Introduction

This report describes the implementation of three data analysis tasks, where multiple forecasting models are created for each and the ideal model chosen based on appropriate test statistics. The three tasks are described below:

- **Task 1:** Implementing multiple time series regression models, state space models, and exponential smoothing models to create 4 weeks ahead forecasts for weekly mortality rates in France in terms of R squared, AIC, BIC, and MASE. Each model is to be validated using the appropriate test statistics and the best model is chosen. **Dynamic linear models were not explored in this report since there was no obvious intervention point in the time series for the variables.**
- **Task 2:** Implementing multiple time series regression models, state space models, and exponential smoothing models to create 4 years ahead forecasts for first flowering days (FFD) for in terms of R squared, AIC, BIC, and MASE. Each model is to be validated using the appropriate test statistics and the best model is chosen.**Dynamic linear models were not explored in this report since there was no obvious intervention point in the time series for the variables.**
- **Task 3:** Exploring the Rank-based Order similarity metric and climate conditions by implementing various forecasting models, including time series regression models, state space models, and exponential smoothing models to create 3 years ahead forecasts for the Rank-based Order similarity metric (RBO) in terms of R squared, AIC, BIC, and MASE. Each model is to be validated using the appropriate test statistics and the best model is chosen.

Task 1

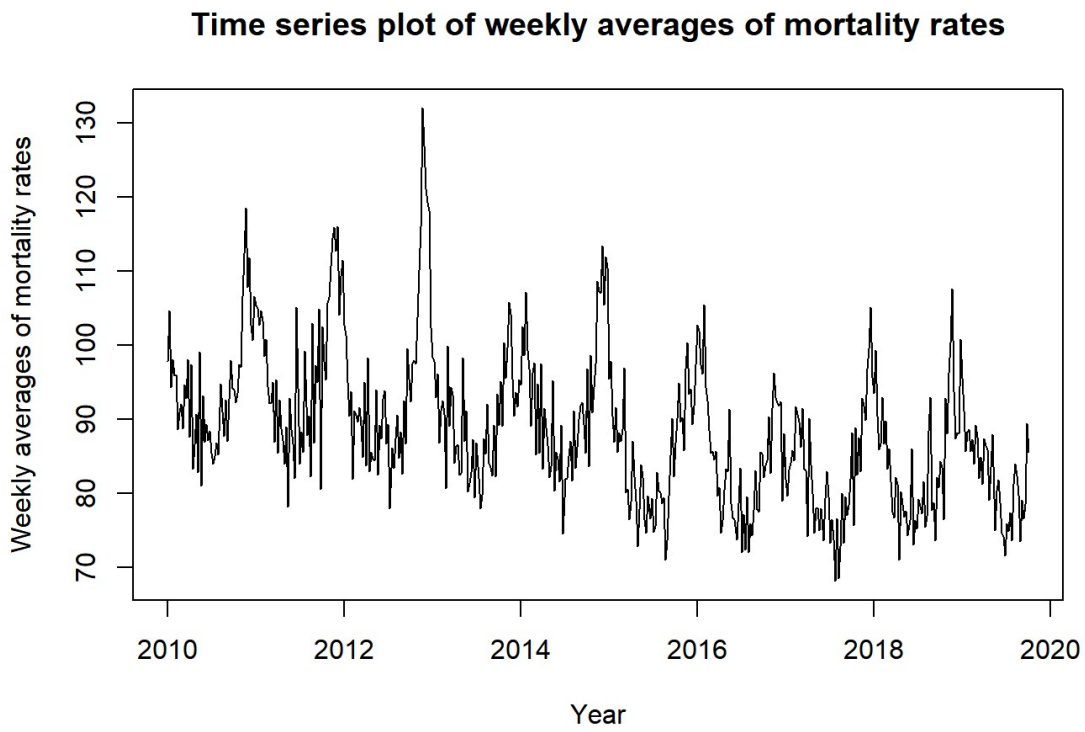
Reading Data and Pre-processing

The dataset is read in and each variable is converted to a time series object. All the series are plotted and each series is seen to be roughly stationary with seasonal peaks.

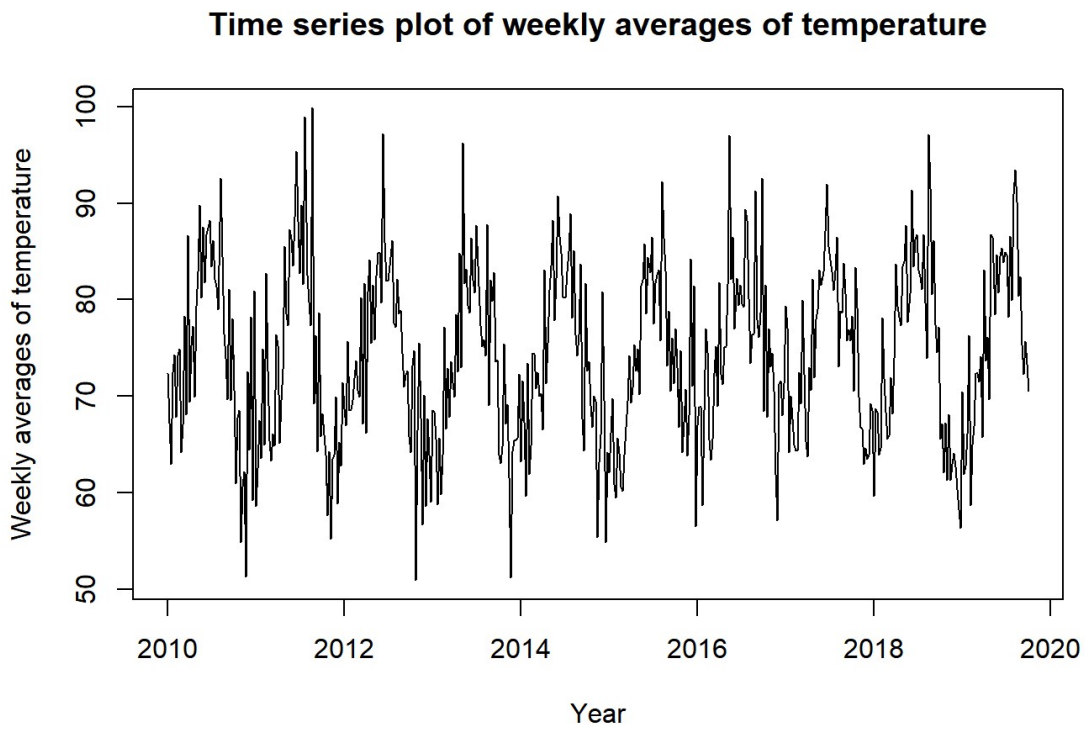
```
setwd("C:/Work/Master in Analytics/Semester 2/Forecasting/Forecasting Final Assignment")
mortdata <- read.csv("mort.csv")
head(mortdata)
```

```
##   X mortality  temp chem1 chem2 particle.size
## 1 1     97.85 72.38 11.51  3.37         72.72
## 2 2    104.64 67.19  8.92  2.59         49.60
## 3 3     94.36 62.94  9.48  3.29         55.68
## 4 4     98.05 72.49 10.28  3.04         55.16
## 5 5     95.85 74.25 10.57  3.39         66.02
## 6 6     95.98 67.88  7.99  2.57         44.01
```

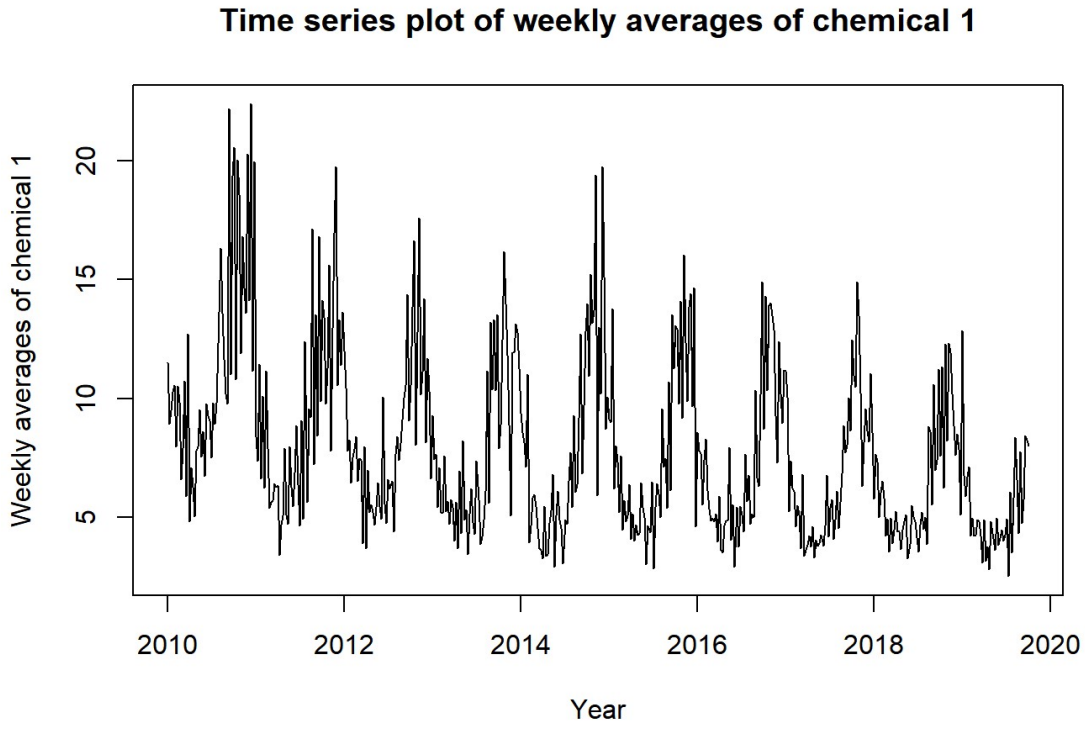
```
mortality <- ts(mortdata$mortality, start=c(2010,1), frequency=52)
temp <- ts(mortdata$temp, start=c(2010,1), frequency=52)
chem1 <- ts(mortdata$chem1, start=c(2010,1), frequency=52)
chem2 <- ts(mortdata$chem2, start=c(2010,1), frequency=52)
particle.size <- ts(mortdata$particle.size, start=c(2010,1),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


plot(mortality, ylab='Weekly averages of mortality rates',xlab='Year',
     main = "Time series plot of weekly averages of mortality rates")
```



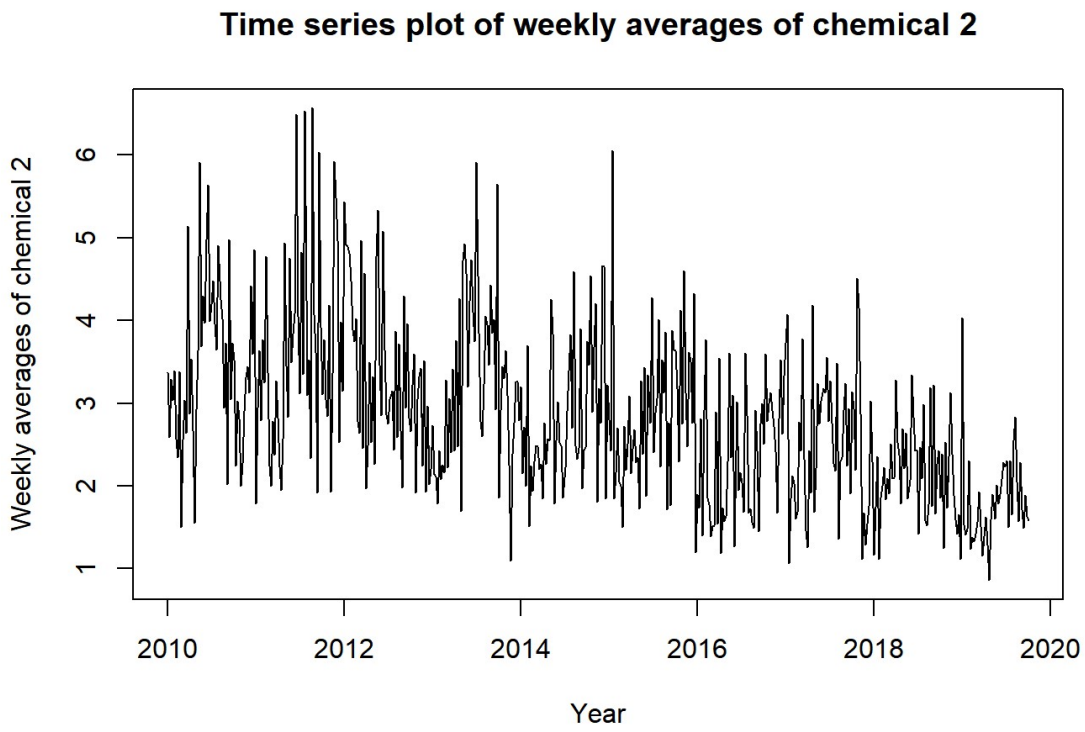

```
plot(temp, ylab='Weekly averages of temperature',xlab='Year',
     main = "Time series plot of weekly averages of temperature")
```



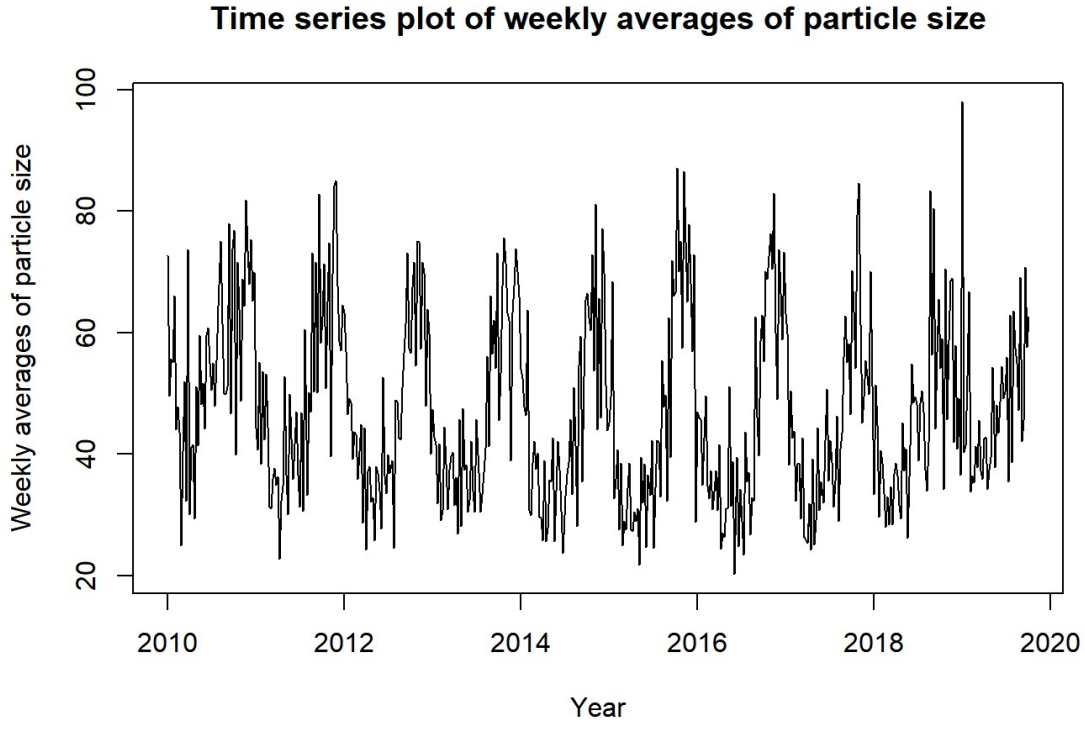

```
plot(chem1, ylab='Weekly averages of chemical 1',xlab='Year',
     main = "Time series plot of weekly averages of chemical 1")
```




```
plot(chem2, ylab='Weekly averages of chemical 2',xlab='Year',
     main = "Time series plot of weekly averages of chemical 2")
```



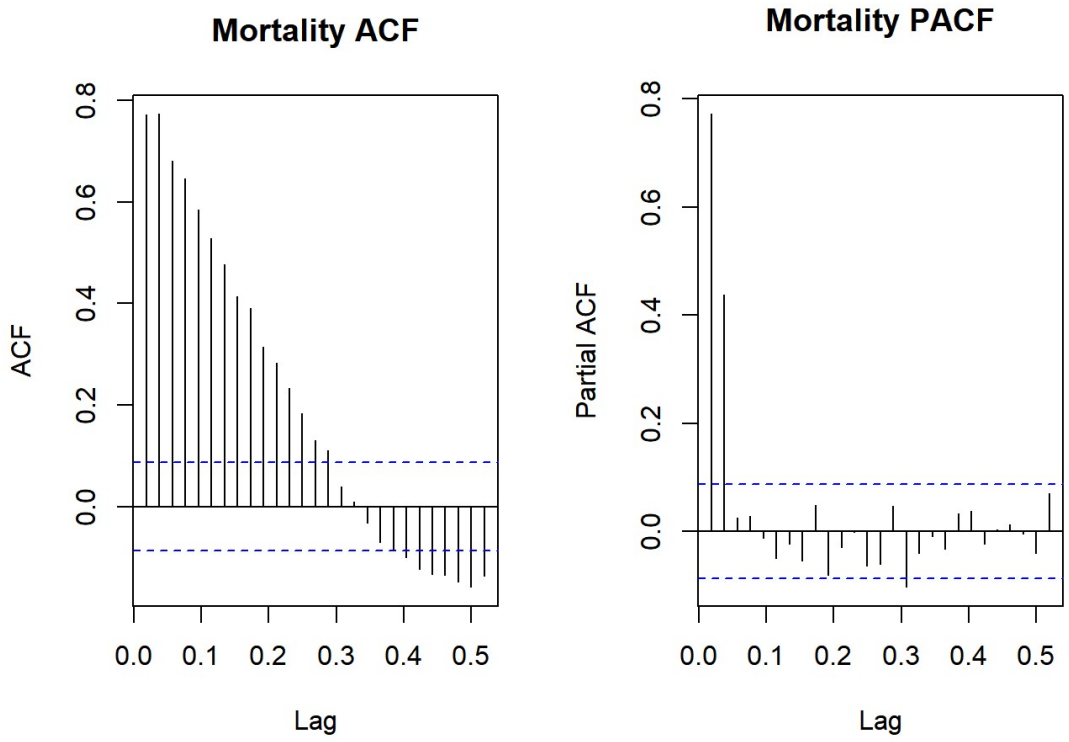

```
plot(particle.size, ylab='Weekly averages of particle size',xlab='Year',
     main = "Time series plot of weekly averages of particle size")
```



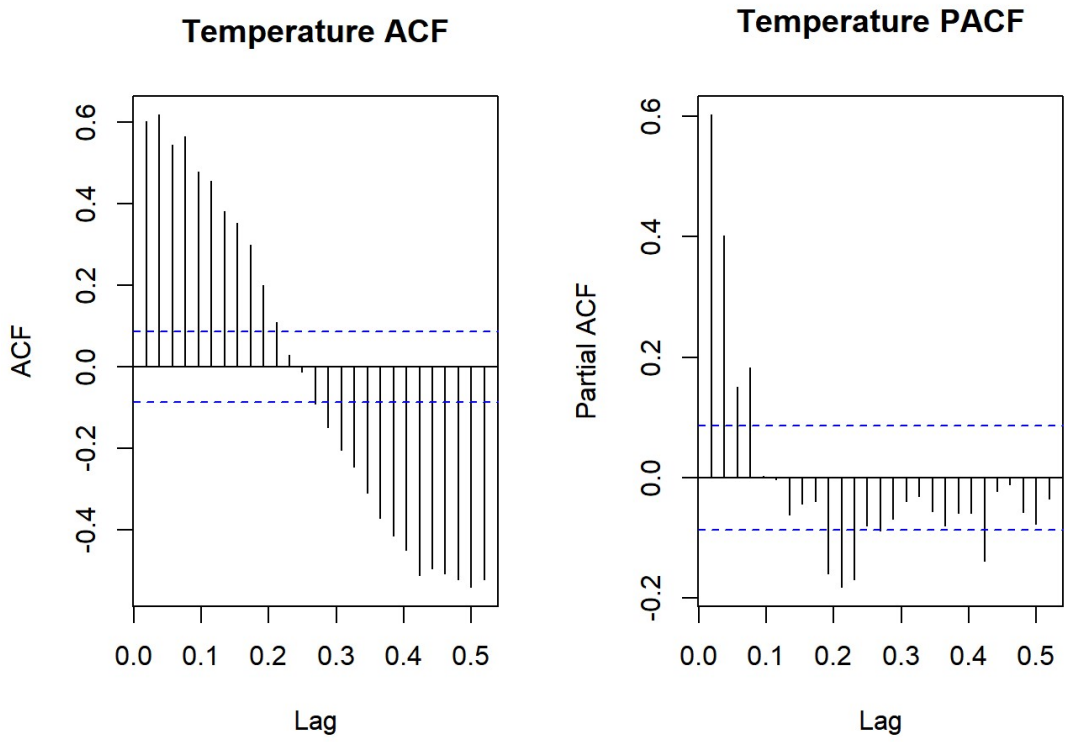
Exploring Stationarity of Variables

Plotting the ACF and PACF for the series demonstrates similar results for all variables. All variables demonstrate a gradually decreasing ACF plot , with most peaks lying above the 95% confidence interval, and the PACF plot shows multiple significant lags. This suggests that the series is non-stationary.

