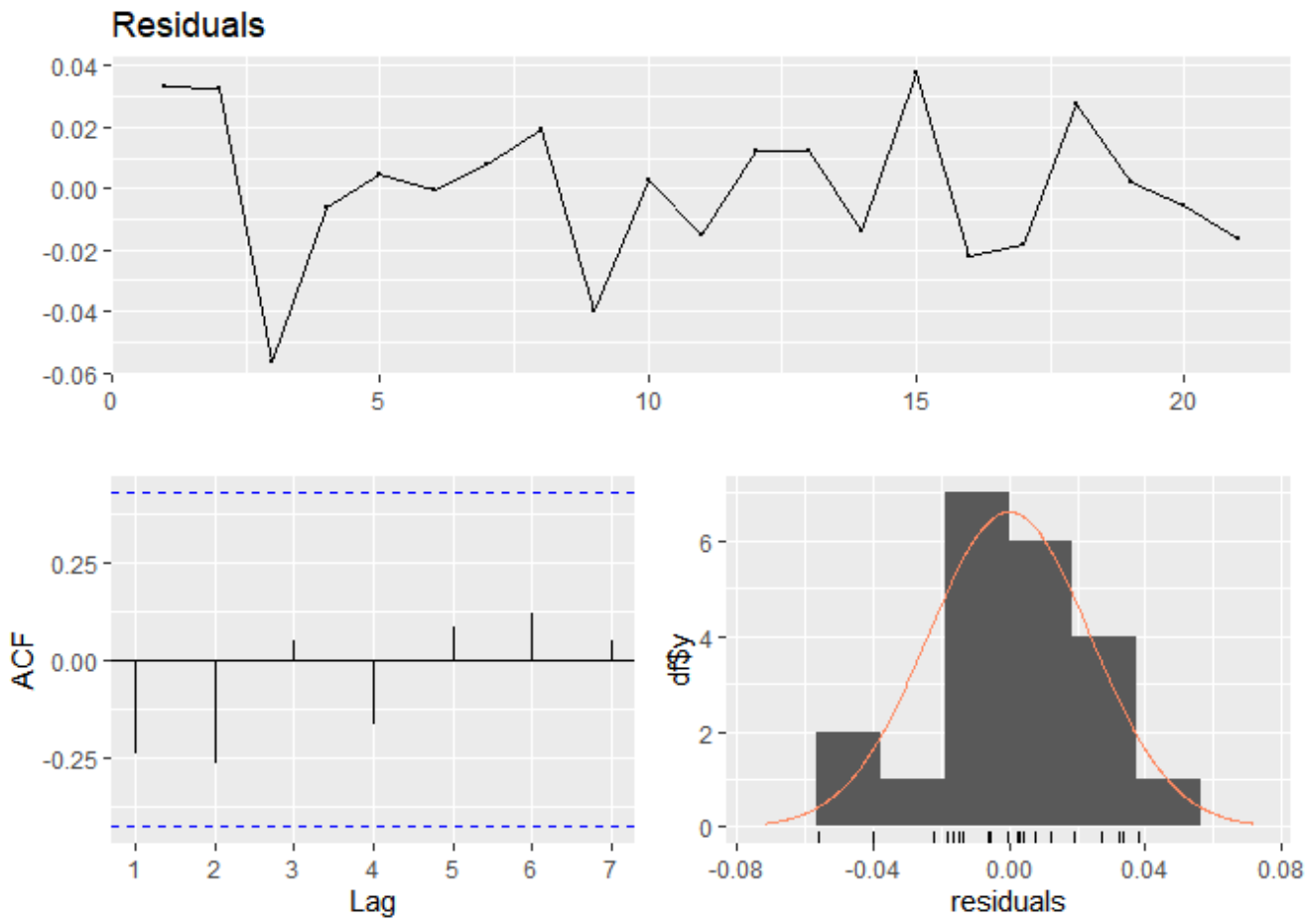

Figure 76

```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 4.0189, df = 4, p-value = 0.4035
##
## Model df: 0.   Total lags used: 4
```

```
bgtest(model2.2$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model2.2$model
## LM test = 1.2962, df = 1, p-value = 0.2549
```

```
shapiro.test(model2.2$model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
```

```
## data:  model2.2$model$residuals
## W = 0.97008, p-value = 0.7348
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 76`. The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are not significant. Furthermore, from the Shapiro-Wilk test we can conclude that we do not reject null hypothesis of normality as p-value > 0.05.

```
VIF.model2.2<-vif(model2.2$model)
VIF.model2.2
```

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

```
VIF.model2.2>10
```

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

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the vif function, we can observe that there is issue with multicollinearity as VIF values z.t1 and z.t2 are greater than 10.

3.6.3 Koyck Distributed Lag Model

Similar to the previous models, we shall compare each predictor with the Koyck dlm and select the best in terms of aic, bic, and mase values.

```
#FFD & Temperature
model3.1<-koyckDlm(x=as.vector(rbo.temp.ts),y=as.vector(rbo.rbo.ts))
#FFD & Rainfall
model3.2<-koyckDlm(x=as.vector(rbo.rain.ts),y=as.vector(rbo.rbo.ts))
#FFD & Radiation
model3.3<-koyckDlm(x=as.vector(rbo.rad.ts),y=as.vector(rbo.rbo.ts))
#FFD & Relative Humidity
model3.4<-koyckDlm(x=as.vector(rbo.hum.ts),y=as.vector(rbo.rbo.ts))
#Compare
predictor<-c("Temperature","Rainfall","Radiation","Relative Humidity")
kdlm.mase<-MASE(model3.1,model3.2,model3.3,model3.4)
kdlm.aic<-c(AIC(model3.1),AIC(model3.2),AIC(model3.3),AIC(model3.4))
```

```
## [1] -98.54907
## [1] 76.22068
## [1] -93.8669
## [1] -106.3136
```

```
kdlm.bic<-c(BIC(model3.1),BIC(model3.2),BIC(model3.3),BIC(model3.4))
```

```
## [1] -92.94428
## [1] 81.82547
## [1] -88.26211
## [1] -100.7088
```

```
kdlm<-data.frame(predictor,kdlm.mase,kdlm.aic,kdlm.bic)
arrange(kdlm,MASE)
```

```
##                  predictor  n       MASE   kdlm.aic   kdlm.bic
## model3.4 Relative Humidity 30  0.8400135 -106.31363 -100.70884
## model3.1       Temperature 30  0.9535116  -98.54907  -92.94428
## model3.3         Radiation 30  1.0314227  -93.86690  -88.26211
## model3.2          Rainfall 30 19.1057647   76.22068   81.82547
```

From the output above, we can see that predictor Relative Humidity is the best choice as it has the lowest MASE, AIC, and BIC values. Hence we will use this predictor for further analysis.