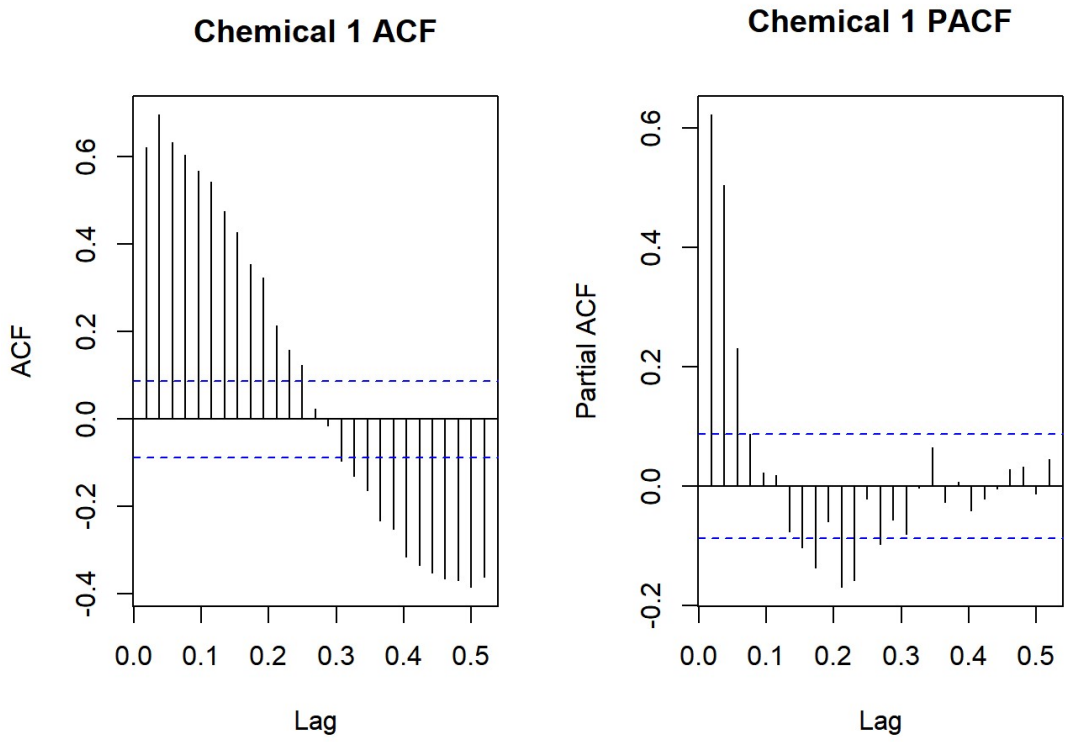
```
par(mfrow=c(1,2))
acf(mortality, max.lag = 24, main="Mortality ACF")
pacf(mortality, max.lag = 24, main = "Mortality PACF")
```



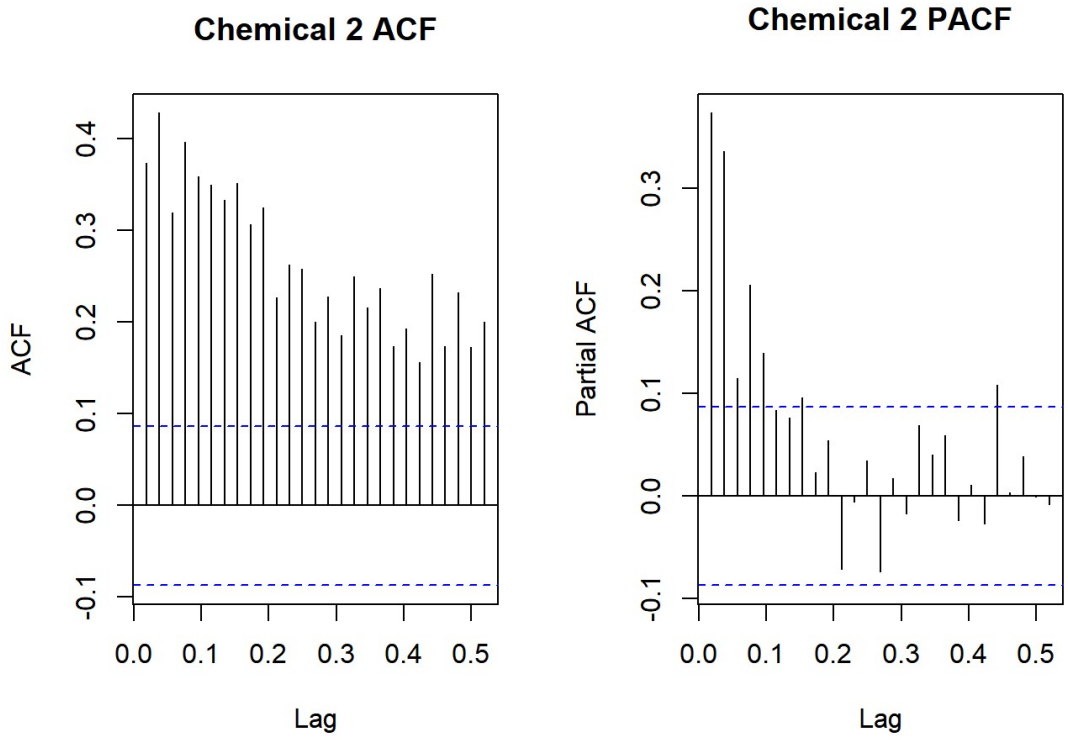

```
par(mfrow=c(1,2))
acf(temp, max.lag = 24, main="Temperature ACF")
pacf(temp, max.lag = 24, main = "Temperature PACF")
```



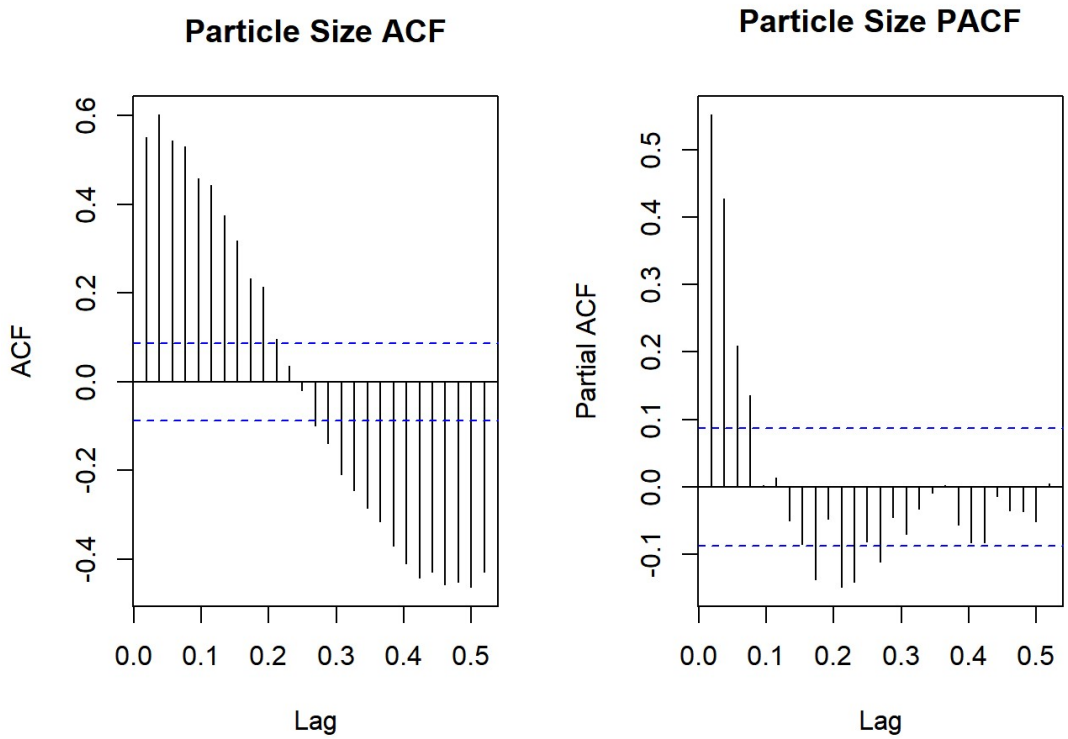

```
par(mfrow=c(1,2))
acf(chem1, max.lag = 24, main="Chemical 1 ACF")
pacf(chem1, max.lag = 24, main = "Chemical 1 PACF")
```




```
par(mfrow=c(1,2))
acf(chem2, max.lag = 24, main="Chemical 2 ACF")
pacf(chem2, max.lag = 24, main = "Chemical 2 PACF")
```




```
par(mfrow=c(1,2))
acf(particle.size, max.lag = 24, main="Particle Size ACF")
pacf(particle.size, max.lag = 24, main = "Particle Size PACF")
```



For further confirmation of non-stationarity in the series, an Augmented Dicky-Fuller test and Phillips-Perron unit root test are performed on the series, the results of which are given below:

- **Mortality:** For the ADF test, the low p-value and low absolute value of the critical value compared to the critical values at 1%, 5%, and 10% significance levels, signify that the series is stationary. However, the Phillip's Perron unit test produces a z tau value that is

more extreme than the critical values, signifying that the data is non-stationary since the null hypothesis cannot be rejected.

- **Temperature:** For the ADF test, the low p-value and low absolute value of the critical value compared to the critical values at 1%, 5%, and 10% significance levels, signify that the series is stationary. However, the Phillip's Perron unit test produces a z tau value that is more extreme than the critical values, signifying that the data is non-stationary since the null hypothesis cannot be rejected.
- **Chemical 1:** For the ADF test, the absolute value of the critical value is higher compared to the critical values at 5%, and 10% significance levels, signifying that the series is non-stationary since the null hypothesis cannot be rejected. The Phillip's Perron unit test produces a z tau value that is more extreme than the critical values, signifying that the data is non-stationary since the null hypothesis cannot be rejected.
- **Chemical 2:** For the ADF test, the low p-value and low absolute value of the critical value compared to the critical values at 1%, 5%, and 10% significance levels, signify that the series is stationary. However, the Phillip's Perron unit test produces a z tau value that is more extreme than the critical values, signifying that the data is non-stationary since the null hypothesis cannot be rejected.
- **Particle Size** For the ADF test, the low p-value and low absolute value of the critical value compared to the critical values at 1%, 5%, and 10% significance levels, signify that the series is stationary. However, the Phillip's Perron unit test produces a z tau value that is more extreme than the critical values, signifying that the data is non-stationary since the null hypothesis cannot be rejected.

```
adf.mortality = ur.df(mortality, type = "none", lags = 1, selectlags = "AIC")
summary(adf.mortality)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.1256  -3.7511   0.0502   3.6939  20.9358
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.001961   0.002903  -0.676      0.5
## z.diff.lag -0.505434   0.038383 -13.168   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.823 on 504 degrees of freedom
## Multiple R-squared:  0.2574, Adjusted R-squared:  0.2545
## F-statistic: 87.35 on 2 and 504 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -0.6755
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

```
pp.mortality = ur.pp(mortality, type = "Z-tau", lags = "short")
summary(pp.mortality)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.618  -4.155  -0.370   4.019  22.264
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 20.22464    2.52168    8.02 7.41e-15 ***
## y.l1         0.77173    0.02825   27.32  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.359 on 505 degrees of freedom
## Multiple R-squared:  0.5964, Adjusted R-squared:  0.5956
## F-statistic: 746.3 on 1 and 505 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic, type: Z-tau  is: -7.9408
##
##          aux. Z statistics
## Z-tau-mu            7.8809
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.445446 -2.867533 -2.569954
```

```
adf.temp = ur.df(temp, type = "none", lags = 1, selectlags = "AIC")
summary(adf.temp)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -22.8049  -4.1532  -0.1683   4.4153  21.5154
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.002683   0.004093  -0.656    0.512
## z.diff.lag -0.519365   0.038052 -13.649   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.878 on 504 degrees of freedom
## Multiple R-squared:  0.2719, Adjusted R-squared:  0.269
## F-statistic: 94.12 on 2 and 504 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -0.6556
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

```
pp.temp = ur.pp(temp, type = "Z-tau", lags = "short")
summary(pp.temp)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.6204  -4.6804  -0.0509   4.6633  23.7841
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 29.54652    2.65853   11.11   <2e-16 ***
## y.l1         0.60211    0.03554   16.94   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.211 on 505 degrees of freedom
## Multiple R-squared:  0.3624, Adjusted R-squared:  0.3612
## F-statistic: 287.1 on 1 and 505 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic, type: Z-tau  is: -12.0726
##
##          aux. Z statistics
## Z-tau-mu           11.9834
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.445446 -2.867533 -2.569954
```

```
adf.chem1 = ur.df(chem1, type = "none", lags = 1, selectlags = "AIC")
summary(adf.chem1)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5785 -1.1611  0.0874  1.5744 12.4693
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.02973    0.01351  -2.201   0.0282 *
## z.diff.lag -0.58331    0.03612 -16.150   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.613 on 504 degrees of freedom
## Multiple R-squared:  0.3641, Adjusted R-squared:  0.3616
## F-statistic: 144.3 on 2 and 504 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -2.2012
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

```
pp.chem1 = ur.pp(chem1, type = "Z-tau", lags = "short")
summary(pp.chem1)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.1240 -1.7605 -0.5254  1.3137 13.1374
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.98653    0.30487   9.796   <2e-16 ***
## y.l1         0.62152    0.03481  17.855   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.949 on 505 degrees of freedom
## Multiple R-squared:  0.387,  Adjusted R-squared:  0.3858
## F-statistic: 318.8 on 1 and 505 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic, type: Z-tau  is: -11.7515
##
##          aux. Z statistics
## Z-tau-mu           10.5921
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.445446 -2.867533 -2.569954
```

```
adf.chem2 = ur.df(chem2, type = "none", lags = 1, selectlags = "AIC")
summary(adf.chem2)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6726 -0.4772  0.0717  0.6809  3.6885
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.03645    0.01466  -2.487   0.0132 *
## z.diff.lag -0.52667    0.03781 -13.931   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9815 on 504 degrees of freedom
## Multiple R-squared:  0.3054, Adjusted R-squared:  0.3027
## F-statistic: 110.8 on 2 and 504 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -2.4868
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

```
pp.chem2 = ur.pp(chem2, type = "Z-tau", lags = "short")
summary(pp.chem2)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2311 -0.6721 -0.1768  0.5706  3.9163
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.77789    0.12537  14.181   <2e-16 ***
## y.l1         0.37427    0.04132   9.057   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9761 on 505 degrees of freedom
## Multiple R-squared:  0.1397, Adjusted R-squared:  0.138
## F-statistic: 82.03 on 1 and 505 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic, type: Z-tau  is: -16.8389
##
##          aux. Z statistics
## Z-tau-mu           15.7764
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.445446 -2.867533 -2.569954
```

```
adf.particle.size = ur.df(particle.size, type = "none", lags = 1, selectlags = "AIC")
summary(adf.particle.size)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -33.961  -6.857   0.668   8.580  55.164
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.01843    0.01070  -1.723   0.0855 .
## z.diff.lag -0.54545    0.03720 -14.663   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.83 on 504 degrees of freedom
## Multiple R-squared:  0.313,  Adjusted R-squared:  0.3103
## F-statistic: 114.8 on 2 and 504 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -1.7232
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

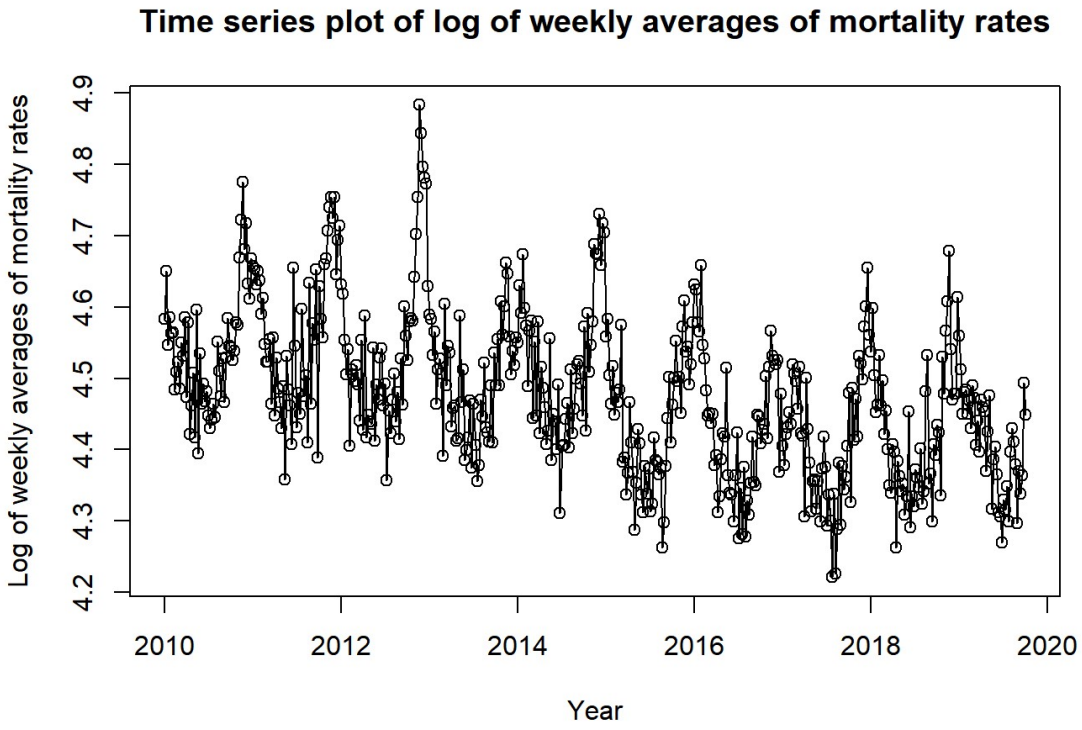
```
pp.particle.size = ur.pp(particle.size, type = "Z-tau", lags = "short")
summary(pp.particle.size)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -34.935  -8.853  -1.172   7.184  56.511
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 21.16583    1.83956   11.51   <2e-16 ***
## y.l1         0.55288    0.03698   14.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.59 on 505 degrees of freedom
## Multiple R-squared:  0.3068, Adjusted R-squared:  0.3054
## F-statistic: 223.5 on 1 and 505 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic, type: Z-tau  is: -13.3151
##
##          aux. Z statistics
## Z-tau-mu           12.6758
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.445446 -2.867533 -2.569954
```

Applying Transformations to Mortality

Log transformations are performed to the series to see if any improvements occur to the Phillips Perron test. Only the differencing causes the Philips Perron root test to produce a less extreme critical value compared to the critical values at 1%, 5%, and 10% significance levels.

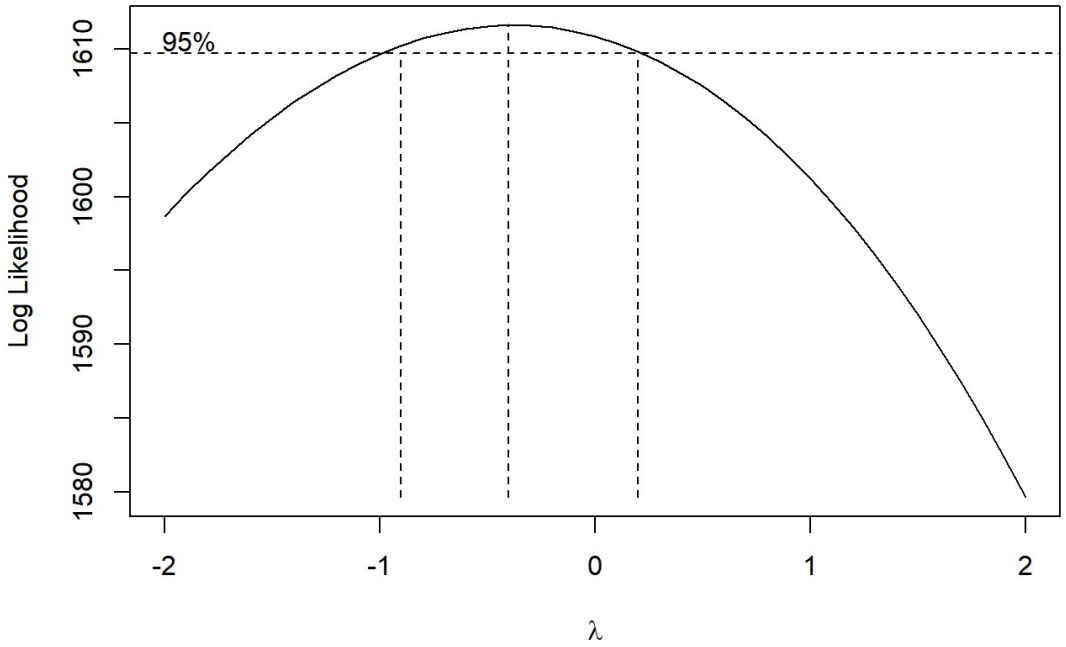
```
# Trying a log transformation to make the series stationary
log.mortality = log(mortality)
plot(log.mortality,ylab='Log of weekly averages of mortality rates',xlab='Year',type='o',
     main = "Time series plot of log of weekly averages of mortality rates")
```