```
summary(model3.4)
```

```
##
## 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
```

By using the `summary` function, we can observe that in `model3.4` the lag weights of the predictors are significant at the 5% statistical level as p-value < 0.05. Furthermore, there is a adjusted R-square value of 0.2967, which accounts for 29.67% of the variability in the data. Here we can also see that for the lags, positive lag coefficients show that higher RBO values indicate similarity of the order of the first flowering occurrence. First lag indicates that flowering is neither similar or not as Y.1=0.548465.

```
checkresiduals(model3.4$model$residuals)
```

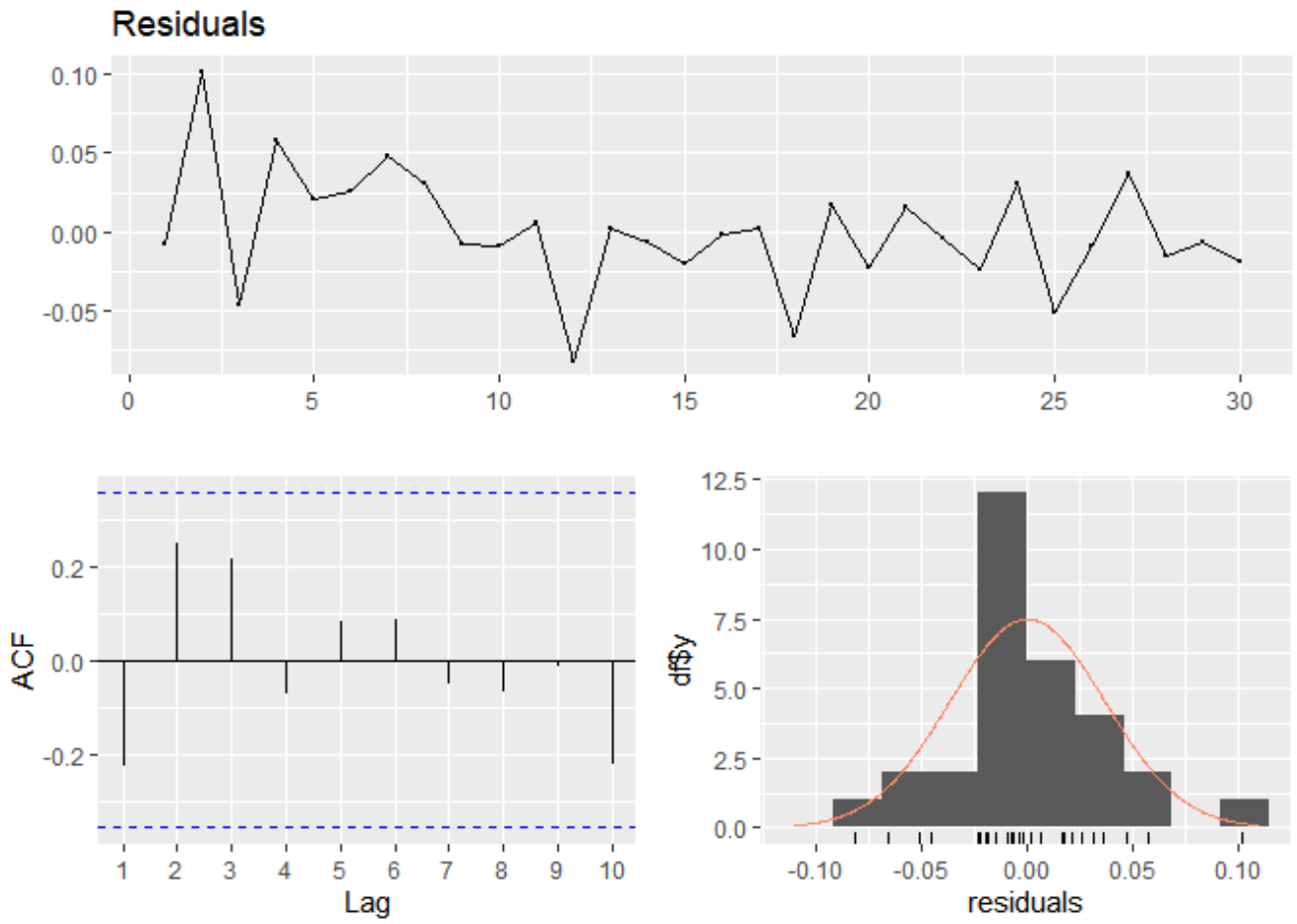

Figure 77

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

```
bgtest(model3.4$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model3.4$model
## LM test = 3.6165, df = 1, p-value = 0.05721
```

```
shapiro.test(model3.4$model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model3.4$model$residuals
## W = 0.96587, p-value = 0.4332
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 77`. The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are not significant. Furthermore, from the Shapiro-Wilk test we can conclude that we do not reject null hypothesis of normality as p-value > 0.05.

```
VIF.model3.4<-vif(model3.4$model)
VIF.model3.4
```

```
##      Y.1      X.t
## 3.449396 3.449396
```

```
VIF.model3.4>10
```

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

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the vif function, we can observe that there no issue with multicollinearity as VIF values are all less than 10.