```
pp.log = ur.pp(log.mortality, type = "Z-tau", lags = "short")
summary(pp.log)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.222136 -0.047468 -0.001654  0.047125  0.231176
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.07411    0.12930   8.307 9.08e-16 ***
## y.l1         0.76014    0.02886  26.341  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0712 on 505 degrees of freedom
## Multiple R-squared:  0.5788, Adjusted R-squared:  0.5779
## F-statistic: 693.8 on 1 and 505 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic, type: Z-tau  is: -8.2043
##
##          aux. Z statistics
## Z-tau-mu            8.1997
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.445446 -2.867533 -2.569954
```

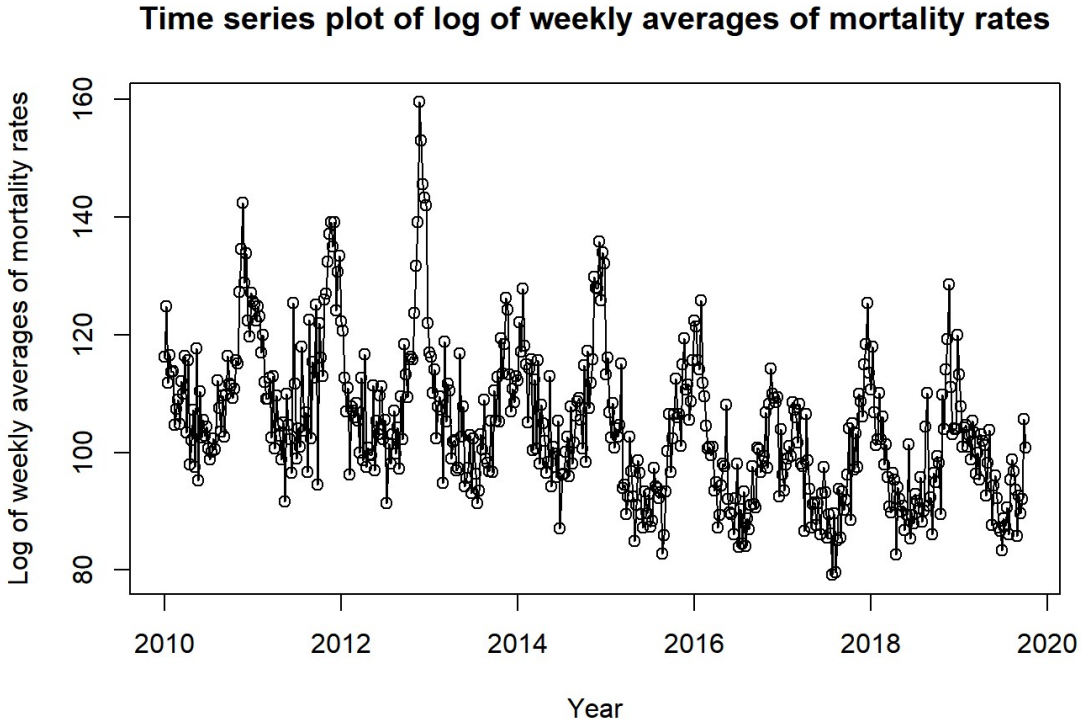
```
# Trying a Box Cox transformation to make the series stationary
bc.mortality = BoxCox.ar(mortality)
```




```
bc.mortality$ci
```

```
## [1] -0.9  0.2
```

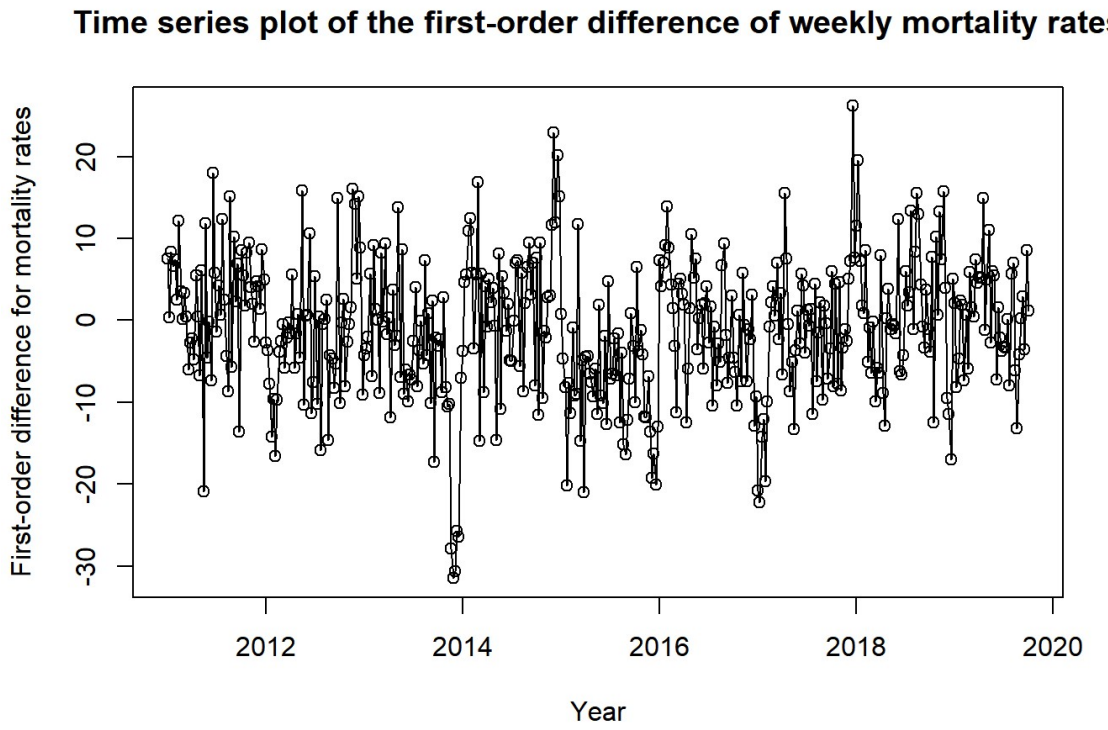
```
bc.mortality = BoxCox(mortality, lambda = 1.05)
plot(bc.mortality, ylab='Log of weekly averages of mortality rates',xlab='Year',type='o',
     main = "Time series plot of log of weekly averages of mortality rates")
```




```
pp.bc = ur.pp(bc.mortality, type = "Z-tau", lags = "short")
summary(pp.bc)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -25.8637  -5.1913  -0.4777   5.0245  28.2269
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 23.83313    2.97840   8.002 8.45e-15 ***
## y.l1         0.77230    0.02822  27.369  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.965 on 505 degrees of freedom
## Multiple R-squared:  0.5973, Adjusted R-squared:  0.5965
## F-statistic: 749.1 on 1 and 505 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic, type: Z-tau  is: -7.9282
##
##          aux. Z statistics
## Z-tau-mu            7.8616
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.445446 -2.867533 -2.569954
```

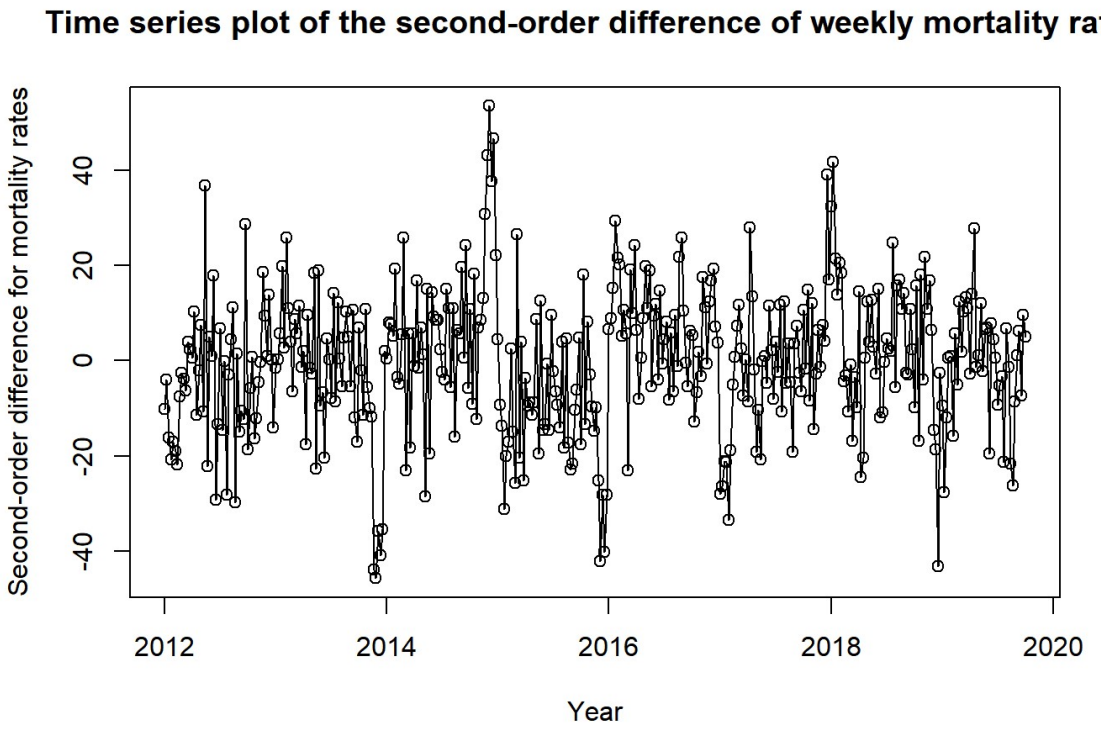
```
# Trying a first-order differencing transformation to make the series stationary
diff1.mortality = diff(mortality, differences = 1, lag = 52)
plot(diff1.mortality, ylab='First-order difference for mortality rates', xlab='Year', type='o',
     main = "Time series plot of the first-order difference of weekly mortality rates")
```




```
pp.diff1 = ur.pp(diff1.mortality, type = "Z-tau", lags = "short")
summary(pp.diff1)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.7801  -5.3851   0.4049   5.0043  24.6196
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.76045    0.38021  -2.000   0.0461 *
## y.l1         0.32284    0.04442   7.269  1.6e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.043 on 453 degrees of freedom
## Multiple R-squared:  0.1044, Adjusted R-squared:  0.1025
## F-statistic: 52.83 on 1 and 453 DF,  p-value: 1.603e-12
##
##
## Value of test-statistic, type: Z-tau  is: -16.1886
##
##          aux. Z statistics
## Z-tau-mu           -2.1216
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.446826 -2.868158 -2.570282
```

```
# Trying a second-order differencing transformation to make the series stationary
diff2.mortality = diff(mortality, differences = 2, lag = 52)
plot(diff2.mortality, ylab='Second-order difference for mortality rates', xlab='Year', type='o',
     main = "Time series plot of the second-order difference of weekly mortality rates")
```




```
pp.diff2 = ur.pp(diff2.mortality, type = "Z-tau", lags = "short")
summary(pp.diff2)
```

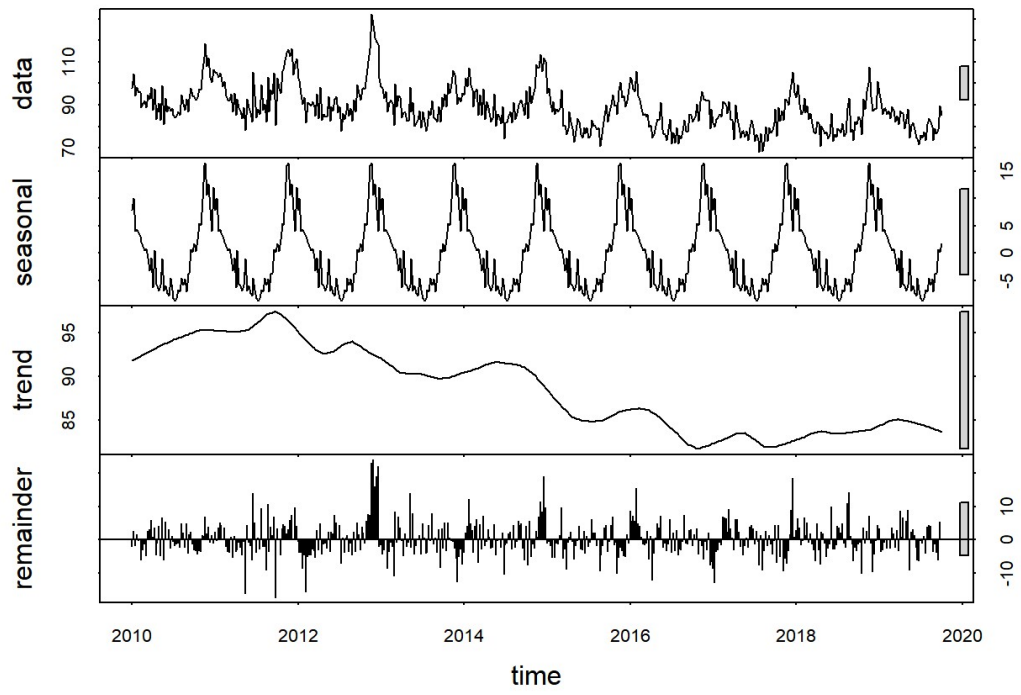
```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -39.788  -9.255   0.368  10.473  40.467
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.1182     0.7180  -0.165    0.869
## y.l1          0.3370     0.0470   7.171 3.62e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.41 on 401 degrees of freedom
## Multiple R-squared:  0.1137, Adjusted R-squared:  0.1114
## F-statistic: 51.42 on 1 and 401 DF,  p-value: 3.618e-12
##
##
## Value of test-statistic, type: Z-tau  is: -14.983
##
##          aux. Z statistics
## Z-tau-mu           -0.1784
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.448566 -2.868946 -2.570696
```

Verifying Seasonality

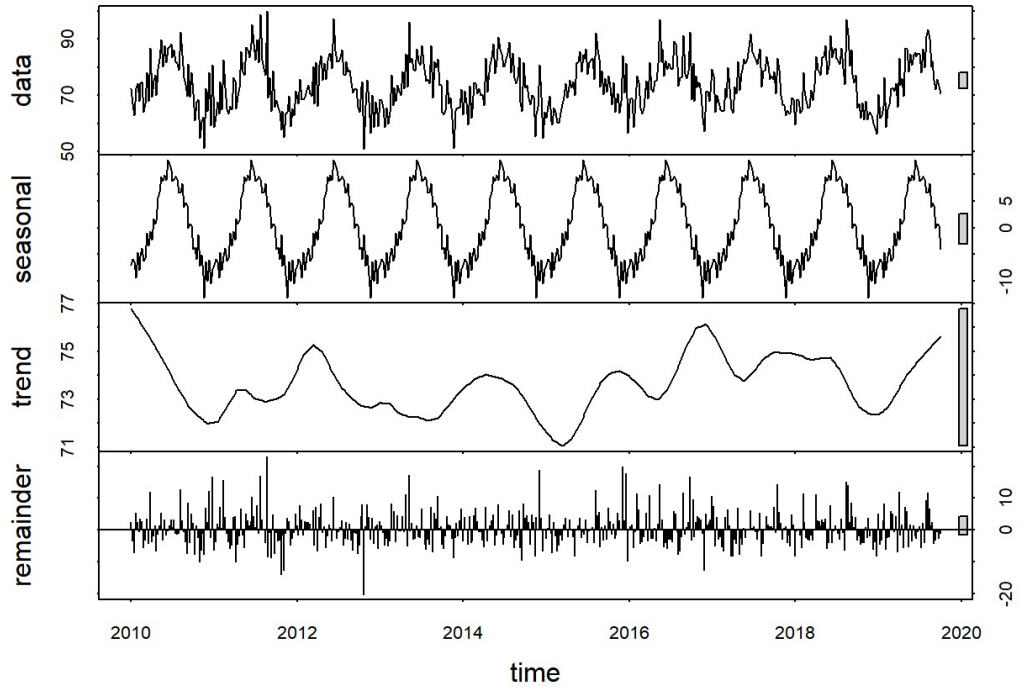
The seasonal components of the series are explored using the STL function. The results are interpreted below:

- **Mortality:** The results of the STL function show a sizable seasonal component in the mortality series with a rise and fall roughly occurring every year. There does not seem to be any discernible trend.
- **Temperature:** The results of the STL function show a sizable seasonal component in the temperature series with a rise and fall roughly occurring every year. There does not seem to be any discernible trend but the trend component is much larger than the other components.
- **Chemical 1:** The results of the STL function show a seasonal component in the Chemical 1 series with a rise and fall roughly occurring every year. There seems to be a decreasing trend overall in the series.
- **Chemical 2:** The results of the STL function show a seasonal component in the Chemical 2 series with a rise and fall roughly occurring every year. There seems to be a decreasing trend overall in the series.
- **Particle Size** The results of the STL function show a seasonal component in the particle size series with a rise and fall roughly occurring every year. There seems to be a decreasing trend overall in the series till 2015 and then increases till 2020.

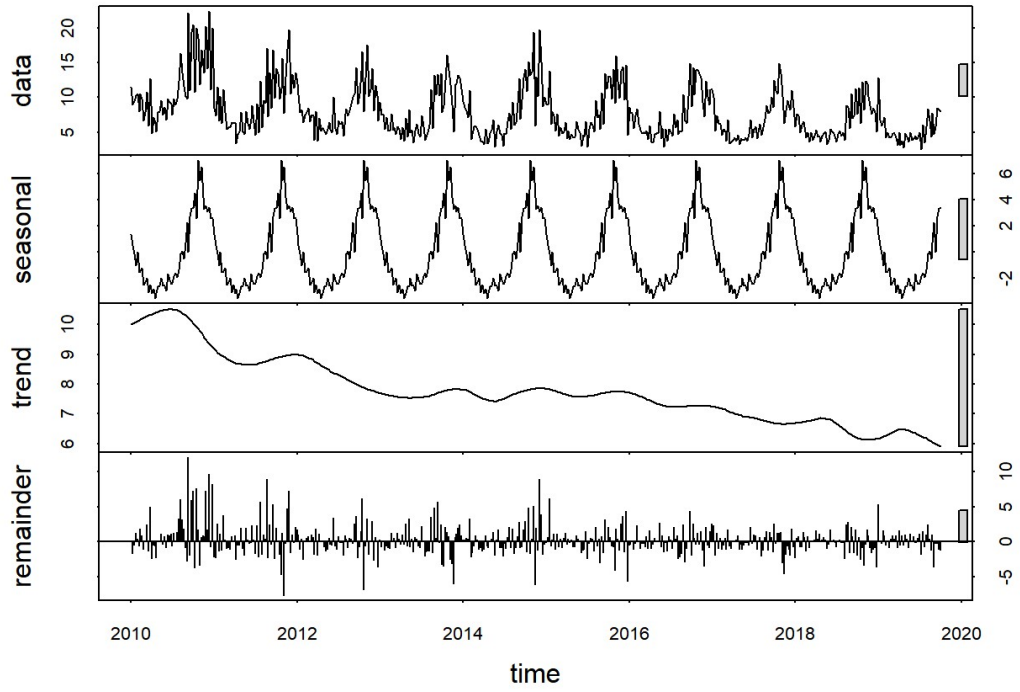
```
# mortality: decomposing

decomp_mortality = stl(mortality, t.window = 52, s.window = "periodic", robust=TRUE)
plot(decomp_mortality)
```



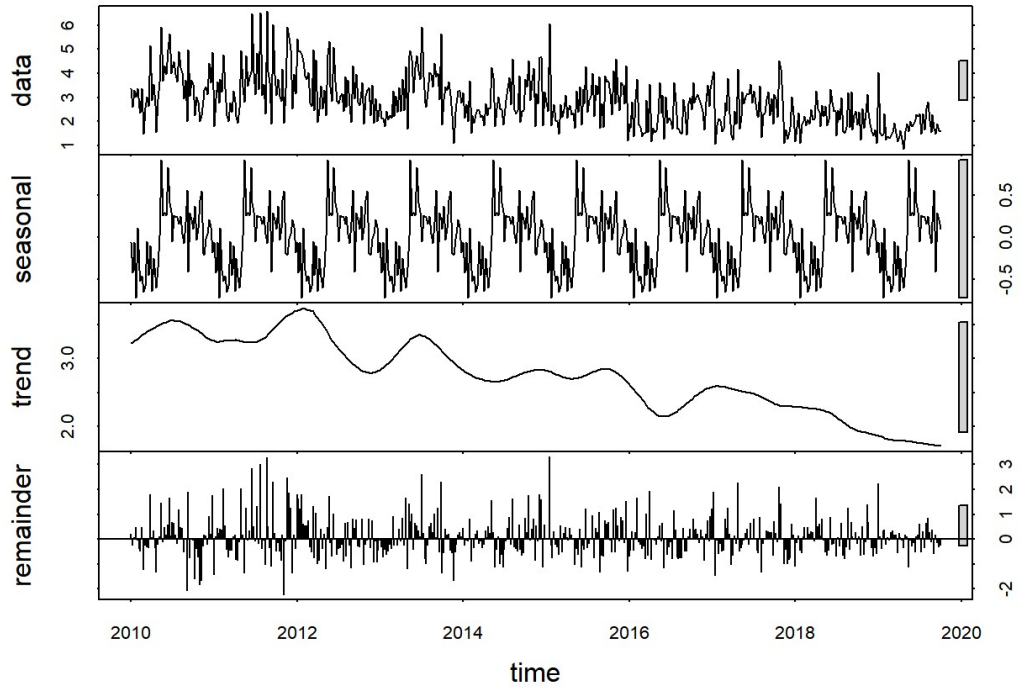