3.6.4 Autoregressive Distributed Lag Model

Similar to previous models, we will compare and find the best fit and predictors based on the aic, bic, and mase values. To perform the Autoregressive DLM, we will first create a loop function for p and q and use the `ardlDLM` function to identify the most appropriate number of lags for the DLM.

```
for (i in 1:5){
for(j in 1:5){
  model4.1 = ardlDlm(x = as.vector(rbo.temp.ts), y = as.vector(rbo.rbo.ts), p = i ,
                     q = j)
  cat("p =", i, "q =", j, "AIC =", AIC(model4.1$model), "BIC =",
      BIC(model4.1$model), "MASE =",
      MASE(model4.1)$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
```

From the output above, we can see that for P=1 Q=5 has the lowest AIC value, P=1 Q=3 has the lowest BIC value, and P=5 Q=5 have the lowest MASE value. Therefore we will compare the predictors with these 3 arguments.

```
#P=1 Q=5
#RBO & Temperature
model4.2.1<-ardlDlm(x=as.vector(rbo.temp.ts),y=as.vector(rbo.rbo.ts),p=1,q=5)
#RBO & Rainfall
```

```
model4.2.2<-ardlDlm(x=as.vector(rbo.rain.ts),y=as.vector(rbo.rbo.ts),p=1,q=5)
#RBO & Radiation
model4.2.3<-ardlDlm(x=as.vector(rbo.rad.ts),y=as.vector(rbo.rbo.ts),p=1,q=5)
#RBO & Relative Humidity
model4.2.4<-ardlDlm(x=as.vector(rbo.hum.ts),y=as.vector(rbo.rbo.ts),p=1,q=5)

#P=1 Q=3
#RBO & Temperature
model4.3.1<-ardlDlm(x=as.vector(rbo.temp.ts),y=as.vector(rbo.rbo.ts),p=1,q=3)
#RBO & Rainfall
model4.3.2<-ardlDlm(x=as.vector(rbo.rain.ts),y=as.vector(rbo.rbo.ts),p=1,q=3)
#RBO & Radiation
model4.3.3<-ardlDlm(x=as.vector(rbo.rad.ts),y=as.vector(rbo.rbo.ts),p=1,q=3)
#RBO & Relative Humidity
model4.3.4<-ardlDlm(x=as.vector(rbo.hum.ts),y=as.vector(rbo.rbo.ts),p=1,q=3)

#P=5 Q=5
#RBO & Temperature
model4.4.1<-ardlDlm(x=as.vector(rbo.temp.ts),y=as.vector(rbo.rbo.ts),p=5,q=5)
#RBO & Rainfall
model4.4.2<-ardlDlm(x=as.vector(rbo.rain.ts),y=as.vector(rbo.rbo.ts),p=5,q=5)
#RBO & Radiation
model4.4.3<-ardlDlm(x=as.vector(rbo.rad.ts),y=as.vector(rbo.rbo.ts),p=5,q=5)
#RBO & Relative Humidity
model4.4.4<-ardlDlm(x=as.vector(rbo.hum.ts),y=as.vector(rbo.rbo.ts),p=5,q=5)

predictor<-c("Temperature","Rainfall","Radiation","Relative Humidity")
adlm.mase1<-MASE(model4.2.1,model4.2.2,model4.2.3,model4.2.4)
adlm.aic1<-c(AIC(model4.2.1),AIC(model4.2.2),AIC(model4.2.3),AIC(model4.2.4))
```

```
## [1] -97.33529
## [1] -95.80256
## [1] -100.2033
## [1] -97.27466
```

```
adlm.bic1<-c(BIC(model4.2.1),BIC(model4.2.2),BIC(model4.2.3),BIC(model4.2.4))
```

```
## [1] -86.01242
## [1] -84.47969
## [1] -88.88038
## [1] -85.95179
```

```
adlm1<-data.frame(predictor,adlm.mase1,adlm.aic1,adlm.bic1)
colnames(adlm1)<-c("predictor","n","MASE","AIC","BIC")

adlm.mase2<-MASE(model4.3.1,model4.3.2,model4.3.3,model4.3.4)
adlm.aic2<-c(AIC(model4.3.1),AIC(model4.3.2),AIC(model4.3.3),AIC(model4.3.4))
```

```
## [1] -109.2081
## [1] -106.9248
## [1] -110.6338
## [1] -108.3405
```

```
adlm.bic2<-c(BIC(model4.3.1),BIC(model4.3.2),BIC(model4.3.3),BIC(model4.3.4))
```

```
## [1] -99.88269
## [1] -97.59935
## [1] -101.3084
## [1] -99.01504
```

```
adlm2<-data.frame(predictor,adlm.mase2,adlm.aic2,adlm.bic2)
colnames(adlm2)<-c("predictor","n","MASE","AIC","BIC")

adlm.mase3<-MASE(model4.4.1,model4.4.2,model4.4.3,model4.4.4)
adlm.aic3<-c(AIC(model4.4.1),AIC(model4.4.2),AIC(model4.4.3),AIC(model4.4.4))
```

```
## [1] -93.93583
## [1] -90.68282
## [1] -93.6481
## [1] -92.45536
```

```
adlm.bic3<-c(BIC(model4.4.1),BIC(model4.4.2),BIC(model4.4.3),BIC(model4.4.4))
```

```
## [1] -77.58057
## [1] -74.32757
## [1] -77.29285
## [1] -76.1001
```

```
adlm3<-data.frame(predictor,adlm.mase3,adlm.aic3,adlm.bic3)
colnames(adlm3)<-c("predictor","n","MASE","AIC","BIC")

adlm4<-rbind(adlm1,adlm2,adlm3)
arrange(adlm4,MASE)
```

```
##                    predictor  n      MASE        AIC        BIC
## model4.4.3         Radiation 26 0.7145698  -93.64810  -77.29285
## model4.4.1       Temperature 26 0.7335979  -93.93583  -77.58057
## model4.4.4 Relative Humidity 26 0.7370787  -92.45536  -76.10010
## model4.2.1       Temperature 26 0.7402468  -97.33529  -86.01242
## model4.4.2          Rainfall 26 0.7469974  -90.68282  -74.32757
```

```
## model4.2.3         Radiation 26 0.7627672 -100.20325  -88.88038
## model4.3.1       Temperature 28 0.7734247 -109.20812  -99.88269
## model4.2.4 Relative Humidity 26 0.7813433  -97.27466  -85.95179
## model4.2.2          Rainfall 26 0.8152025  -95.80256  -84.47969
## model4.3.4 Relative Humidity 28 0.8196903 -108.34047  -99.01504
## model4.3.3         Radiation 28 0.8202653 -110.63384 -101.30840
## model4.3.2          Rainfall 28 0.8322089 -106.92478  -97.59935
```

In the output above, we can observe that `model4.4.3` which is radiation predictor (P5 Q5) has the lowest MASE values, `model4.3.3`, radiation predictor (P1 Q3) has the lowest AIC values, and BIC values. In our case, we will select `model4.4.3` as it has the lowest MASE values for further analysis.

```
summary(model4.4.3)
```

```
##
## 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
```

By using the `summary` function, we can observe that in `model4.4.3` the lag weights of the predictors are not significant at the 5% statistical level as p-value > 0.05. Furthermore, there is a adjusted R-square value of 0.3689, which accounts for 36.89% of the variability in the data. Here we can also see that for

the lags, positive lag coefficients show that higher RBO values indicate similarity of the order of the first flowering occurrence. First lag indicates that flowering is not similar as X.1=0..026744.

```
checkresiduals(model4.4.3$model$residuals)
```

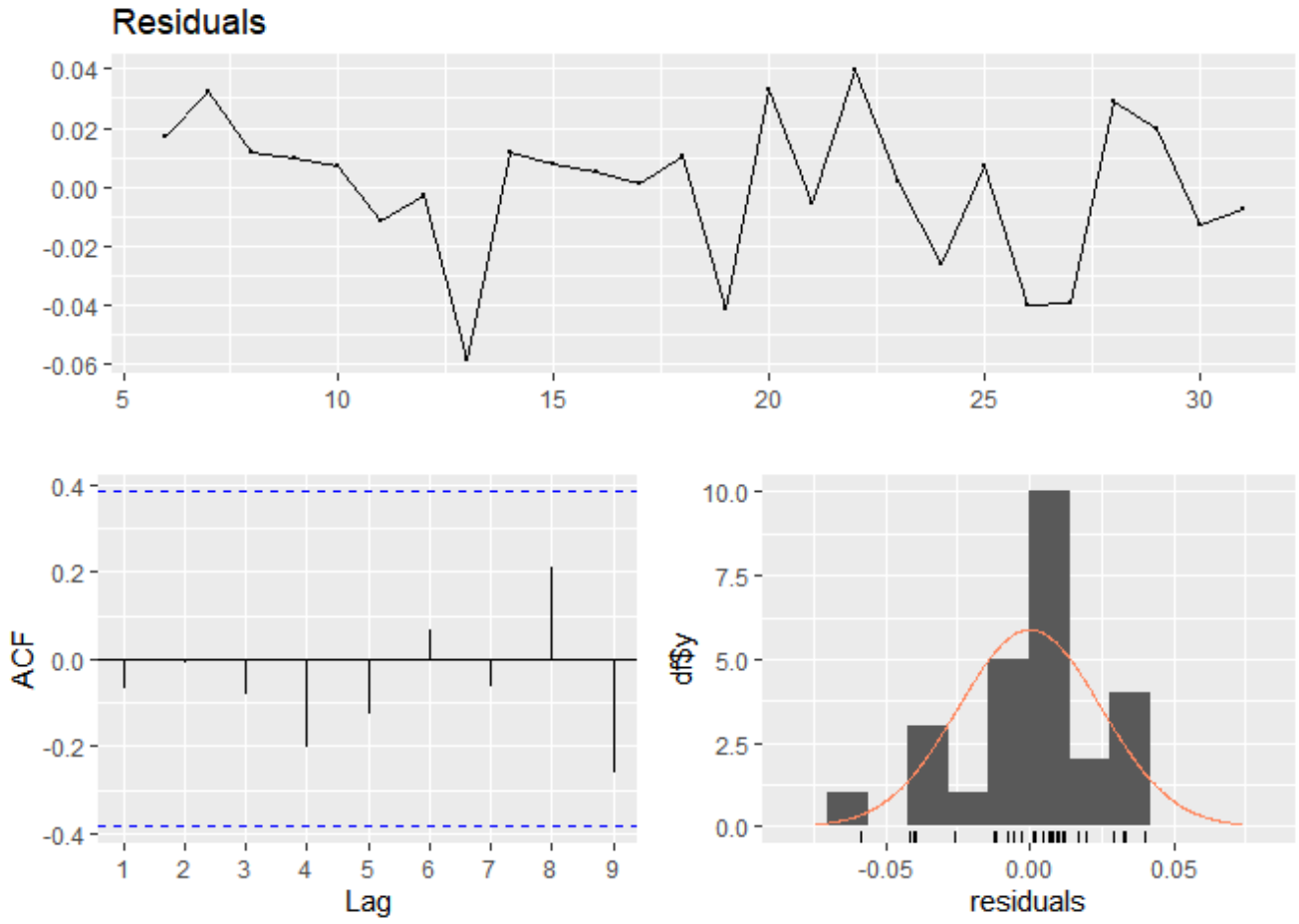

Figure 78

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

```
bgtest(model4.4.3$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model4.4.3$model
## LM test = 1.6982, df = 1, p-value = 0.1925
```

```
shapiro.test(model4.4.3$model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model4.4.3$model$residuals
## W = 0.94347, p-value = 0.1625
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 78` . The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are not significant. Furthermore, from the Shapiro-Wilk test we can conclude that we do not reject null hypothesis of normality as p-value > 0.05.

```
VIF.model4.4.3<-vif(model4.4.3$model)
VIF.model4.4.3
```

```
##       X.t L(X.t, 1) L(X.t, 2) L(X.t, 3) L(X.t, 4) L(X.t, 5) L(y.t, 1) L(y.t, 2)
##  1.884005  2.448098  2.454577  2.374717  2.484701  1.966329  2.901087  3.066142
## L(y.t, 3) L(y.t, 4) L(y.t, 5)
##  3.162682  2.626466  3.015791
```

```
VIF.model4.4.3>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
```

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the vif function, we can observe that there no issue with multicollinearity as VIF values are all less than 10.

3.7.0 Dynamic Lag Models

We will fit the dynamic linear models as the model is for assessing the effect of the intervention on the time series. We will compute the following dynamic lag models and choose the best one with the lowest mase value for further analysis.

```
Y.t=log(rbo.rbo.ts)
T=3
P.t=1*(seq(rbo.rbo.ts)==T)
P.t.1=Lag(P.t,+1)

model5.1 = dynlm(Y.t ~ L(Y.t , k = 1 ) + P.t.1 + P.t + trend(Y.t))
model5.2 = dynlm(Y.t ~ L(Y.t , k = 1 ) + P.t.1 + P.t + L(Y.t , k = 2 )
                 + trend(Y.t))

model5.mase<-MASE(lm(model5.1),lm(model5.2))
arrange(model5.mase,MASE)
```

```
##               n      MASE
## lm(model5.1) 30 0.6548883
## lm(model5.2) 29 0.6862023
```

Here in this output we can see that `model5.1` has the best MASE value so we will use this model for further analysis.