```
# temperature: decomposing

decomp_temp = stl(temp, t.window = 52, s.window = "periodic", robust=TRUE)
plot(decomp_temp)
```



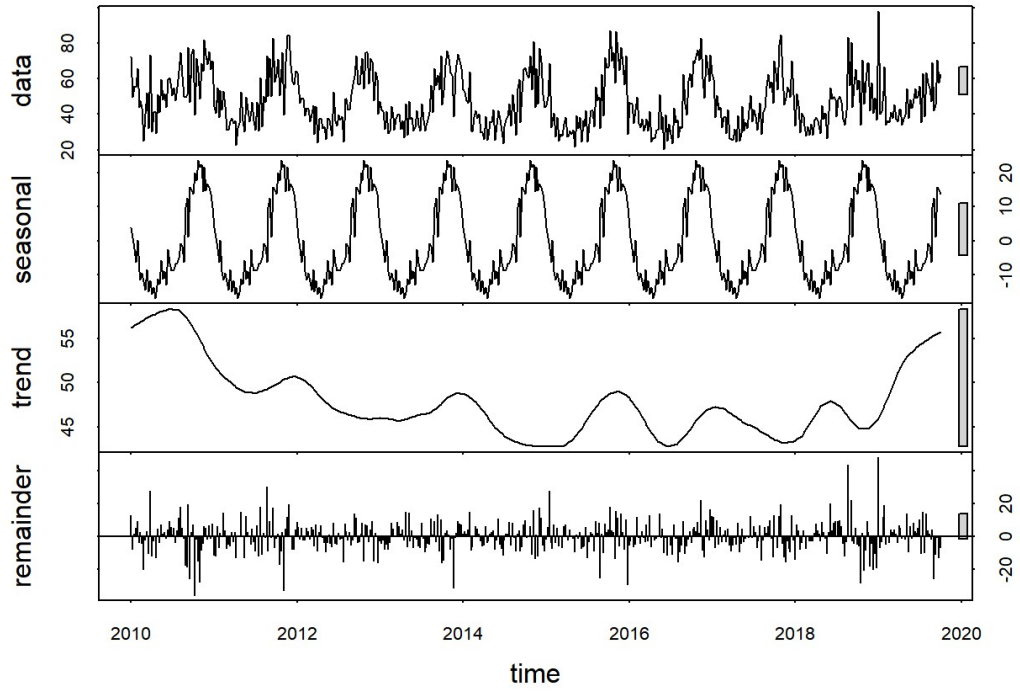

```
# chem1: decomposing

decomp_chem1 = stl(chem1, t.window = 52, s.window = "periodic", robust=TRUE)
plot(decomp_chem1)
```




```
# chem2: decomposing

decomp_chem2 = stl(chem2, t.window = 52, s.window = "periodic", robust=TRUE)
plot(decomp_chem2)
```




```
# particle size: decomposing

decomp_particle.size = stl(particle.size, t.window = 52, s.window = "periodic", robust=TRUE)
plot(decomp_particle.size)
```

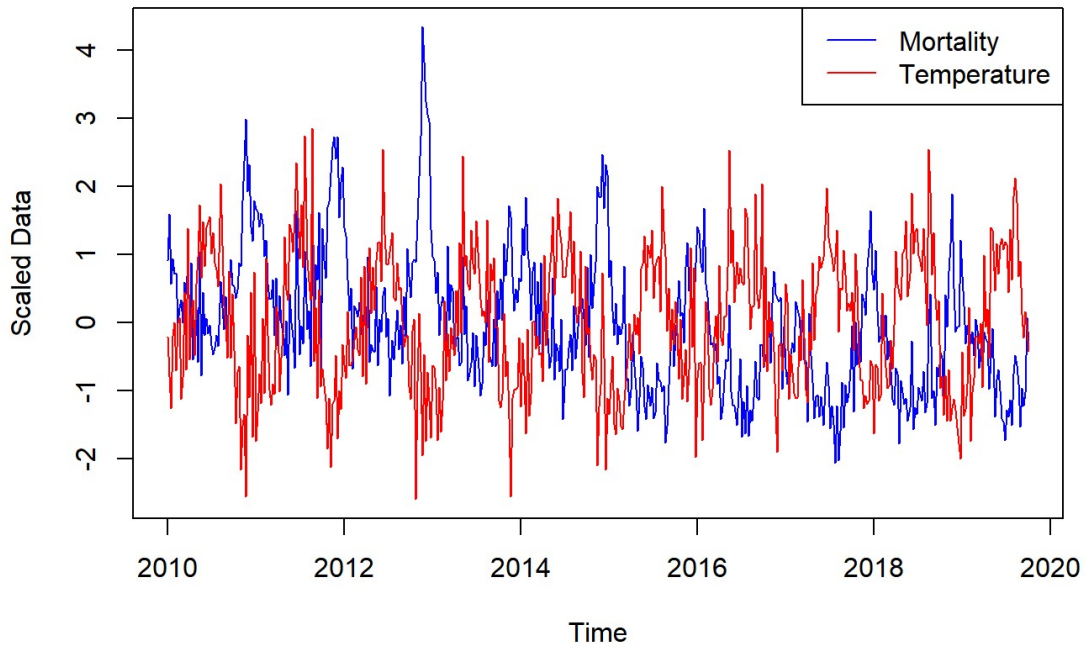


Correlating Predictor Variables with Dependent Variable

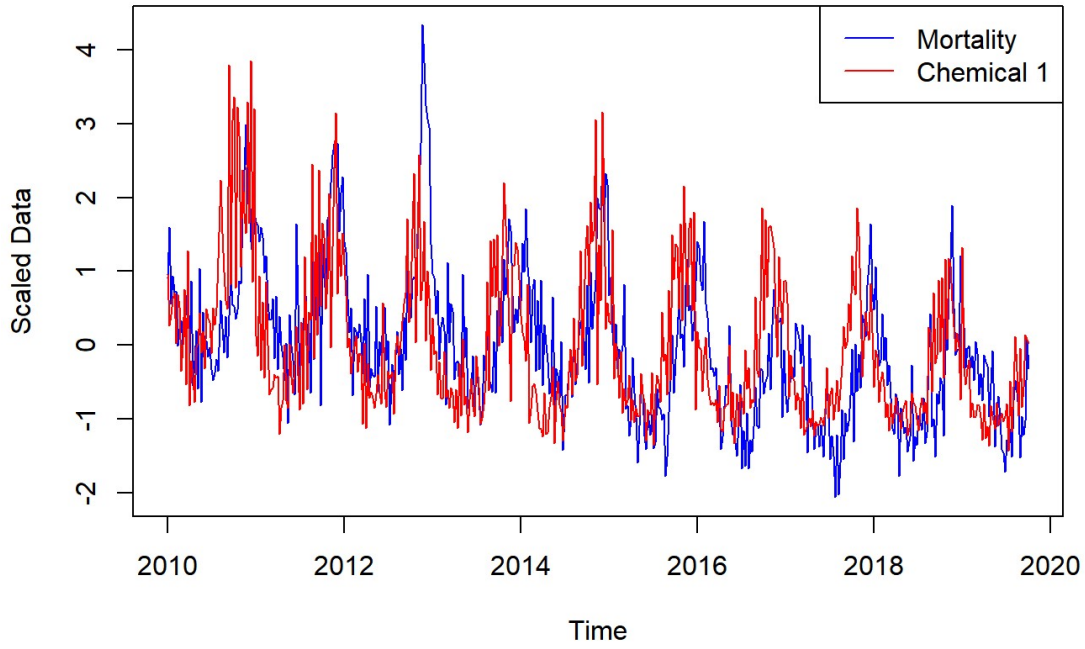
The correlations between the predictor series and the mortality series can be visually determined by plotting the scaled series together on the same plot, so the predictor series are scaled and plotted along with the scaled mortality series.

- **Temperature:** Temperature and mortality seem to have an opposite relationship; whenever temperature rises, mortality rates fall, and vice-versa.
- **Chemical 1:** Chemical 1 and mortality seem to have a relationship where mortality trends seem to closely follow Chemical 1 trends.
- **Chemical 2:** Chemical 2 and mortality seem to have a relationship where mortality trends seem to closely follow Chemical 2 trends.
- **Particle Size** Particle size and mortality seem to have a relationship where mortality trends seem to closely follow particle size trends.

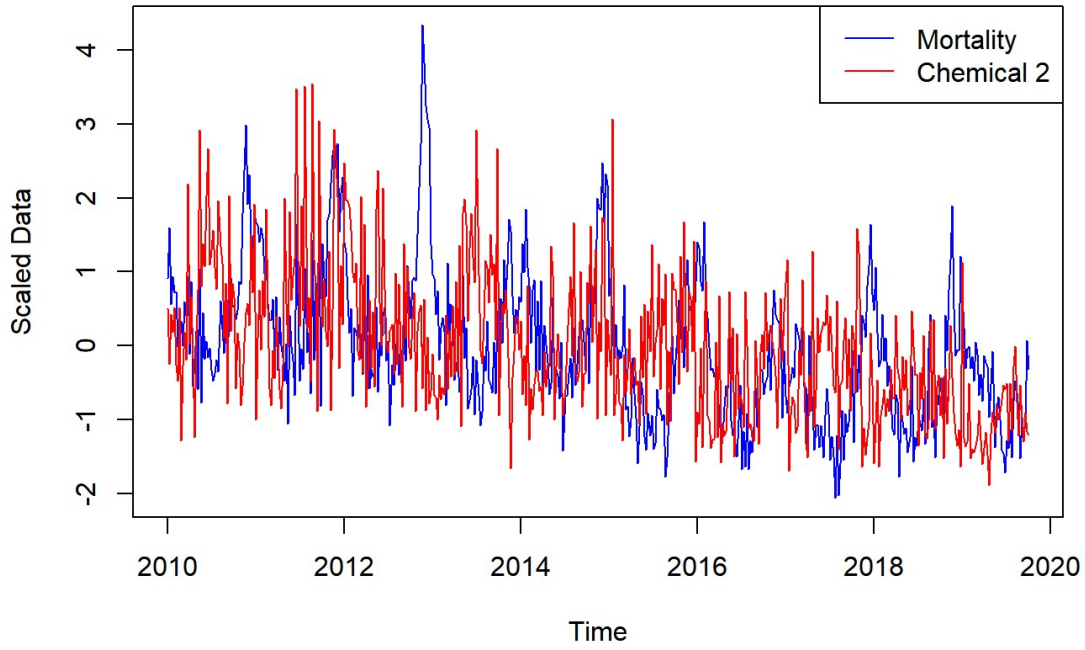
```
mortdata.ts <- ts(mortdata[,c(2,3)], start=c(2010,1), frequency=52)
mortdata_scaled = scale(mortdata.ts)
plot(mortdata_scaled, plot.type="s", col=c("blue", "red"), ylab="Scaled Data")
legend("topright", lty=1, col=c("blue", "red"), c("Mortality", "Temperature"))
```



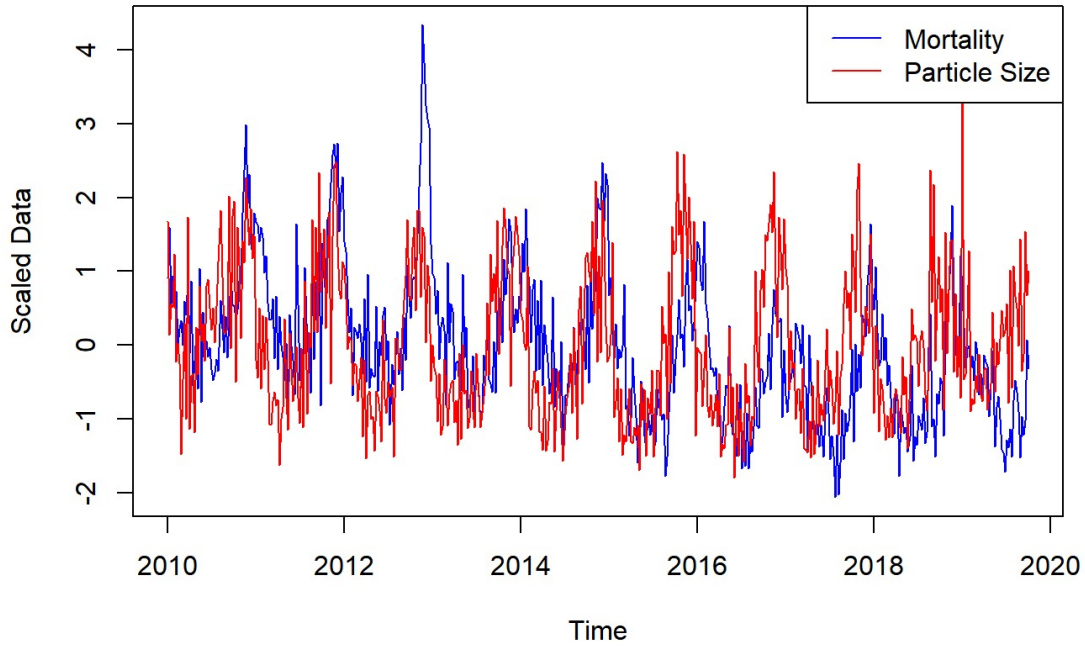

```
mortdata.ts <- ts(mortdata[,c(2,4)], start=c(2010,1), frequency=52)
mortdata_scaled = scale(mortdata.ts)
plot(mortdata_scaled, plot.type="s", col=c("blue", "red"), ylab="Scaled Data")
legend("topright", lty=1, col=c("blue", "red"), c("Mortality", "Chemical 1"))
```




```
mortdata.ts <- ts(mortdata[,c(2,5)], start=c(2010,1), frequency=52)
mortdata_scaled = scale(mortdata.ts)
plot(mortdata_scaled, plot.type="s", col=c("blue", "red"), ylab="Scaled Data")
legend("topright", lty=1, col=c("blue", "red"), c("Mortality", "Chemical 2"))
```




```
mortdata.ts <- ts(mortdata[,c(2,6)], start=c(2010,1), frequency=52)
mortdata_scaled = scale(mortdata.ts)
plot(mortdata_scaled, plot.type="s", col=c("blue", "red"), ylab="Scaled Data")
legend("topright", lty=1, col=c("blue", "red"), c("Mortality", "Particle Size"))
```



Since a visual confirmation of the correlation between predictor and dependent variables is insufficient, the variables can be correlated using the cor() function in order to obtain a numerical representation of correlation. Obtaining the correlation values shows that particle size and chemical 1 are highly positively correlated with a correlation of 0.866, and the most negatively correlated variable pair are temperature and mortality, with a value of -0.439.

```
#correlate variables
mortdata.ts <- ts(mortdata[,2:6], start=c(2010,1), frequency=52)
cor(mortdata.ts)
```

```
##                mortality        temp       chem1     chem2 particle.size
## mortality      1.0000000 -0.43863962  0.55744759 0.2569989    0.44387133
## temp          -0.4386396  1.00000000 -0.09785582 0.4043740   -0.01723095
## chem1          0.5574476 -0.09785582  1.00000000 0.5130047    0.86611747
## chem2          0.2569989  0.40437401  0.51300467 1.0000000    0.46793404
## particle.size  0.4438713 -0.01723095  0.86611747 0.4679340    1.00000000
```

Fitting Distributed Lag Models

Polynomial DLM

A polynomial DLM is first fitted using the polyDlm() function, using each predictor variable to predict mortality rates. Prior to fitting, the ideal lag value is found using the finiteDLMauto() function, which is made to generate multiple models with q ranging from 1 to 10 and uses AIC error to determine the suitability of each model. The results of finiteDLMauto() suggest that the ideal value for q is 10, and a polynomial order of 2.

```
#PolyDLM
finiteDLMauto(x = as.vector(temp), y = as.vector(mortality), q.min = 1, q.max = 10,
              model.type = "poly", error.type = "AIC", trace = TRUE)
```

```
##     q - k    MASE      AIC      BIC   GMRAE     MBRAE R.Adj.Sq Ljung-Box
## 10 10 - 2 1.17648 3497.652 3518.705 1.12767   1.59802  0.35578         0
## 9   9 - 2 1.18299 3506.309 3527.372 1.18128   1.38697  0.35236         0
## 8   8 - 2 1.17701 3514.855 3535.928 1.12205   3.91769  0.34922         0
## 7   7 - 2 1.18820 3524.671 3545.754 1.15834  -0.46471  0.34433         0
## 6   6 - 2 1.19297 3533.400 3554.493 1.14234  -0.48109  0.34079         0
## 4   4 - 2 1.18380 3538.893 3560.006 1.15602   3.59213  0.35068         0
## 5   5 - 2 1.19268 3539.727 3560.830 1.17163   0.49712  0.34110         0
## 3   3 - 2 1.18522 3545.264 3566.387 1.16148   0.52743  0.35136         0
## 2   2 - 2 1.20264 3560.529 3581.662 1.21803   0.85931  0.33981         0
## 1   1 - 2 1.22128 3587.837 3604.751 1.18045 -10.87632  0.31364         0
```

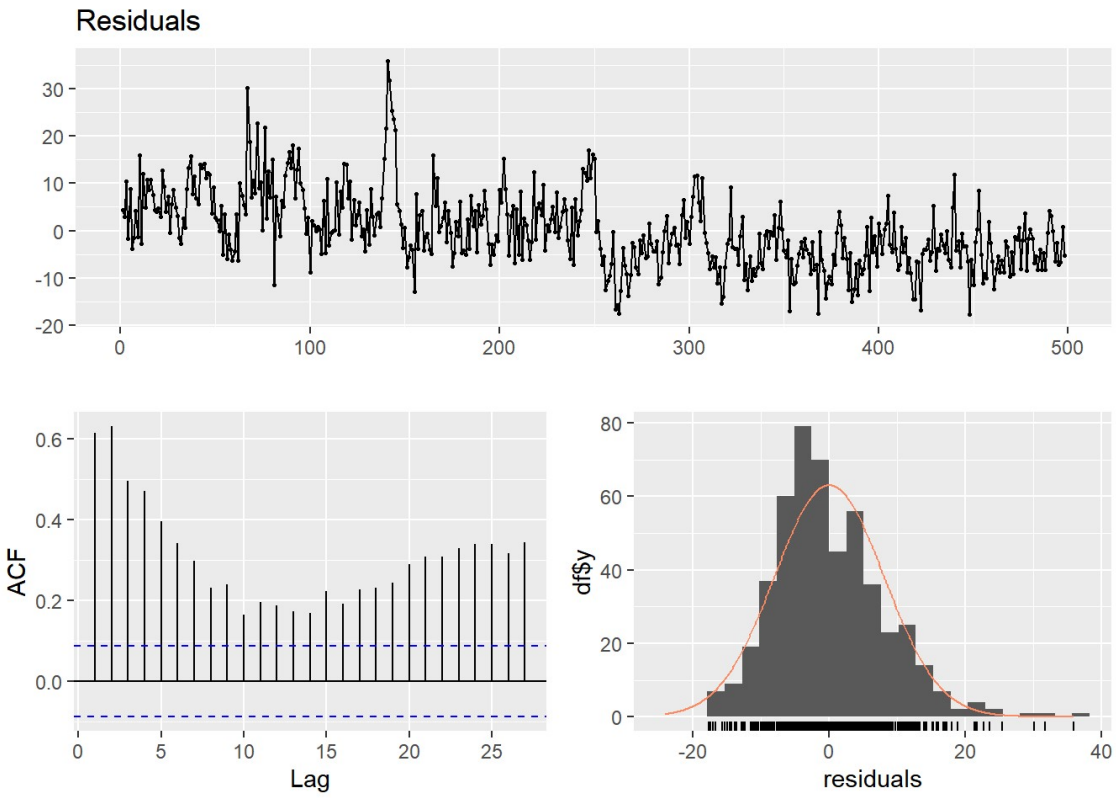
```
model.poly = polyDlm(x = as.vector(temp), y = as.vector(mortality), q = 10, k = 2, show.beta = TRUE)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value  P(>|t|)
## beta.0   -0.1640     0.0374   -4.40 1.35e-05
## beta.1   -0.1680     0.0198   -8.46 3.13e-16
## beta.2   -0.1630     0.0112  -14.60 2.06e-40
## beta.3   -0.1510     0.0144  -10.50 1.68e-23
## beta.4   -0.1310     0.0190   -6.87 1.98e-11
## beta.5   -0.1020     0.0208   -4.94 1.10e-06
## beta.6   -0.0662     0.0189   -3.50 5.09e-04
## beta.7   -0.0220     0.0142   -1.55 1.21e-01
## beta.8    0.0303     0.0112    2.71 7.07e-03
## beta.9    0.0904     0.0202    4.47 9.87e-06
## beta.10   0.1590     0.0380    4.18 3.52e-05
```