```
summary(model5.1)
```

```
##
## Time series regression with "ts" data:
## Start = 1985, End = 2014
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, k = 1) + P.t.1 + P.t + trend(Y.t))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.115888 -0.016278  0.000912  0.020945  0.075368
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.122700   0.046850  -2.619  0.01477 *
## L(Y.t, k = 1)  0.520202   0.172122   3.022  0.00572 **
## P.t.1         -0.071334   0.048830  -1.461  0.15651
## P.t            0.112167   0.047121   2.380  0.02523 *
## trend(Y.t)    -0.001626   0.001171  -1.388  0.17724
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0435 on 25 degrees of freedom
## Multiple R-squared:  0.5555, Adjusted R-squared:  0.4844
## F-statistic: 7.812 on 4 and 25 DF,  p-value: 0.0003147
```

By using the `summary` function, we can observe that in `model5.1` the lag weights of the predictors are significant at the 5% statistical level as p-value < 0.05. Furthermore, there is a adjusted R-square value of 0.4844, which accounts for 48.44% of the variability in the data. Here we can also see that for the lags, positive lag coefficients show that higher RBO values indicate similarity of the order of the first flowering occurrence. First lag indicates that flowering is not similar as there is a small value.

```
checkresiduals(model5.1)
```

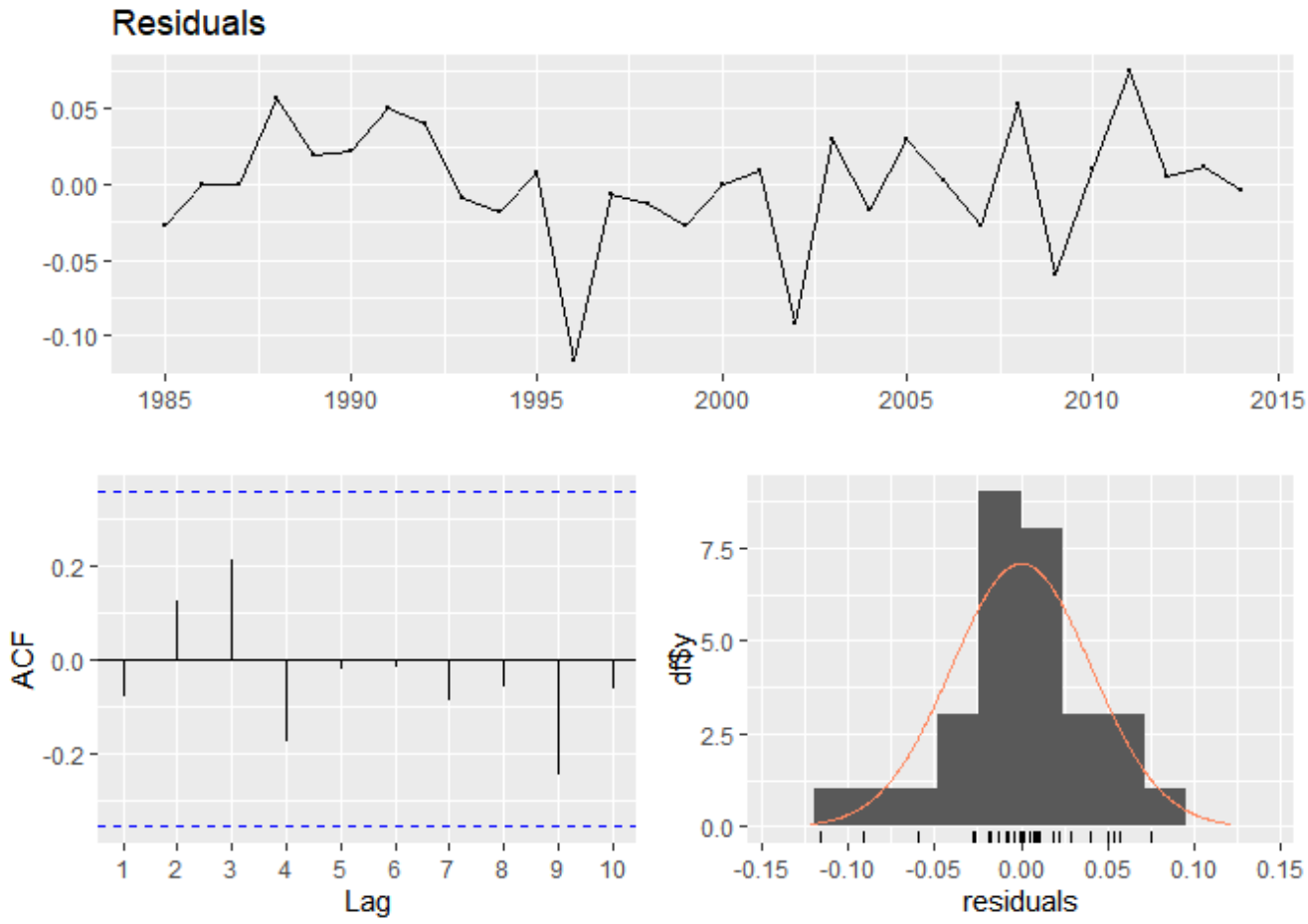

Figure 79

```
##
##  Breusch-Godfrey test for serial correlation of order up to 8
##
## data:  Residuals
## LM test = 4.2148, df = 8, p-value = 0.8372
```

```
shapiro.test(residuals(model5.1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(model5.1)
## W = 0.93435, p-value = 0.06415
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 79`. The Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are not significant. Furthermore, from the Shapiro-Wilk test we can conclude that we do not reject null hypothesis of normality as p-value > 0.05.

```
VIF.model5.1<-vif(model5.1)
VIF.model5.1
```

```
## L(Y.t, k = 1)         P.t.1           P.t    trend(Y.t)
##      1.652871      1.217799      1.134044      1.628151
```

```
VIF.model5.1>10
```

```
## L(Y.t, k = 1)         P.t.1           P.t    trend(Y.t)
##         FALSE         FALSE         FALSE         FALSE
```

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the vif function, we can observe that there no issue with multicollinearity as VIF values are all less than 10.