```
summary(model.poly)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.704  -5.455  -1.030   4.688  35.845
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 139.822492   4.092647  34.164  < 2e-16 ***
## z.t0         -0.164268   0.037366  -4.396 1.35e-05 ***
## z.t1         -0.007570   0.021630  -0.350   0.7265
## z.t2          0.003986   0.002144   1.859   0.0636 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.059 on 494 degrees of freedom
## Multiple R-squared:  0.3597, Adjusted R-squared:  0.3558
## F-statistic: 92.49 on 3 and 494 DF,  p-value: < 2.2e-16
```

```
checkresiduals(model.poly$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 247.02, df = 10, p-value < 2.2e-16
```

```
MASE(model.poly)
```

```
##                MASE
## model.poly 1.176475
```

```
finiteDLMauto(x = as.vector(chem1), y = as.vector(mortality), q.min = 1, q.max = 10,
              model.type = "poly", error.type = "AIC", trace = TRUE)
```

```
##     q - k    MASE      AIC      BIC   GMRAE    MBRAE R.Adj.Sq Ljung-Box
## 10 10 - 2 0.93396 3293.549 3314.602 0.86449  2.31374  0.57240         0
## 9   9 - 2 0.93557 3300.181 3321.244 0.86102  0.46560  0.57151         0
## 8   8 - 2 0.94430 3310.924 3331.997 0.90656  1.09978  0.56719         0
## 7   7 - 2 0.95280 3327.740 3348.823 0.87296  2.90888  0.55744         0
## 6   6 - 2 0.99619 3363.070 3384.163 0.94182  0.57085  0.53046         0
## 5   5 - 2 1.03078 3401.120 3422.223 1.00588 -0.32075  0.49980         0
## 4   4 - 2 1.04649 3419.705 3440.817 1.03542  0.28767  0.48743         0
## 3   3 - 2 1.10856 3482.544 3503.667 1.03547  5.06770  0.42712         0
## 2   2 - 2 1.12296 3505.901 3527.034 1.07337 -0.44202  0.40737         0
## 1   1 - 2 1.17123 3554.683 3571.597 1.10156  1.64889  0.35708         0
```

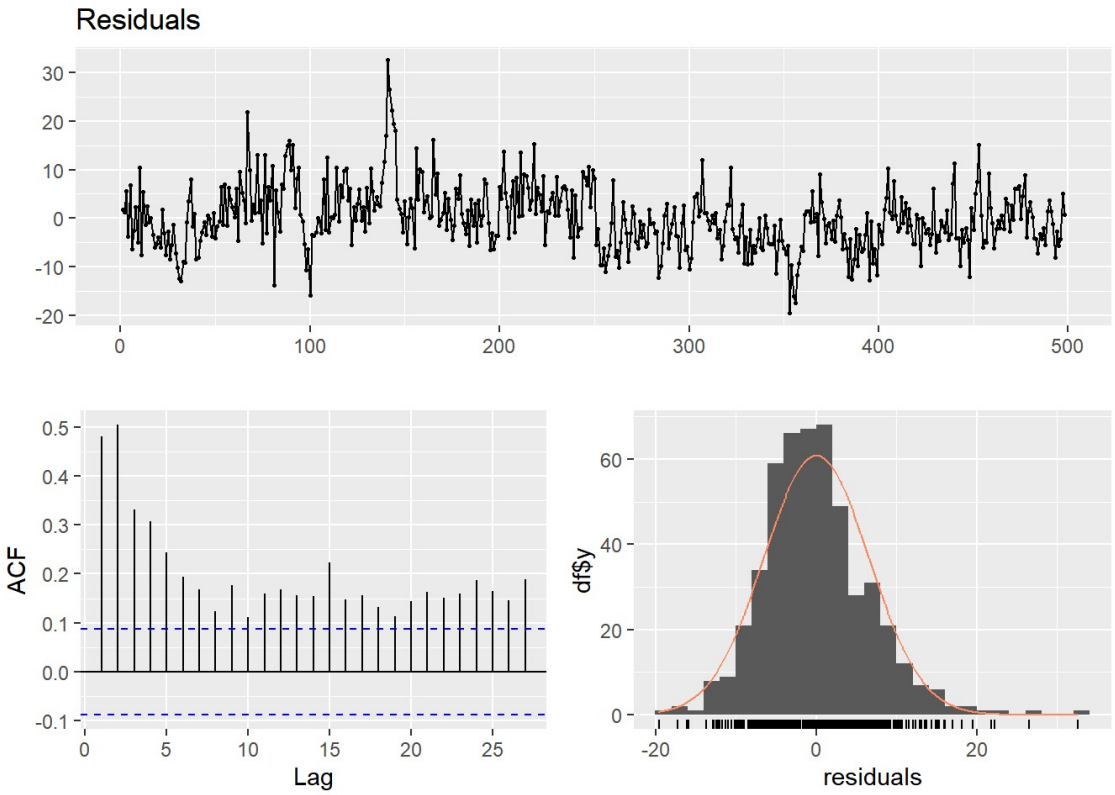
```
model.poly = polyDlm(x = as.vector(chem1), y = as.vector(mortality), q = 10, k = 2, show.beta = TRUE)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value  P(>|t|)
## beta.0     0.200     0.0841    2.38 1.78e-02
## beta.1     0.196     0.0432    4.54 7.17e-06
## beta.2     0.196     0.0229    8.54 1.76e-16
## beta.3     0.199     0.0318    6.24 9.25e-10
## beta.4     0.205     0.0434    4.73 3.00e-06
## beta.5     0.215     0.0477    4.51 8.19e-06
## beta.6     0.228     0.0434    5.25 2.27e-07
## beta.7     0.244     0.0318    7.67 9.10e-14
## beta.8     0.264     0.0229   11.50 2.57e-27
## beta.9     0.287     0.0432    6.64 8.20e-11
## beta.10    0.314     0.0841    3.73 2.14e-04
```

```
summary(model.poly)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -19.519  -4.292  -0.555   3.686  32.589
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 68.444286   0.855229  80.030   <2e-16 ***
## z.t0         0.199889   0.084077   2.377   0.0178 *
## z.t1        -0.005372   0.049603  -0.108   0.9138
## z.t2         0.001675   0.004911   0.341   0.7332
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.566 on 494 degrees of freedom
## Multiple R-squared:  0.575,  Adjusted R-squared:  0.5724
## F-statistic: 222.8 on 3 and 494 DF,  p-value: < 2.2e-16
```

```
checkresiduals(model.poly$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 168.02, df = 10, p-value < 2.2e-16
```

```
MASE(model.poly)
```

```
##                MASE
## model.poly 0.9339622
```

```
finiteDLMauto(x = as.vector(chem2), y = as.vector(mortality), q.min = 1, q.max = 10,
              model.type = "poly", error.type = "AIC", trace = TRUE)
```

```
##     q - k    MASE      AIC      BIC   GMRAE    MBRAE R.Adj.Sq Ljung-Box
## 10 10 - 2 1.35411 3642.451 3663.504 1.31797  0.79372  0.13840         0
## 9   9 - 2 1.35682 3654.305 3675.368 1.30669 -3.22017  0.12875         0
## 8   8 - 2 1.36179 3663.630 3684.703 1.31267  0.75744  0.12369         0
## 7   7 - 2 1.38246 3678.230 3699.313 1.37411  1.02719  0.10917         0
## 6   6 - 2 1.40239 3692.263 3713.356 1.43494 -0.12283  0.09538         0
## 5   5 - 2 1.41014 3707.417 3728.520 1.40647  0.58092  0.08039         0
## 4   4 - 2 1.40671 3712.577 3733.690 1.40871  0.71148  0.08353         0
## 3   3 - 2 1.42626 3729.382 3750.505 1.41997  1.02776  0.06601         0
## 2   2 - 2 1.41012 3733.217 3754.350 1.35394  0.81565  0.07128         0
## 1   1 - 2 1.42040 3745.481 3762.395 1.40304  0.64035  0.06332         0
```

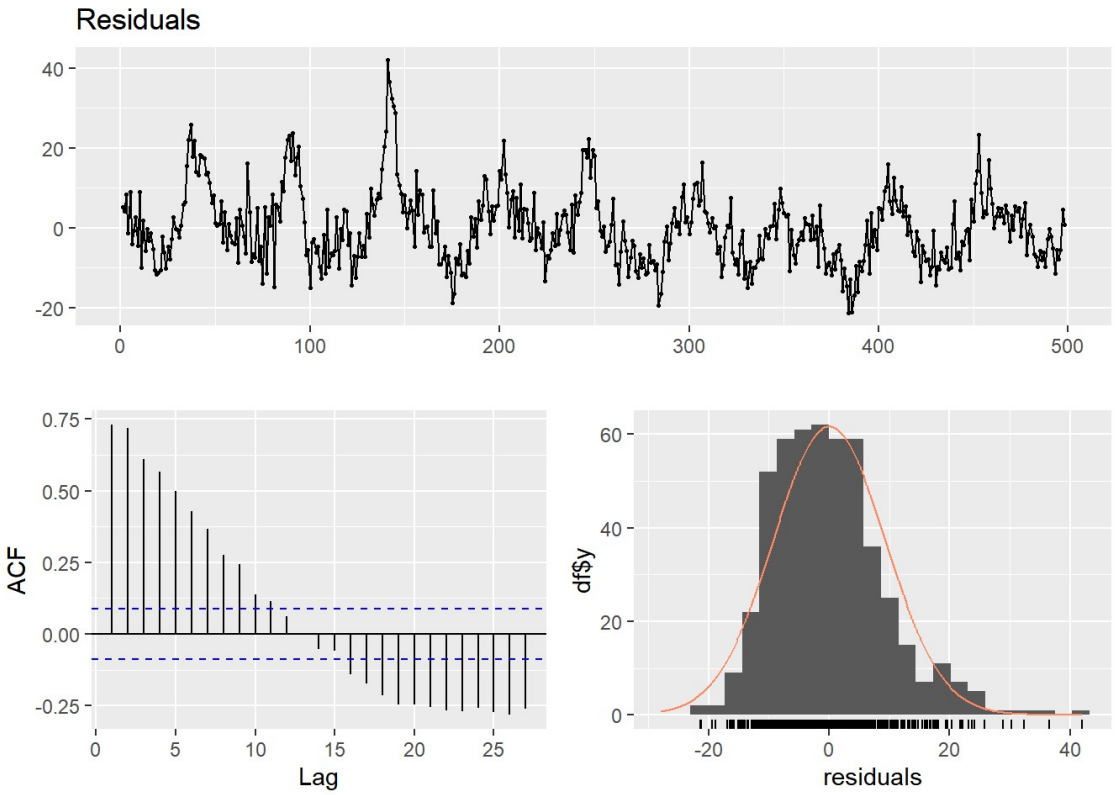
```
model.poly = polyDlm(x = as.vector(chem2), y = as.vector(mortality), q = 10, k = 2, show.beta = TRUE)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value  P(>|t|)
## beta.0    0.1500      0.339   0.443 6.58e-01
## beta.1    0.0852      0.207   0.411 6.82e-01
## beta.2    0.0636      0.140   0.454 6.50e-01
## beta.3    0.0854      0.143   0.599 5.49e-01
## beta.4    0.1500      0.166   0.904 3.67e-01
## beta.5    0.2590      0.177   1.460 1.45e-01
## beta.6    0.4100      0.167   2.460 1.43e-02
## beta.7    0.6050      0.144   4.220 2.96e-05
## beta.8    0.8430      0.142   5.960 4.98e-09
## beta.9    1.1200      0.209   5.390 1.11e-07
## beta.10   1.4500      0.340   4.270 2.40e-05
```

```
summary(model.poly)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -21.395  -6.739  -0.733   5.057  42.045
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 73.63491    1.79049  41.126   <2e-16 ***
## z.t0         0.14996    0.33883   0.443    0.658
## z.t1        -0.08644    0.17546  -0.493    0.622
## z.t2         0.02164    0.01701   1.272    0.204
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.32 on 494 degrees of freedom
## Multiple R-squared:  0.1436, Adjusted R-squared:  0.1384
## F-statistic: 27.61 on 3 and 494 DF,  p-value: < 2.2e-16
```

```
checkresiduals(model.poly$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 309.14, df = 10, p-value < 2.2e-16
```

```
MASE(model.poly)
```

```
##               MASE
## model.poly 1.354105
```

```
finiteDLMauto(x = as.vector(particle.size), y = as.vector(mortality), q.min = 1, q.max = 10,
              model.type = "poly", error.type = "AIC", trace = TRUE)
```

```
##     q - k    MASE      AIC      BIC   GMRAE    MBRAE R.Adj.Sq Ljung-Box
## 10 10 - 2 1.08188 3434.657 3455.710 0.97240  0.27666  0.43233         0
## 9   9 - 2 1.08826 3444.854 3465.917 1.01983  0.59541  0.42740         0
## 8   8 - 2 1.09586 3457.612 3478.685 1.03829  0.27529  0.41962         0
## 7   7 - 2 1.10611 3479.676 3500.759 1.01061 -0.30885  0.40065         0
## 6   6 - 2 1.14482 3513.372 3534.465 1.04586  0.46711  0.36657         0
## 5   5 - 2 1.17572 3542.424 3563.527 1.08057  0.49206  0.33756         0
## 4   4 - 2 1.21159 3562.553 3583.666 1.13720  0.51422  0.31948         0
## 3   3 - 2 1.26686 3608.085 3629.208 1.20115 -0.07147  0.26544         0
## 2   2 - 2 1.26455 3623.850 3644.983 1.17978  0.59759  0.25180         0
## 1   1 - 2 1.30621 3653.402 3670.316 1.27937  5.12726  0.21888         0
```

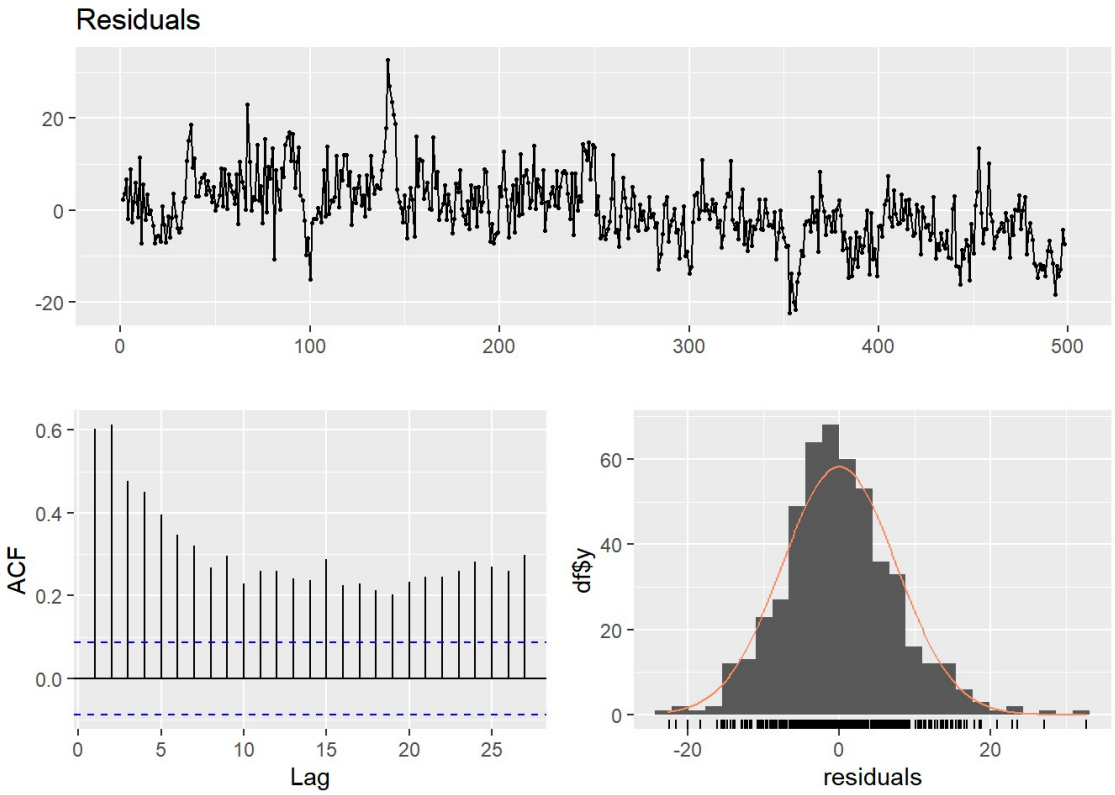
```
model.poly = polyDlm(x = as.vector(particle.size), y = as.vector(mortality), q = 10, k = 2, show.beta = TRUE)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value  P(>|t|)
## beta.0    0.0602    0.02170    2.78 5.73e-03
## beta.1    0.0466    0.01140    4.07 5.51e-05
## beta.2    0.0371    0.00624    5.93 5.59e-09
## beta.3    0.0317    0.00812    3.91 1.06e-04
## beta.4    0.0306    0.01090    2.80 5.33e-03
## beta.5    0.0336    0.01200    2.80 5.28e-03
## beta.6    0.0407    0.01100    3.72 2.24e-04
## beta.7    0.0521    0.00818    6.36 4.54e-10
## beta.8    0.0676    0.00626   10.80 1.78e-24
## beta.9    0.0872    0.01130    7.69 8.19e-14
## beta.10   0.1110    0.02150    5.16 3.66e-07
```

```
summary(model.poly)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.480  -4.595  -0.234   4.395  32.729
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 60.315572   1.522180  39.624  < 2e-16 ***
## z.t0         0.060216   0.021698   2.775  0.00573 **
## z.t1        -0.015749   0.012545  -1.255  0.20993
## z.t2         0.002084   0.001239   1.682  0.09321 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.565 on 494 degrees of freedom
## Multiple R-squared:  0.4358, Adjusted R-squared:  0.4323
## F-statistic: 127.2 on 3 and 494 DF,  p-value: < 2.2e-16
```

```
checkresiduals(model.poly$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 235.13, df = 10, p-value < 2.2e-16
```

```
MASE(model.poly)
```

```
##              MASE
## model.poly 1.08188
```

Model Summary

Model	Model p-value	R-squared	Breusch-Godfrey p-value	MASE
Mortality~Temperature	2.2e-16	0.3558	2.2e-16	1.176
Mortality~Chemical1	2.2e-16	0.5724	2.2e-16	0.934
Mortality~Chemical2	2.2e-16	0.1384	2.2e-16	1.354
Mortality~ParticleSize	2.2e-16	0.4323	2.2e-16	1.082

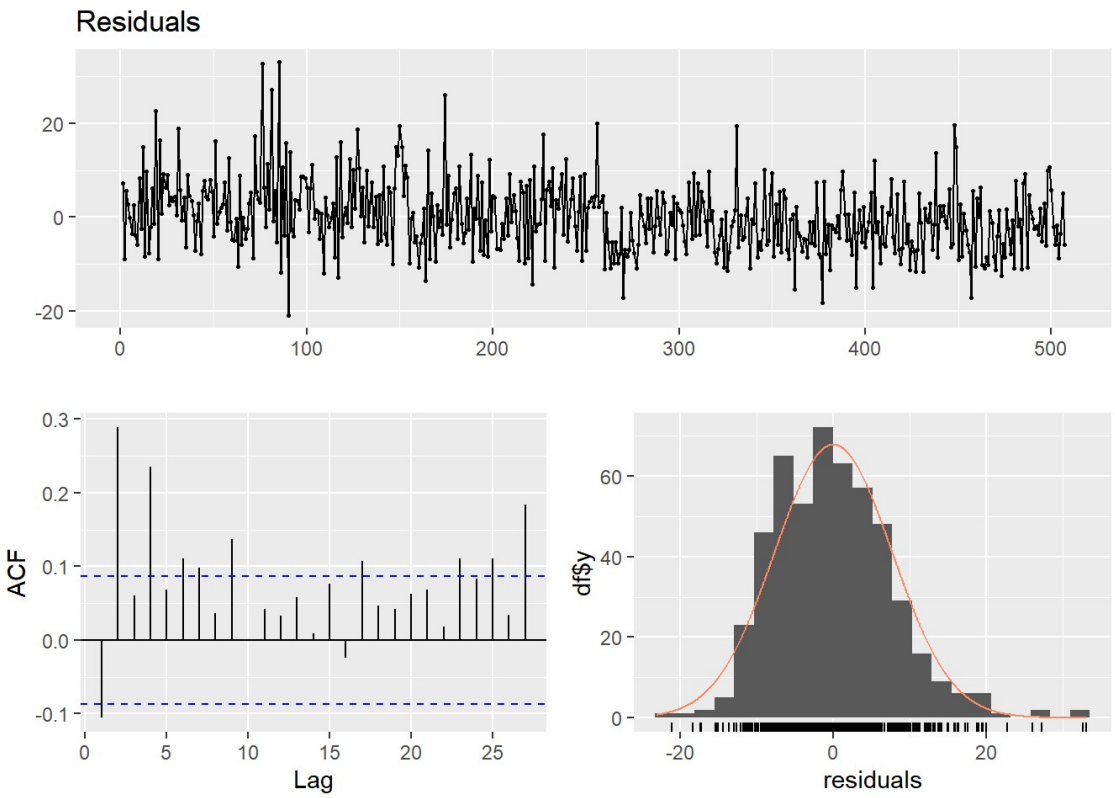
The test statistics for the models demonstrate that chemical 1 is the best predictor for mortality when considering polynomial models due to its relatively low MASE value 0.934 and relatively high R-squared value of 0.5724 which suggests that this model explains the data the best. All the models have a Breusch-Godfrey p-value of 2.2e-16, which implies serial correlaton in the data since the null hypothesis is rejected for this test.

Koyck (Geometric) DLM

```
# Kocyk
model.Koyck = koyckDlm(x = as.vector(temp), y = as.vector(mortality), intercept = TRUE)
summary(model.Koyck)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21.0518  -5.6340  -0.5523   4.7165  33.1339
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 92.12472   11.17552   8.243 1.46e-15 ***
## Y.1          0.47789    0.05572   8.577  < 2e-16 ***
## X.t         -0.61719    0.09227  -6.689 5.98e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.715 on 504 degrees of freedom
## Multiple R-Squared: 0.4072,  Adjusted R-squared: 0.4048
## Wald test: 275.9 on 2 and 504 DF,  p-value: < 2.2e-16
##
## Diagnostic tests:
## NULL
##
##                             alpha       beta       phi
## Geometric coefficients:  176.4462 -0.6171893 0.4778877
```

```
checkresiduals(model.Koyck$model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 102.45, df = 10, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 10
```

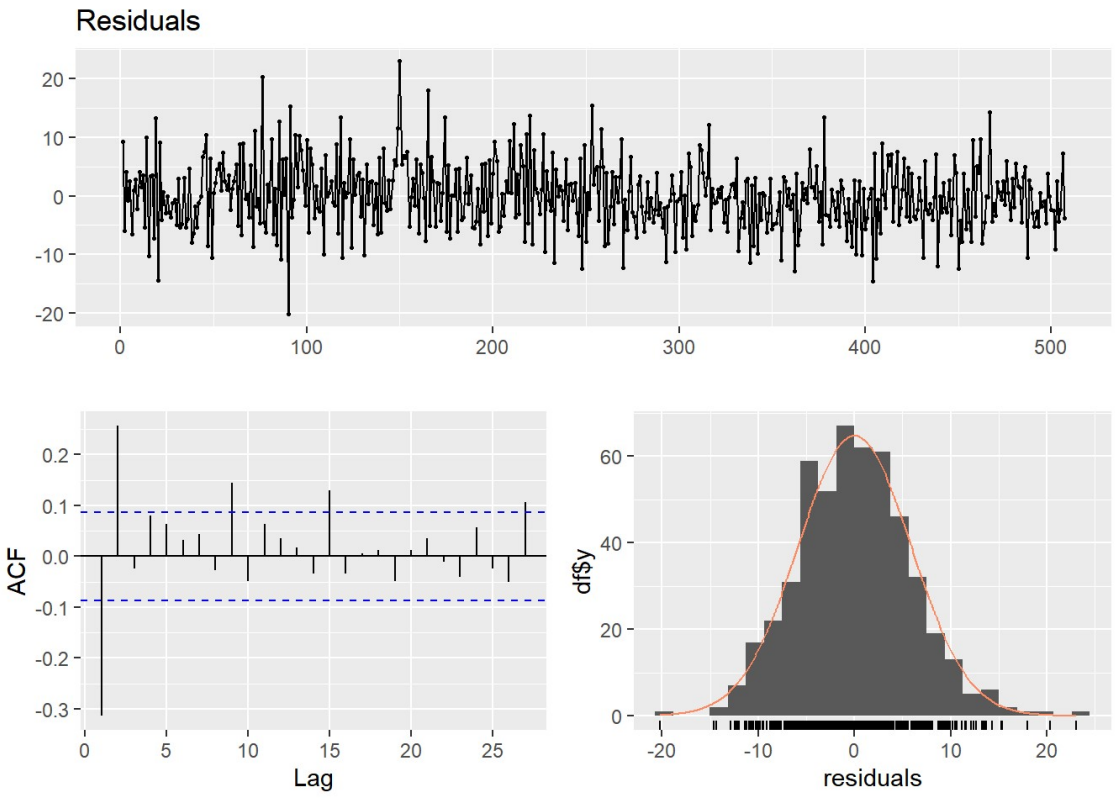
```
MASE(model.Koyck)
```

```
##                MASE
## model.Koyck 1.140721
```

```
model.Koyck = koyckDlm(x = as.vector(chem1), y = as.vector(mortality), intercept = TRUE)
summary(model.Koyck)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -20.19302  -4.18513  -0.06397   3.61514  23.08021
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 24.50952    2.55334   9.599  < 2e-16 ***
## Y.1          0.66674    0.03637  18.331  < 2e-16 ***
## X.t          0.63624    0.15311   4.155 3.81e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.891 on 504 degrees of freedom
## Multiple R-Squared: 0.6544,  Adjusted R-squared: 0.653
## Wald test: 443.5 on 2 and 504 DF,  p-value: < 2.2e-16
##
## Diagnostic tests:
## NULL
##
##                            alpha      beta       phi
## Geometric coefficients:  73.5455 0.6362403 0.6667435
```

```
checkresiduals(model.Koyck$model)
```



```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 103.46, df = 10, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 10
```

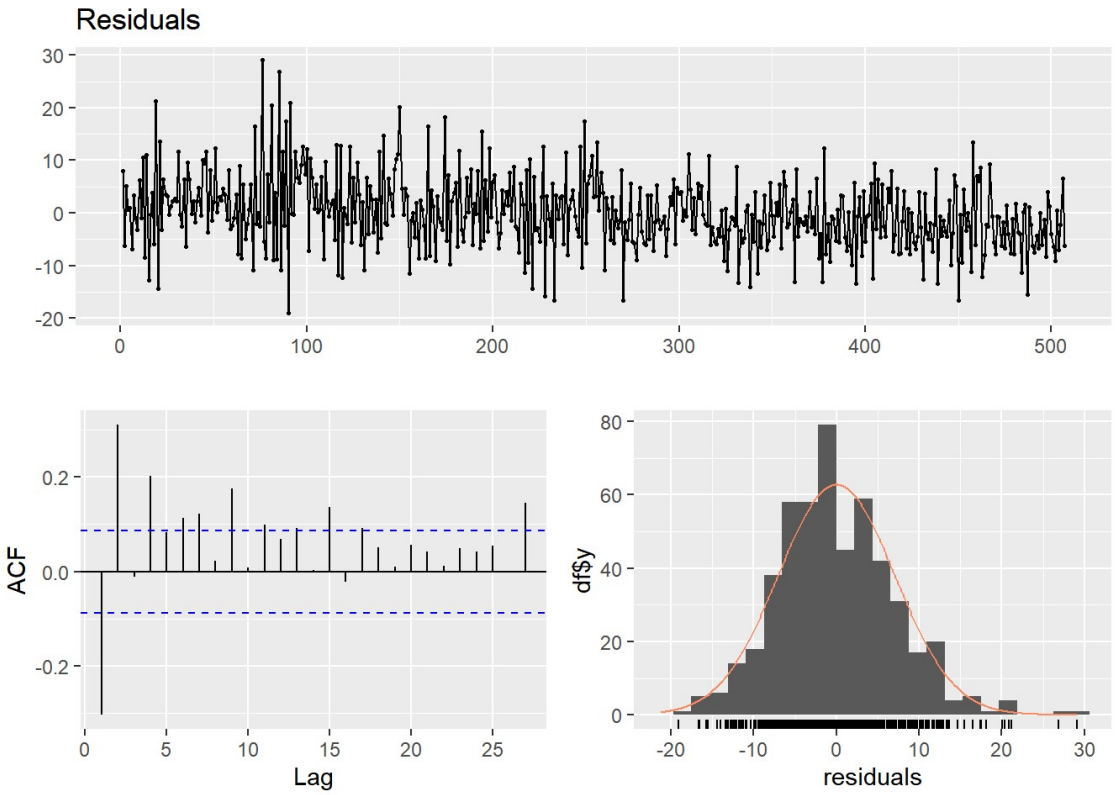
```
MASE(model.Koyck)
```

```
##                  MASE
## model.Koyck 0.8708743
```

```
model.Koyck = koyckDlm(x = as.vector(chem2), y = as.vector(mortality), intercept = TRUE)
summary(model.Koyck)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.0544  -4.8515  -0.5828   4.2734  29.1060
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 22.70597    3.00919   7.546 2.12e-13 ***
## Y.1          0.80752    0.03514  22.979  < 2e-16 ***
## X.t         -1.98938    0.87596  -2.271   0.0236 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.071 on 504 degrees of freedom
## Multiple R-Squared: 0.5021,  Adjusted R-squared: 0.5001
## Wald test: 304.4 on 2 and 504 DF,  p-value: < 2.2e-16
##
## Diagnostic tests:
## NULL
##
##                             alpha      beta       phi
## Geometric coefficients:  117.9669 -1.989383 0.8075225
```

```
checkresiduals(model.Koyck$model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 150.8, df = 10, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 10
```

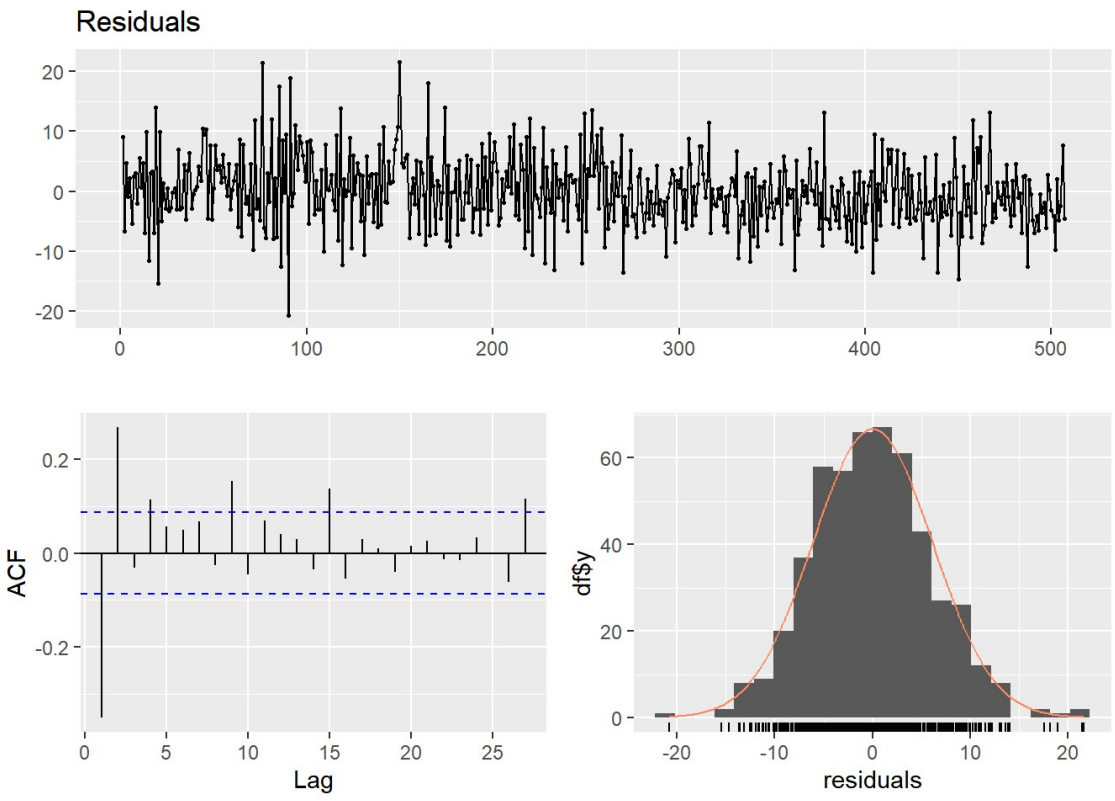
```
MASE(model.Koyck)
```

```
##                 MASE
## model.Koyck 1.037575
```

```
model.Koyck = koyckDlm(x = as.vector(particle.size), y = as.vector(mortality), intercept = TRUE)
summary(model.Koyck)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -20.8011  -4.3973  -0.1927   3.7232  21.6263
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 19.97832    2.44335   8.177 2.39e-15 ***
## Y.1          0.74335    0.03337  22.273  < 2e-16 ***
## X.t          0.05835    0.03945   1.479     0.14
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.147 on 504 degrees of freedom
## Multiple R-Squared: 0.6236,  Adjusted R-squared: 0.6221
## Wald test: 400.4 on 2 and 504 DF,  p-value: < 2.2e-16
##
## Diagnostic tests:
## NULL
##
##                             alpha       beta       phi
## Geometric coefficients:  77.84216 0.05834985 0.7433483
```

```
checkresiduals(model.Koyck$model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 125.71, df = 10, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 10
```

```
MASE(model.Koyck)
```

```
##                  MASE
## model.Koyck 0.9116488
```

Model Summary

Model	Model p-value	R-squared	Breusch-Godfrey p-value	MASE
Mortality~Temperature	2.2e-16	0.4048	2.2e-16	1.141
Mortality~Chemical1	2.2e-16	0.653	2.2e-16	0.871
Mortality~Chemical2	2.2e-16	0.5001	2.2e-16	1.354
Mortality~ParticleSize	2.2e-16	0.4323	2.2e-16	1.038

Once again, the test statistics for the models demonstrate that chemical 1 is the best predictor for mortality when considering polynomial models due to its relatively MASE value 0.871 and relatively high R-squared value of 0.653 which suggests that this model explains the data the best. It also improves upon the polynomial models that were tested prior to this. All the models have a Breusch-Godfrey p-value of 2.2e-16, which implies serial correlaton in the data since the null hypothesis is rejected for this test.

Finite DLM

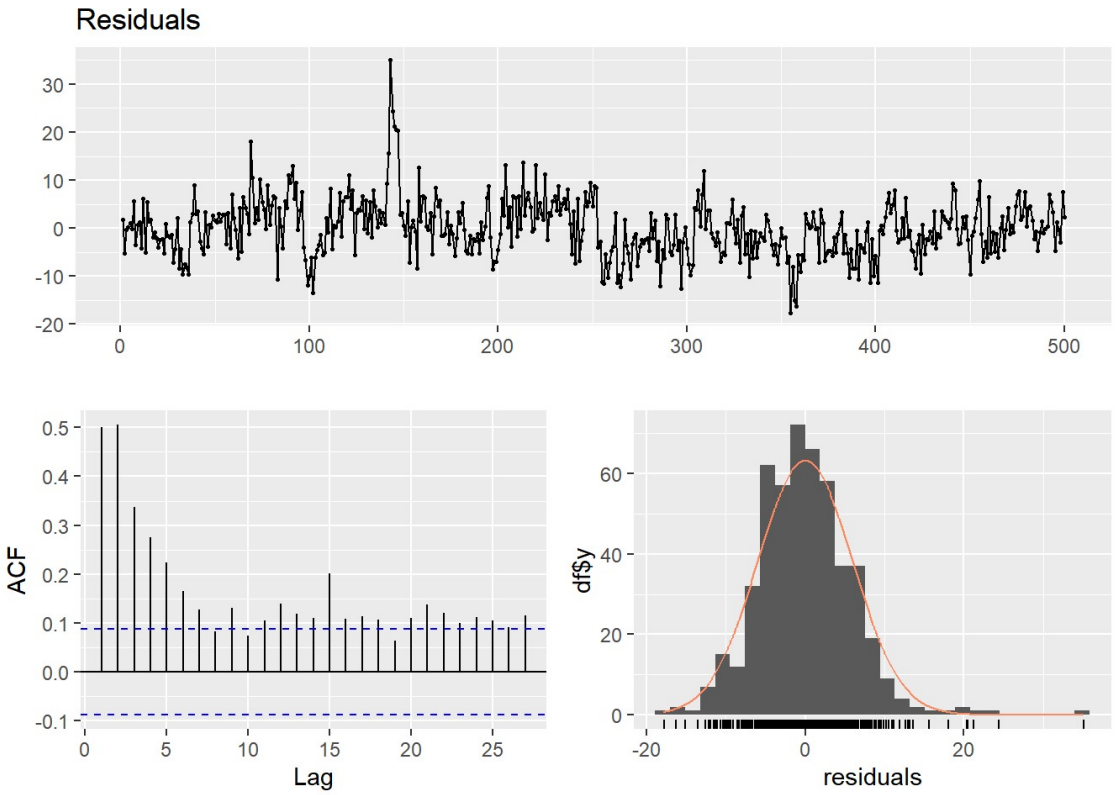
```
# Finite DLM
finiteDLMauto(formula = mortality ~ temp + chem1 + chem2 + particle.size, data = mortdata, q.min = 1, q.max =
10,
              model.type = "dlm", error.type = "AIC", trace = TRUE)
```

```
##    q - k    MASE      AIC      BIC   GMRAE    MBRAE R.Adj.Sq Ljung-Box
## 8      8 0.84703 3278.543 3438.698 0.78651  0.54796  0.61915         0
## 7      7 0.85146 3278.568 3421.932 0.78625  1.01996  0.62053         0
## 10    10 0.84396 3279.046 3472.733 0.79182  1.03923  0.61584         0
## 9      9 0.84432 3279.262 3456.192 0.77719  0.84298  0.61712         0
## 6      6 0.86010 3282.566 3409.124 0.81608  2.78487  0.61882         0
## 5      5 0.86500 3287.322 3397.058 0.83447 -0.48453  0.61691         0
## 4      4 0.87748 3293.368 3386.265 0.86106  0.57910  0.61398         0
## 3      3 0.88541 3305.259 3381.301 0.84886  0.74989  0.60676         0
## 2      2 0.89449 3312.335 3371.506 0.88178 -2.47842  0.60276         0
## 1      1 0.93517 3359.002 3401.287 0.89727  1.90974  0.56803         0
```

```
model.dlm = dlm(formula = mortality ~ temp + chem1 + chem2 + particle.size, data = mortdata , q = 8)
summary(model.dlm)
```