3.8.0 Model Comparison and Selection

With our models selected from each section, we will now compare each models and select the best for further analysis in terms of its mase values.

```
rbo.models<-c("model1.1","model2.2","model3.4","model4.4.3")
rbo.mase<-MASE(model1.1,model2.2,model3.4,model4.4.3)
rbo.mase<-rbo.mase[,-1]
rbo.mase2<-c("model5.1",accuracy(model5.1)[,6])
rbo.comp<-data.frame(rbo.models,rbo.mase)
rbo.comp<-rbind(rbo.comp,rbo.mase2)
rownames(rbo.comp)<-NULL
colnames(rbo.comp)<-c("Model","MASE")
arrange(rbo.comp,MASE)
```

```
##        Model              MASE
## 1   model1.1 0.523527046479962
## 2   model2.2 0.645180426887817
## 3   model5.1 0.654888310187682
## 4 model4.4.3 0.714569812384021
## 5   model3.4 0.840013507894619
```

Based on the lowest MASE values, we can now choose `model1.1` and `model2.2` for further analysis in the next section.

3.9.0 Forecasting

We will first forecast the data for `model1.1` which is a finite DLM with temperature predictor. Here we will also use the covariate values provided for point forecasting.

```
rbo.covar[,2]
```

```
## # A tibble: 4 × 1
##   Temperature
##         <dbl>
## 1        20.7
## 2        20.5
## 3        20.5
## 4        20.6
```

```
frc.model1.1<-forecast(model=model1.1,
                       x=c(20.74,20.49,20.52,20.56),
                       h=3)$forecast
frc.model1.1
```

```
## [1] 0.7323402 0.7106262 0.7145942
```

In the output above, we are able to see the point forecast for FFD based on `model1.1`. However, due to errors in the package, we are unable to output the 95% confidence intervals for `model1.1`. We will now plot the three year ahead forecast plots.

```
y.extended<-c(rbo$RBO,frc.model1.1)
plot(ts(y.extended,start=1984),type="o",col="red",ylim=c(0,1),
     xlab="Year",ylab="RBO",main="RBO 3 Year Forecast (Finite DLM)")
lines(ts(as.vector(rbo$RBO),start=1984),col="black",type="o")
legend("bottomleft",lty=1,pch=1,cex=0.7,col=c("black","red"),
       c("RBO Data","Finite DLM Forecast"))
```

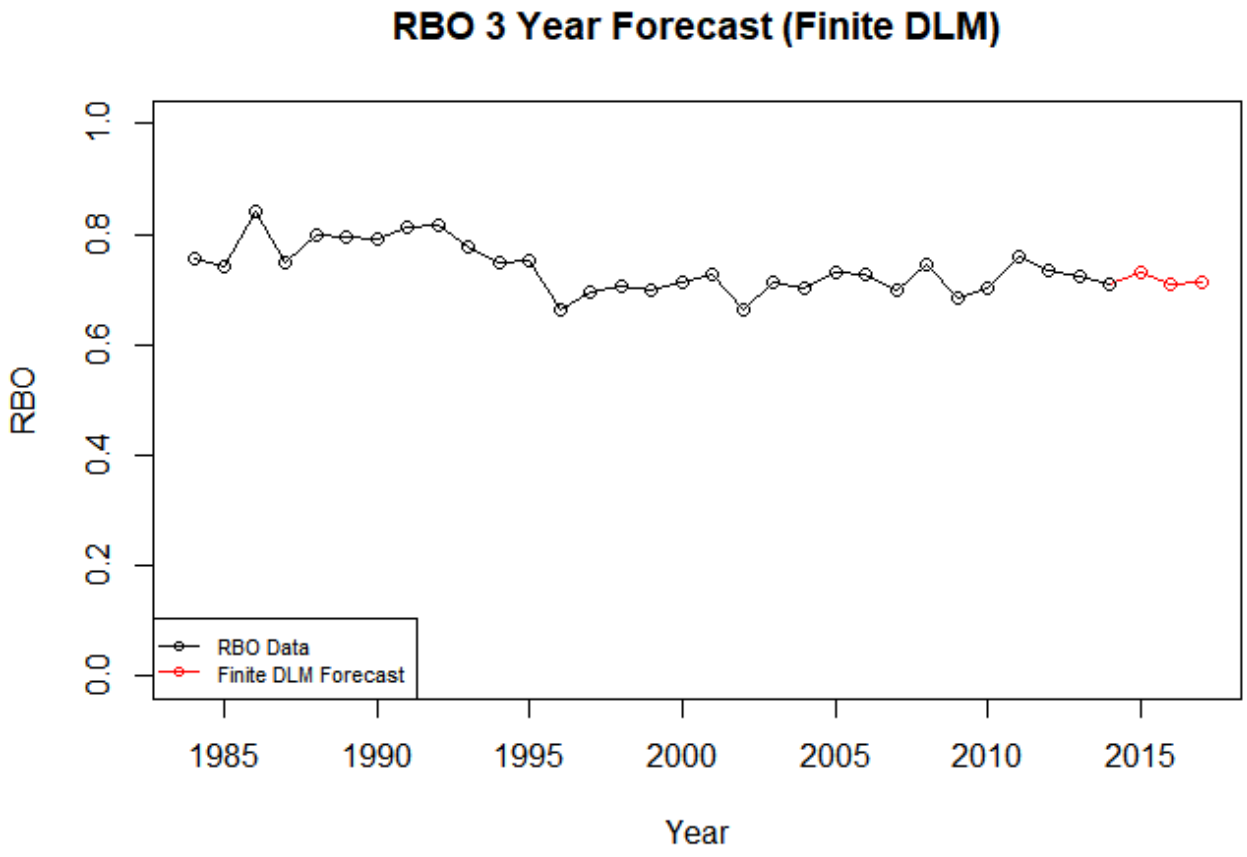

Figure 80

From `Figure 80` above, we can see that finite dlm fit and the plots for the next three years prediction (2015-2017). Next we will forecast the data for `model2.2` which is a Polynomial DLM with rainfall as a predictor.

```
rbo.covar[,3]
```

```
## # A tibble: 4 × 1
##   Rainfall
##      <dbl>
## 1     2.27
## 2     2.38
## 3     2.26
## 4     2.27
```

```
frc.model2.2<-forecast(model=model2.2,
                       x=c(20.74,20.49,20.52,20.56),
                       h=3,
                       interval=TRUE)$forecast
frc.model2.2
```

```
##       Lower Estimate    Upper
## 1 0.9519286 1.002706 1.052334
```

```
## 2 1.1166659 1.171438 1.218347
## 3 1.1999095 1.254158 1.305560
```

In the forecast output above, we can see the 95% lower bound, point forecast and 95% upper bound forecast data for RBO. We will now plot the model against the original data.

```
y.extended2<-c(rbo$RBO,frc.model2.2$Estimate)
plot(ts(y.extended2,start=1984),type="o",col="red",ylim=c(0,1.5),
     xlab="Year",ylab="RBO",main="RBO 3 Year Forecast (Polynomial DLM)")
lines(ts(as.vector(rbo$RBO),start=1984),col="black",type="o")
legend("bottomleft",lty=1,pch=1,cex=0.7,col=c("black","red"),
       c("RBO Data","Polynomial DLM Forecast"))
```

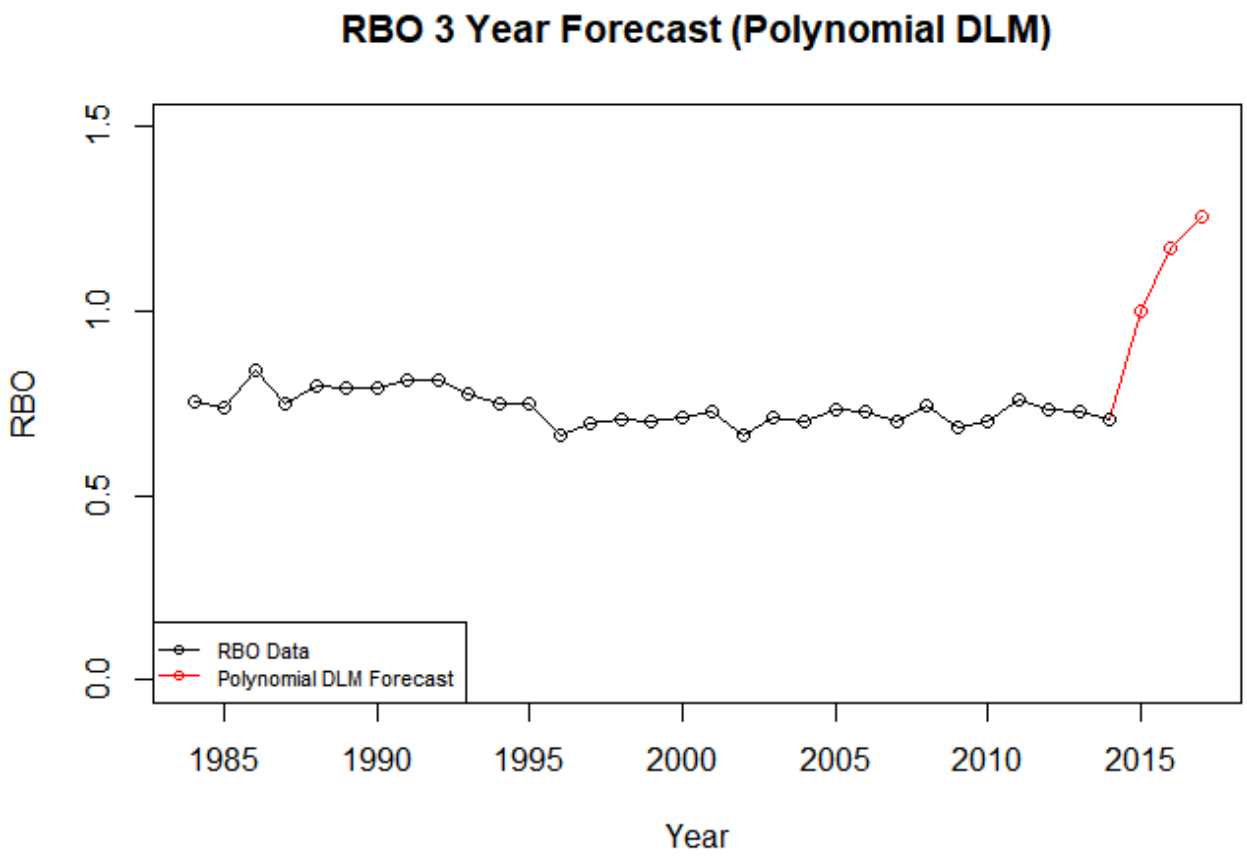

Figure 81

From `Figure 57` above, we can see that Polynomial DLM fit and the plots for the next three years prediction (2015-2017). Interestingly, compared to Finite DLM forecast, the point forecast shows that the data would increase over the next four years.

3.10.0 Conclusion

From the previous sections, we have analyzed that the best models to forecast the next three years of RBO values are Finite Distributed Lag Model with temperature as a predictor for the data. With our forecast plots, we can conclude that Finite DLM predictor plot is more stable compared to the Polynomial DLM plot.

4.0.0 Task 3 (B)

4.1.0 Dynamic Lag Model Modelling

We will now perform a dynamic lag modelling using the `dynlm` package to obtain the 3 year ahead forecasts for droughts in consideration. Since there are 13 years of droughts from 1996-2009, we shall input T=13 as an argument in the model.

```
Y.t=log(rbo.rbo.ts)
T=13
P.t=1*(seq(rbo.rbo.ts)==T)
P.t.1=Lag(P.t,+1)

dro.model1.1 = dynlm(Y.t ~ L(Y.t , k = 1 ) + P.t.1 + P.t + trend(Y.t))
dro.model1.2 = dynlm(Y.t ~ L(Y.t , k = 1 ) + P.t.1 + P.t + L(Y.t , k = 2 )
                + trend(Y.t))

model.mase<-MASE(lm(dro.model1.1),lm(dro.model1.2))
arrange(model.mase,MASE)
```

```
##                   n      MASE
## lm(dro.model1.2) 29 0.6793374
## lm(dro.model1.1) 30 0.6982148
```