```
##
## Call:
## lm(formula = as.formula(model.formula), data = design)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.709  -4.095  -0.223   3.427  35.099
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     100.588632   5.723392  17.575   <2e-16 ***
## temp.t            0.137964   0.073760   1.870   0.0621 .
## temp.1           -0.172512   0.078314  -2.203   0.0281 *
## temp.2           -0.171627   0.080219  -2.139   0.0329 *
## temp.3           -0.036842   0.081201  -0.454   0.6502
## temp.4            0.018708   0.080274   0.233   0.8158
## temp.5           -0.080036   0.079453  -1.007   0.3143
## temp.6           -0.104285   0.078938  -1.321   0.1871
## temp.7           -0.039912   0.076460  -0.522   0.6019
## temp.8            0.003413   0.073992   0.046   0.9632
## chem1.t          -0.432251   0.250073  -1.729   0.0846 .
## chem1.1           0.422631   0.254806   1.659   0.0979 .
## chem1.2           0.599852   0.269777   2.224   0.0267 *
## chem1.3           0.362176   0.272205   1.331   0.1840
## chem1.4           0.338243   0.275013   1.230   0.2194
## chem1.5          -0.054342   0.276443  -0.197   0.8442
## chem1.6           0.396061   0.275595   1.437   0.1514
## chem1.7           0.132888   0.261566   0.508   0.6117
## chem1.8          -0.184293   0.250221  -0.737   0.4618
## chem2.t           0.646338   0.527653   1.225   0.2212
## chem2.1           0.127011   0.533612   0.238   0.8120
## chem2.2           0.600082   0.544201   1.103   0.2707
## chem2.3          -0.194781   0.542878  -0.359   0.7199
## chem2.4           0.283078   0.549317   0.515   0.6066
## chem2.5           0.171385   0.543533   0.315   0.7527
## chem2.6           0.628067   0.534139   1.176   0.2403
## chem2.7           0.111900   0.523815   0.214   0.8309
## chem2.8           0.239583   0.516086   0.464   0.6427
## particle.size.t   0.132990   0.053459   2.488   0.0132 *
## particle.size.1  -0.049296   0.053702  -0.918   0.3591
## particle.size.2  -0.054961   0.055342  -0.993   0.3212
## particle.size.3  -0.067376   0.055848  -1.206   0.2283
## particle.size.4  -0.041350   0.057858  -0.715   0.4752
## particle.size.5   0.053328   0.055985   0.953   0.3413
## particle.size.6  -0.048686   0.055723  -0.874   0.3827
## particle.size.7   0.048278   0.054100   0.892   0.3727
## particle.size.8   0.051096   0.053714   0.951   0.3420
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.184 on 463 degrees of freedom
## Multiple R-squared:  0.6466, Adjusted R-squared:  0.6192
## F-statistic: 23.53 on 36 and 463 DF,  p-value: < 2.2e-16
##
## AIC and BIC values for the model:
##        AIC      BIC
## 1 3278.543 3438.698
```

```
checkresiduals(model.dlm$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 40
##
## data:  Residuals
## LM test = 204.63, df = 40, p-value < 2.2e-16
```

```
MASE(model.dlm)
```

```
##                MASE
## model.dlm 0.8470314
```

A finite DLM is fitted using the dlm() function. Prior to fitting, the ideal lag value is found using the finiteDLMauto() function, which is made to generate multiple models with q ranging from 1 to 10 and uses AIC error to determine the suitability of each model. The results of finiteDLMauto() suggest that the ideal value for q is 8.

The summary of the finite DLM shows a low p-value of 2.2e-16, which suggests that the model is not significant. Additionally, a low R-squared value of 0.6192 means that the model explains the data decently well. The first and second lags of temperature, the second lag of chemical 2, and the particle size at t are significant.

The Breusch-Godfrey test for serial correlation produces a low p-value of 2.2e-16, which means that there is evidence of serial correlation in the model, making it less suitable to model mortality rates.

Analysing the residuals for the finite DLM shows that the residuals are normally distributed, suggesting that the model may account for the data well and the errors in the model are random. The ACF plot also shows peaks above the 95% confidence interval, with the the first few peaks/lags being the largest.

The MASE of the DLM model is seen to be 0.847, a marginal improvement over the polynomial

model.

ARDL Model

```
#ARDL
columns = c("p","q","AIC","BIC")
df = data.frame(matrix(nrow = 0, ncol = length(columns)))
colnames(df) = columns

for(i in 1:10){
  for(j in 1:10){
    model.ARDL = ardlDlm(formula = mortality ~ temp + chem1 + chem2 + particle.size, data = mortdata, p = i,
q = j)
    new_row = data.frame(p = i, q=j, AIC=AIC(model.ARDL$model), BIC=BIC(model.ARDL$model))
    df = bind_rows(df, new_row)
  }
}

df[order(df$AIC),]
```

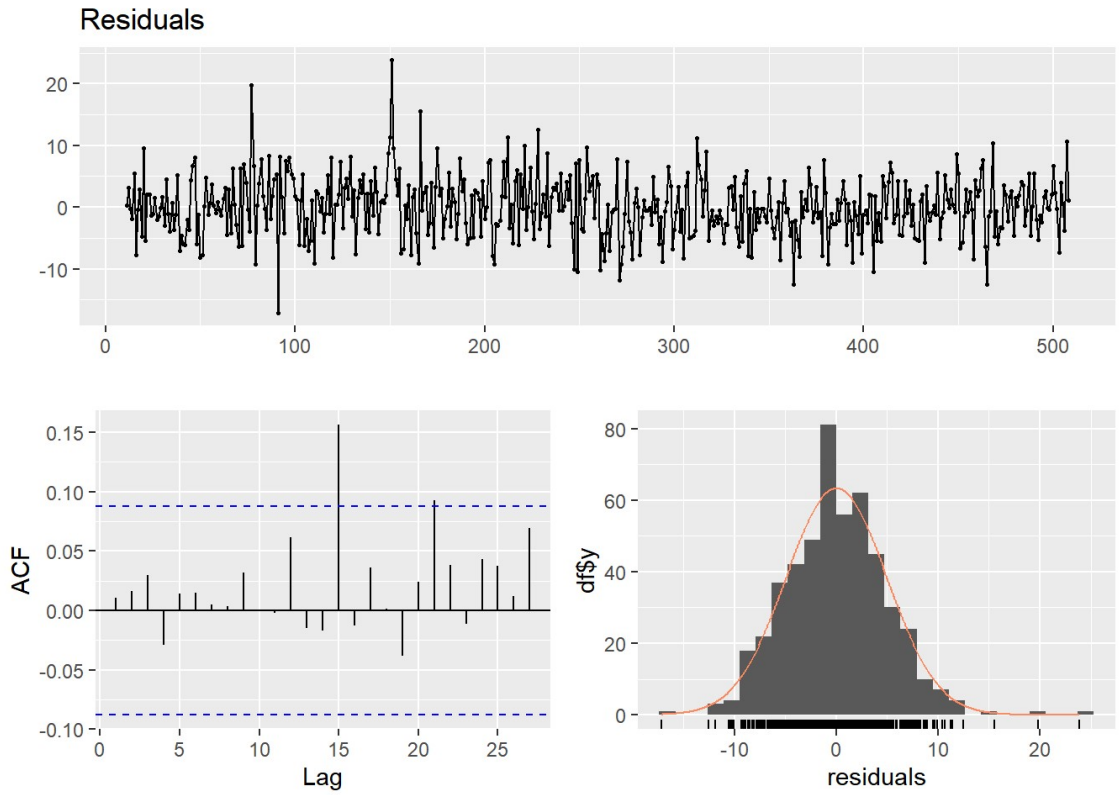
```
##      p  q      AIC      BIC
## 20   2 10 3055.242 3156.296
## 50   5 10 3057.152 3208.734
## 10   1 10 3057.256 3141.468
## 40   4 10 3057.598 3192.337
## 30   3 10 3058.559 3176.456
## 19   2  9 3059.228 3156.118
## 9    1  9 3061.211 3141.251
## 39   4  9 3061.352 3191.943
## 60   6 10 3061.379 3229.803
## 49   5  9 3061.518 3208.960
## 29   3  9 3062.461 3176.201
## 70   7 10 3064.361 3249.627
## 18   2  8 3065.650 3158.371
## 59   6  9 3065.707 3229.999
## 38   4  8 3068.546 3194.984
## 48   5  8 3068.626 3211.923
## 69   7  9 3068.641 3249.783
## 8    1  8 3068.688 3144.551
## 28   3  8 3069.252 3178.832
## 17   2  7 3069.314 3157.863
## 80   8 10 3069.594 3271.702
## 72   8  2 3070.752 3239.337
## 62   7  2 3071.561 3223.359
## 37   4  7 3072.045 3194.327
## 47   5  7 3072.237 3211.385
## 7    1  7 3072.291 3143.974
## 92  10  2 3072.301 3274.409
## 82   9  2 3072.350 3257.704
## 73   8  3 3072.742 3245.541
## 27   3  7 3072.765 3178.180
## 58   6  8 3072.933 3233.088
## 63   7  3 3073.558 3229.573
## 52   6  2 3073.621 3208.616
## 16   2  6 3073.672 3158.044
## 79   8  9 3073.807 3271.799
## 93  10  3 3074.297 3280.617
## 83   9  3 3074.334 3263.901
## 74   8  4 3074.567 3251.580
## 42   5  2 3074.771 3192.947
## 64   7  4 3075.261 3235.492
## 90   9 10 3075.277 3294.229
## 53   6  3 3075.599 3214.813
## 84   9  4 3076.131 3269.911
## 36   4  6 3076.153 3194.274
## 94  10  4 3076.198 3286.728
## 75   8  5 3076.316 3257.544
## 46   5  6 3076.338 3211.333
## 68   7  8 3076.478 3253.491
## 57   6  7 3076.600 3232.614
## 43   5  3 3076.747 3199.144
## 54   6  4 3077.098 3220.530
## 26   3  6 3077.111 3178.357
## 65   7  5 3077.125 3241.573
## 6    1  6 3077.134 3144.632
## 15   2  5 3077.312 3157.503
## 85   9  5 3077.700 3275.692
## 95  10  5 3077.837 3292.578
## 76   8  6 3078.263 3263.706
## 44   5  4 3078.490 3205.108
## 66   7  6 3078.835 3247.499
## 55   6  5 3078.987 3226.638
## 89   9  9 3079.395 3294.238
## 99  10  9 3079.576 3311.159
## 86   9  6 3079.647 3281.852
```

```
## 35   4  5 3079.819 3193.775
## 96  10  6 3079.829 3298.780
## 32   4  2 3079.898 3181.239
## 77   8  7 3080.016 3269.674
## 67   7  7 3080.335 3253.215
## 45   5  5 3080.370 3211.208
## 56   6  6 3080.540 3232.410
## 5    1  5 3080.633 3143.942
## 14   2  4 3080.641 3156.648
## 25   3  5 3080.703 3177.776
## 78   8  8 3081.279 3275.150
## 87   9  7 3081.412 3287.830
## 100 10 10 3081.481 3317.274
## 97  10  7 3081.617 3304.778
## 33   4  3 3081.832 3187.396
## 88   9  8 3082.591 3293.221
## 98  10  8 3083.230 3310.603
## 34   4  4 3083.718 3193.505
## 13   2  3 3083.766 3155.583
## 4    1  4 3084.051 3143.167
## 24   3  4 3084.268 3177.164
## 22   3  2 3085.541 3170.032
## 12   2  2 3087.354 3154.978
## 23   3  3 3087.515 3176.231
## 3    1  3 3087.542 3142.462
## 2    1  2 3091.552 3142.270
## 61   7  1 3130.889 3278.470
## 71   8  1 3132.061 3296.431
## 91  10  1 3133.741 3331.639
## 81   9  1 3134.118 3315.260
## 51   6  1 3134.439 3265.216
## 41   5  1 3137.607 3251.563
## 31   4  1 3142.215 3239.334
## 11   2  1 3152.521 3215.919
## 21   3  1 3152.577 3232.844
## 1    1  1 3171.658 3218.171
```

```
model.ARDL <- ardlDlm(formula = mortality ~ temp + chem1 + chem2 + particle.size, data = mortdata, p = 2, q = 10)
summary(model.ARDL)
```

```
##
## Time series regression with "ts" data:
## Start = 11, End = 508
##
## Call:
## dynlm(formula = as.formula(model.text), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.1995  -3.2874  -0.2863   3.0884  23.8930
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     35.221541   7.148102   4.927 1.15e-06 ***
## temp.t           0.176789   0.057549   3.072 0.002249 **
## temp.1          -0.198223   0.059715  -3.319 0.000971 ***
## temp.2          -0.119512   0.058062  -2.058 0.040103 *
## chem1.t         -0.347093   0.191930  -1.808 0.071170 .
## chem1.1          0.677347   0.185986   3.642 0.000300 ***
## chem1.2          0.490333   0.199399   2.459 0.014286 *
## chem2.t          0.258399   0.405238   0.638 0.524010
## chem2.1         -0.603089   0.395017  -1.527 0.127491
## chem2.2          0.613144   0.402235   1.524 0.128088
## particle.size.t  0.119890   0.041199   2.910 0.003783 **
## particle.size.1 -0.074322   0.038727  -1.919 0.055570 .
## particle.size.2 -0.076907   0.041081  -1.872 0.061810 .
## mortality.1      0.326897   0.046020   7.103 4.48e-12 ***
## mortality.2      0.334966   0.046882   7.145 3.41e-12 ***
## mortality.3     -0.022881   0.047665  -0.480 0.631428
## mortality.4      0.009469   0.047002   0.201 0.840428
## mortality.5      0.025593   0.046972   0.545 0.586107
## mortality.6     -0.009524   0.046759  -0.204 0.838696
## mortality.7     -0.021893   0.046720  -0.469 0.639569
## mortality.8     -0.039870   0.046895  -0.850 0.395639
## mortality.9      0.068229   0.044430   1.536 0.125288
## mortality.10    -0.015421   0.041806  -0.369 0.712383
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.073 on 475 degrees of freedom
## Multiple R-squared:  0.756,  Adjusted R-squared:  0.7447
## F-statistic: 66.88 on 22 and 475 DF,  p-value: < 2.2e-16
```

```
checkresiduals(model.ARDL$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 26
##
## data:  Residuals
## LM test = 38.666, df = 26, p-value = 0.05245
```

```
MASE(model.ARDL)
```

```
##                MASE
## model.ARDL 0.7240674
```

An ARDL model is constructed using the ardlDlm() function. Before fitting the model, the optimal lag value for the independent series (p) and the optimal lag value for the dependent series (q) are determined through a process of multiple iterations. In each iteration, a new model is created with different p and q values. The AIC and BIC errors are computed for each model to assess their suitability. Based on the outcomes of this procedure, it is recommended that the optimal value for p is 2, while the optimal value for q is 10.

The summary of the ARDL model shows a very low p-value of 2.2e-16, which suggests that the model is significant. A relatively high R-squared value of 0.7447 makes it better than the Koyck and polynomial models and reveals that the model explains most of the data better.

The following lags are significant:

- Temperature at t, its first lag, and its second lag
- Chemical 1 at its first and second lags.
- Particle size at t
- Mortality at its first and second lags.

The Breusch-Godfrey test produces a high p-value of 0.05245, which fails to reject the null hypothesis of independently distributed data. Therefore, it is possible to conclude that there is no

serial correlation in the model.

Analysing the residuals for the ARDL model shows that the residuals are relatively small compared to the Koyck and polynomial models and are roughly normally distributed, suggesting that the model accounts for the data well and the errors in the model are random. The ACF plot shows that most of the peaks are within the 95% confidence interval, with 2 peaks lying outside the interval. Finally, the MASE of 0.724 is observed to be an improvement on the previous models tested so far.

Exponential Smoothing Models

A simple exponential smoothing model, a Holt's linear trend model, and a Holt's linear trend model with additive damped trend are implemented using the mortality series. Models that take into account seasonal components are not considered due to the weekly nature of the time series data.

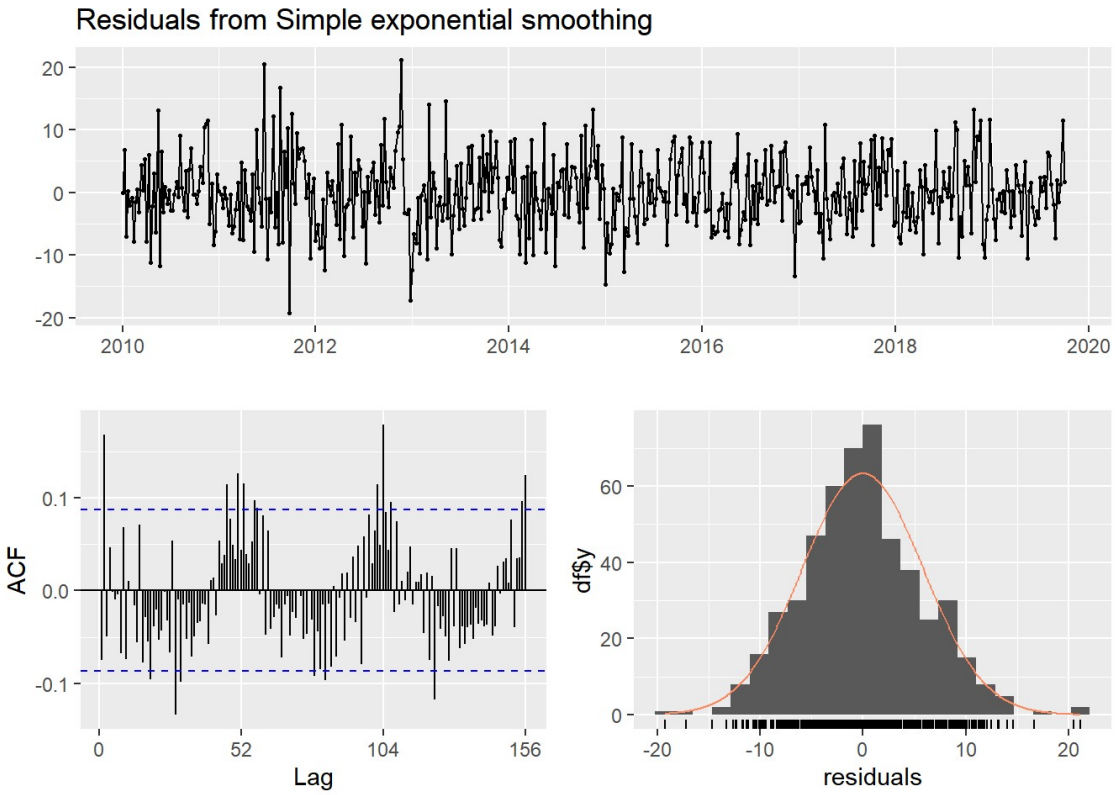
The three above models are created for the mortality series using the ses() and holt() functions and a zoomed view of the forecast is plotted to obtain a better view regarding the trend.

The results of the point forecasts show that the Holt's linear trend model predicts slightly higher mortality rates compared to the other two models, and the forecasts for SES and the additive damped trend model are quite similar. This causes the forecast plot for the SES model to be covered by the additive damped trend model.