From the output above we can see that `dro.model1.2` contains the lowest MASE values hence we will select this model for further analysis.

```
summary(dro.model1.2)
```

```
##
## Time series regression with "ts" data:
## Start = 1986, End = 2014
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, k = 1) + P.t.1 + P.t + L(Y.t, k = 2) +
##     trend(Y.t))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.092874 -0.017957 -0.000999  0.019701  0.083729
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.115797   0.047982  -2.413  0.02417 *
```

```
## L(Y.t, k = 1)  0.133342   0.185797   0.718  0.48018
## P.t.1         -0.064887   0.047607  -1.363  0.18609
## P.t           -0.126084   0.041910  -3.008  0.00626 **
## L(Y.t, k = 2)  0.319041   0.167720   1.902  0.06974 .
## trend(Y.t)    -0.002749   0.001213  -2.267  0.03312 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04097 on 23 degrees of freedom
## Multiple R-squared:  0.6372, Adjusted R-squared:  0.5583
## F-statistic: 8.079 on 5 and 23 DF,  p-value: 0.0001609
```

By using the `summary` function, we can observe that in this model the lag weights of the predictors are significant at the 5% statistical level as p-value < 0.05. Furthermore, there is a adjusted R-square value of 0.5583, which accounts for 55.83% of the variability in the data. Here we can also see that for the lags, positive lag coefficients show that higher RBO values indicate similarity of the order of the first flowering occurrence. First lag indicates that flowering is not similar as there is a small value.

```
checkresiduals(dro.model1.2)
```

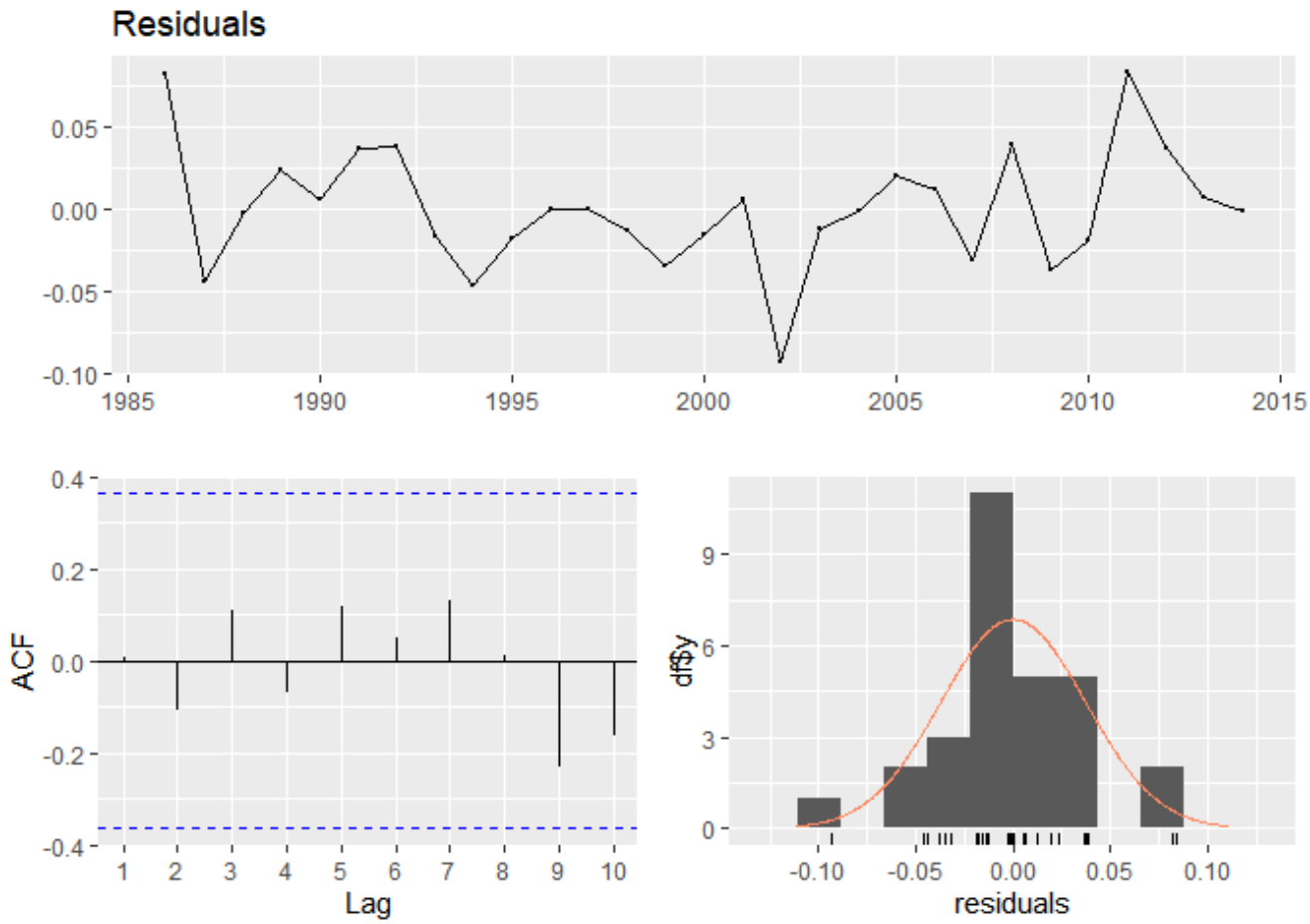

Figure 82

```
##
##  Breusch-Godfrey test for serial correlation of order up to 9
##
```

```
## data:  Residuals
## LM test = 13.672, df = 9, p-value = 0.1345
```

```
shapiro.test(residuals(dro.model1.2))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(dro.model1.2)
## W = 0.96303, p-value = 0.3895
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 79`. The Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are not significant. Furthermore, from the Shapiro-Wilk test we can conclude that we do not reject null hypothesis of normality as p-value > 0.05.

```
dro.VIF.model<-vif(dro.model1.2)
dro.VIF.model
```

```
## L(Y.t, k = 1)         P.t.1           P.t L(Y.t, k = 2)    trend(Y.t)
##      2.159735      1.303569      1.010226      1.764767      1.777855
```

```
dro.VIF.model>10
```

```
## L(Y.t, k = 1)         P.t.1           P.t L(Y.t, k = 2)    trend(Y.t)
##         FALSE         FALSE         FALSE         FALSE         FALSE
```

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the vif function, we can observe that there no issue with multicollinearity as VIF values are all less than 10.

4.2.0 Forecasting