```
fit1.ses <- ses(mortality, initial="simple", h=4)
summary(fit1.ses)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = mortality, h = 4, initial = "simple")
##
##   Smoothing parameters:
##     alpha = 0.5111
##
##   Initial states:
##     l = 97.85
##
##   sigma:  5.8818
## Error measures:
##                       ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.05078709 5.881821 4.586997 -0.3726948 5.154724 0.6869107
##                     ACF1
## Training set -0.07502973
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95     Hi 95
## 2019.769       84.66395 77.12609 92.20180 73.13579  96.19211
## 2019.788       84.66395 76.19865 93.12924 71.71739  97.61050
## 2019.808       84.66395 75.36324 93.96466 70.43974  98.88816
## 2019.827       84.66395 74.59691 94.73098 69.26775 100.06015
```

```
checkresiduals(fit1.ses)
```

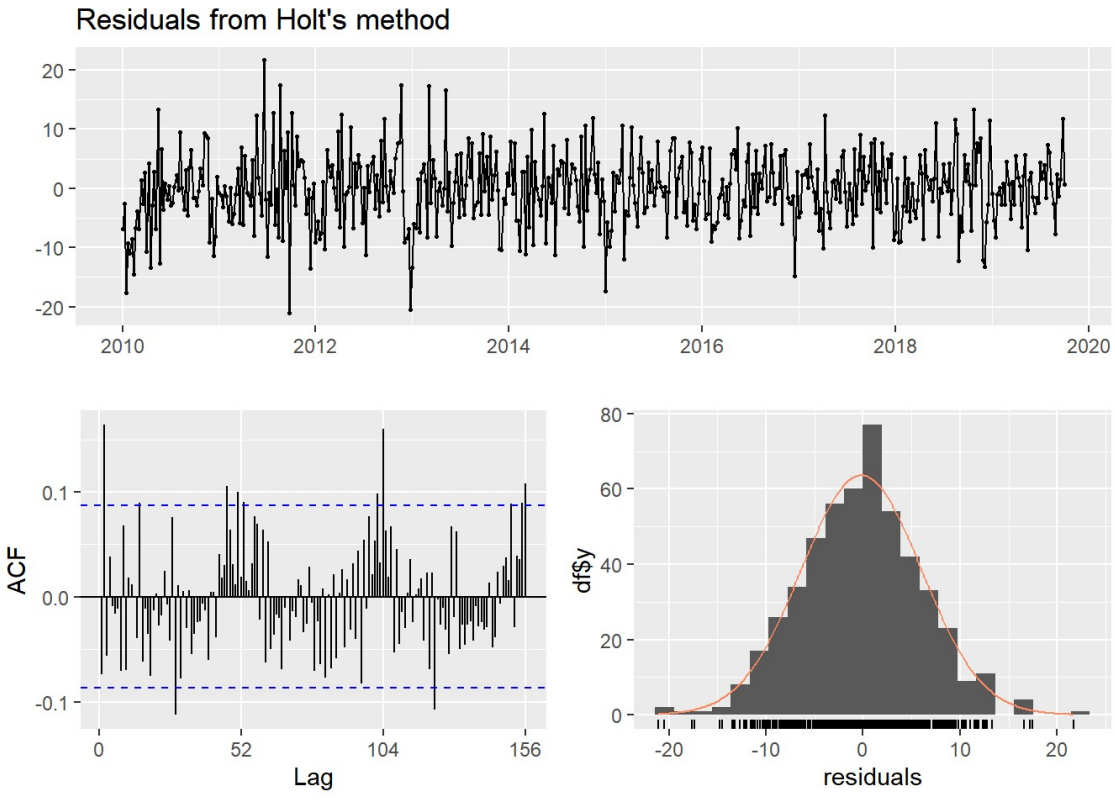



```
##
##  Ljung-Box test
##
## data:  Residuals from Simple exponential smoothing
## Q* = 190.84, df = 102, p-value = 2.351e-07
##
## Model df: 0.   Total lags used: 102
```

```
fit2.holt <- holt(mortality, initial="simple", h=4)
summary(fit2.holt)
```

```
##
## Forecast method: Holt's method
##
## Model Information:
## Holt's method
##
## Call:
##  holt(y = mortality, h = 4, initial = "simple")
##
##   Smoothing parameters:
##     alpha = 0.5525
##     beta  = 0.1119
##
##   Initial states:
##     l = 97.85
##     b = 6.79
##
##   sigma:  6.1983
## Error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.194873 6.198283 4.849227 -0.4238146 5.444693 0.7261802
##                      ACF1
## Training set -0.07427104
##
## Forecasts:
##          Point Forecast    Lo 80     Hi 80    Lo 95     Hi 95
## 2019.769       85.85713 77.91371  93.80055 73.70872  98.00554
## 2019.788       86.52651 76.98978  96.06325 71.94134 101.11169
## 2019.808       87.19589 75.83923  98.55255 69.82738 104.56441
## 2019.827       87.86527 74.49547 101.23508 67.41792 108.31263
```

```
checkresiduals(fit2.holt)
```

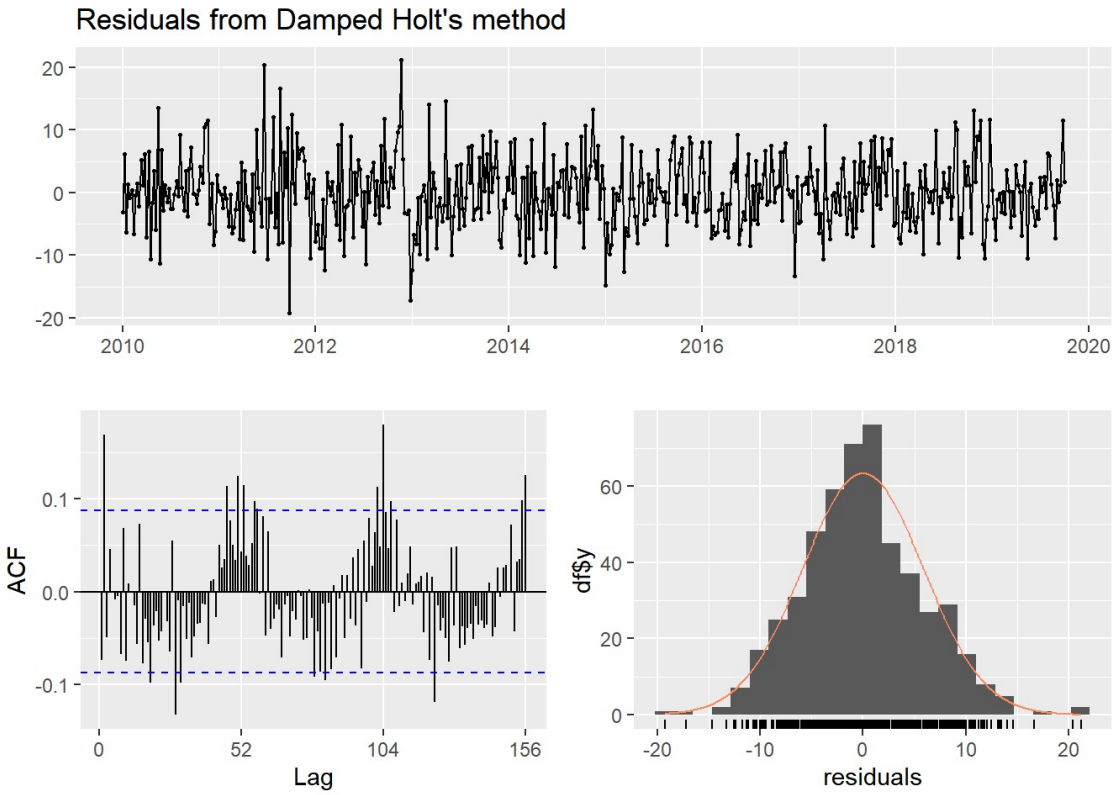


```
##
##  Ljung-Box test
##
## data:  Residuals from Holt's method
## Q* = 141.64, df = 102, p-value = 0.005774
##
## Model df: 0.   Total lags used: 102
```

```
fit3.holt <- holt(mortality, damped=TRUE, initial="simple", h=4) # Fit with additive damped trend
summary(fit3.holt)
```

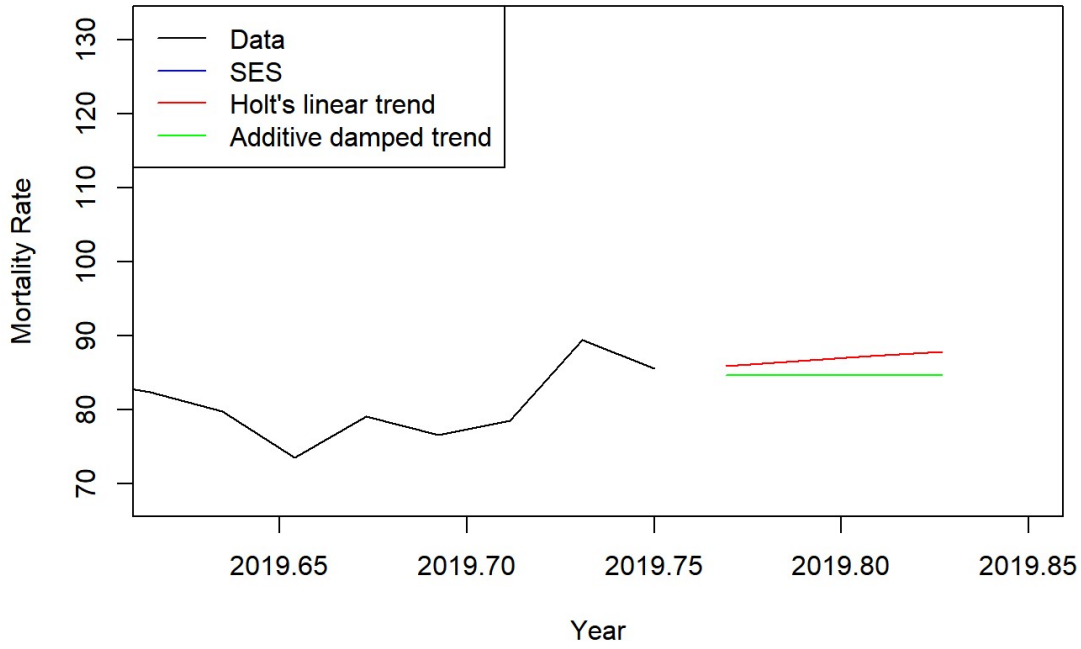
```
##
## Forecast method: Damped Holt's method
##
## Model Information:
## Damped Holt's method
##
## Call:
##  holt(y = mortality, h = 4, damped = TRUE, initial = "simple")
##
##   Smoothing parameters:
##     alpha = 0.5082
##     beta  = 1e-04
##     phi   = 0.9119
##
##   Initial states:
##     l = 102.0415
##     b = -1.1632
##
##   sigma:  5.9071
##
##      AIC     AICc      BIC
## 4976.632 4976.800 5002.015
##
## Error measures:
##                       ME     RMSE      MAE        MPE     MAPE     MASE
## Training set -0.02068605 5.877951 4.588308 -0.3396637 5.154426 0.687107
##                    ACF1
## Training set -0.0738815
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95     Hi 95
## 2019.769       84.64364 77.07339 92.21388 73.06595  96.22133
## 2019.788       84.64444 76.15242 93.13646 71.65702  97.63187
## 2019.808       84.64517 75.32181 93.96854 70.38632  98.90403
## 2019.827       84.64584 74.55921 94.73248 69.21967 100.07202
```

```
checkresiduals(fit3.holt)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt's method
## Q* = 189.16, df = 102, p-value = 3.536e-07
##
## Model df: 0.   Total lags used: 102
```

```
plot(mortality, type="l", ylab="Mortality Rate", xlab="Year", fcol="white", plot.conf=FALSE, , xlim=c(2019.6
2,2019.85))
lines(fit1.ses$mean, col="blue", type="l")
lines(fit2.holt$mean, col="red", type="l")
lines(fit3.holt$mean, col="green", type="l")
legend("topleft", lty=1, col=c("black","blue","red","green"),
       c("Data","SES", "Holt's linear trend", "Additive damped trend"))
```



Model Summary

Model	AIC	BIC	MASE	Ljung-Box p-value	Residual Normality
Simple SES	NA	NA	0.6869107	2.351e-07	Normal
Simple Holt	NA	NA	0.7261802	0.005774	Normal
Additive Damped Trend Holt SES	4976.632	5002.015	0.687107	3.536e-07	Normal

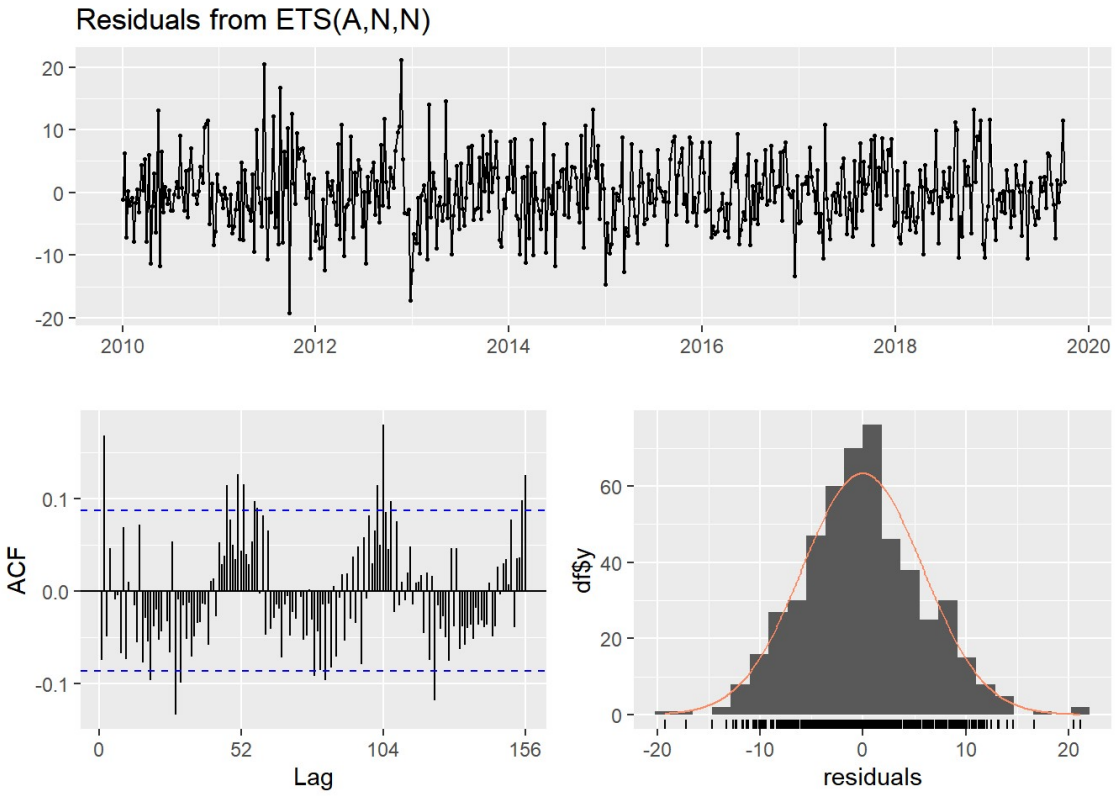
Checking the summaries and residuals for each model reveals that all the models have normal residuals. The Simple SES and Additive Damped Trend Holt SES models have similar MASE values that are lower compared to the Simple Holt model. The Simple SES model can be chosen as the best one due to its slightly lower MASE. All models have low p-values from the Ljung-Box test, rejecting the null hypothesis and showing evidence of serial correlation in the model.

State Space Models

```
etsAN = ets(mortality, model="ANN")
summary(etsAN)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = mortality, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.511
##
##   Initial states:
##     l = 98.9364
##
##   sigma:  5.8932
##
##      AIC     AICc      BIC
## 4971.256 4971.303 4983.947
##
## Training set error measures:
##                       ME     RMSE      MAE         MPE     MAPE      MASE
## Training set -0.05497816 5.88156 4.588578 -0.3769459 5.156433 0.6871475
##                      ACF1
## Training set -0.07517189
```

```
checkresiduals(etsAN)
```

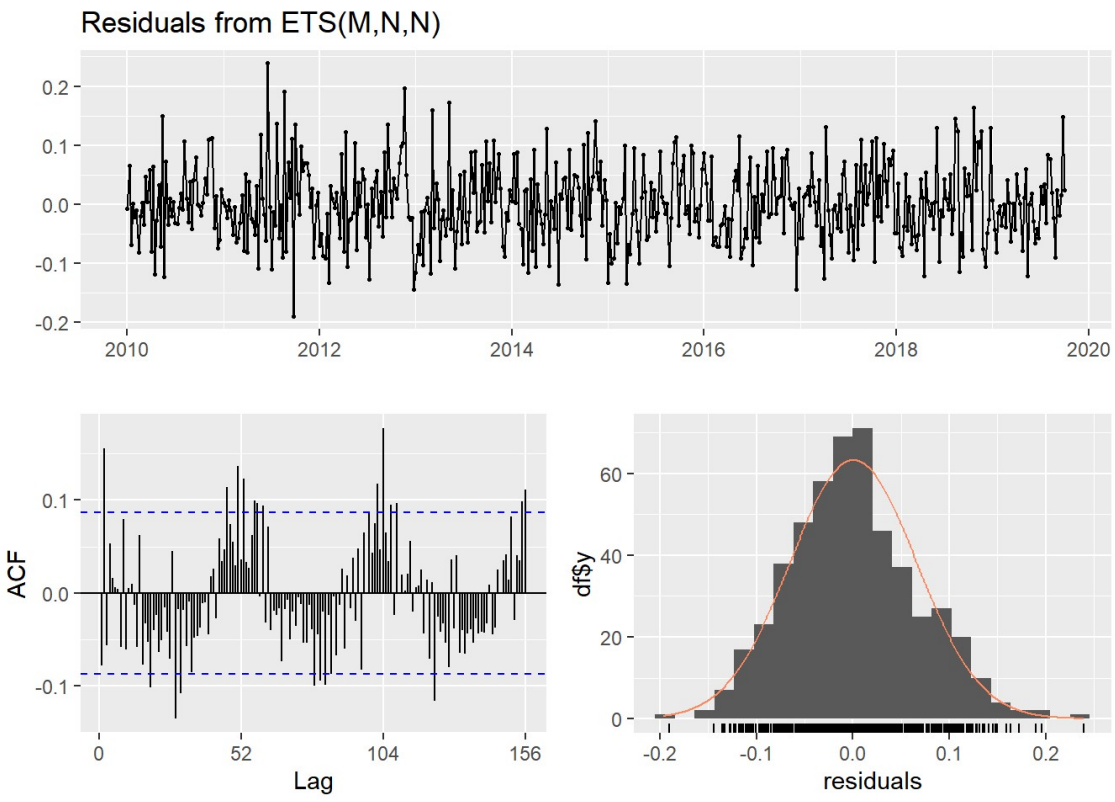



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,N)
## Q* = 191.57, df = 102, p-value = 1.967e-07
##
## Model df: 0.   Total lags used: 102
```

```
etsMN = ets(mortality, model="MNN")
summary(etsMN)
```

```
## ETS(M,N,N)
##
## Call:
##  ets(y = mortality, model = "MNN")
##
##   Smoothing parameters:
##     alpha = 0.4843
##
##   Initial states:
##     l = 98.5582
##
##   sigma:  0.0656
##
##      AIC     AICc      BIC
## 4954.111 4954.159 4966.803
##
## Training set error measures:
##                       ME     RMSE      MAE         MPE     MAPE      MASE
## Training set -0.05730399 5.88508 4.593891 -0.3849281 5.159608 0.6879431
##                      ACF1
## Training set -0.04372931
```

```
checkresiduals(etsMN)
```

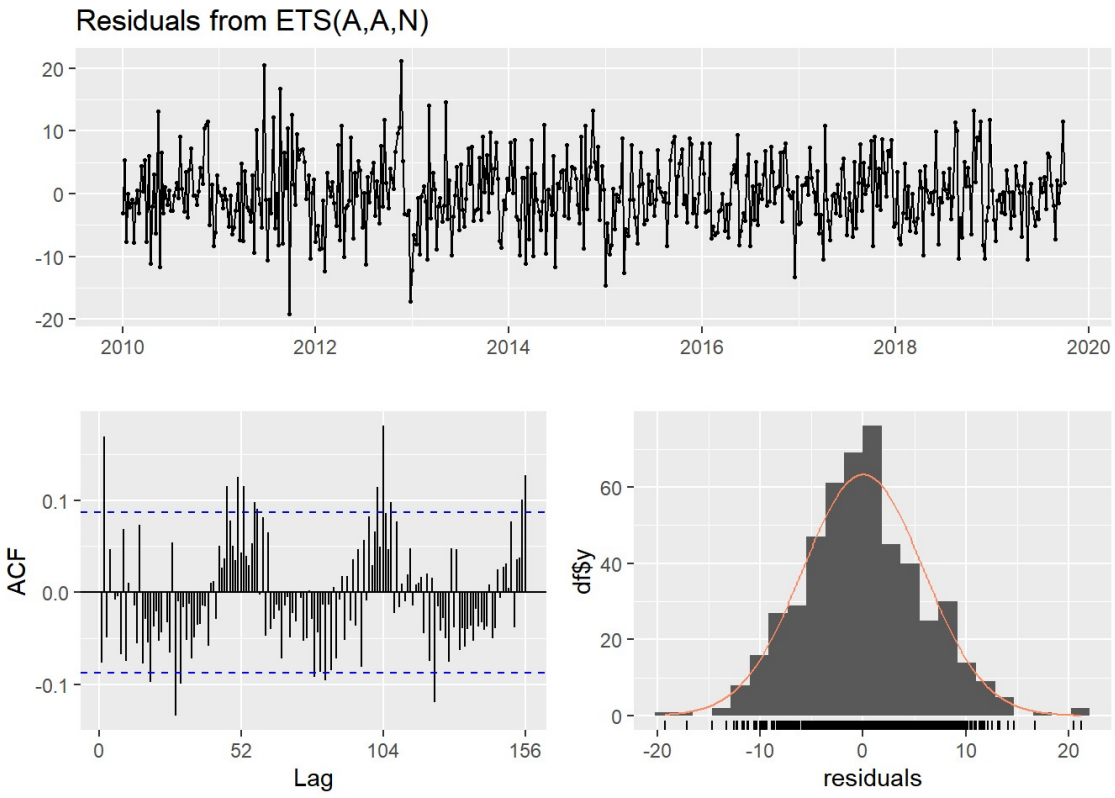



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,N)
## Q* = 207.78, df = 102, p-value = 3.201e-09
##
## Model df: 0.   Total lags used: 102
```

```
etsAA = ets(mortality, model="AAN")
summary(etsAA)
```

```
## ETS(A,A,N)
##
## Call:
##  ets(y = mortality, model = "AAN")
##
##   Smoothing parameters:
##     alpha = 0.5122
##     beta  = 1e-04
##
##   Initial states:
##     l = 100.9765
##     b = -0.029
##
##   sigma:  5.906
##
##      AIC     AICc      BIC
## 4975.460 4975.579 4996.612
##
## Training set error measures:
##                       ME    RMSE      MAE        MPE     MAPE      MASE
## Training set -0.004457548 5.88274 4.590352 -0.3178407 5.155916 0.6874132
##                     ACF1
## Training set -0.07651869
```

```
checkresiduals(etsAA)
```

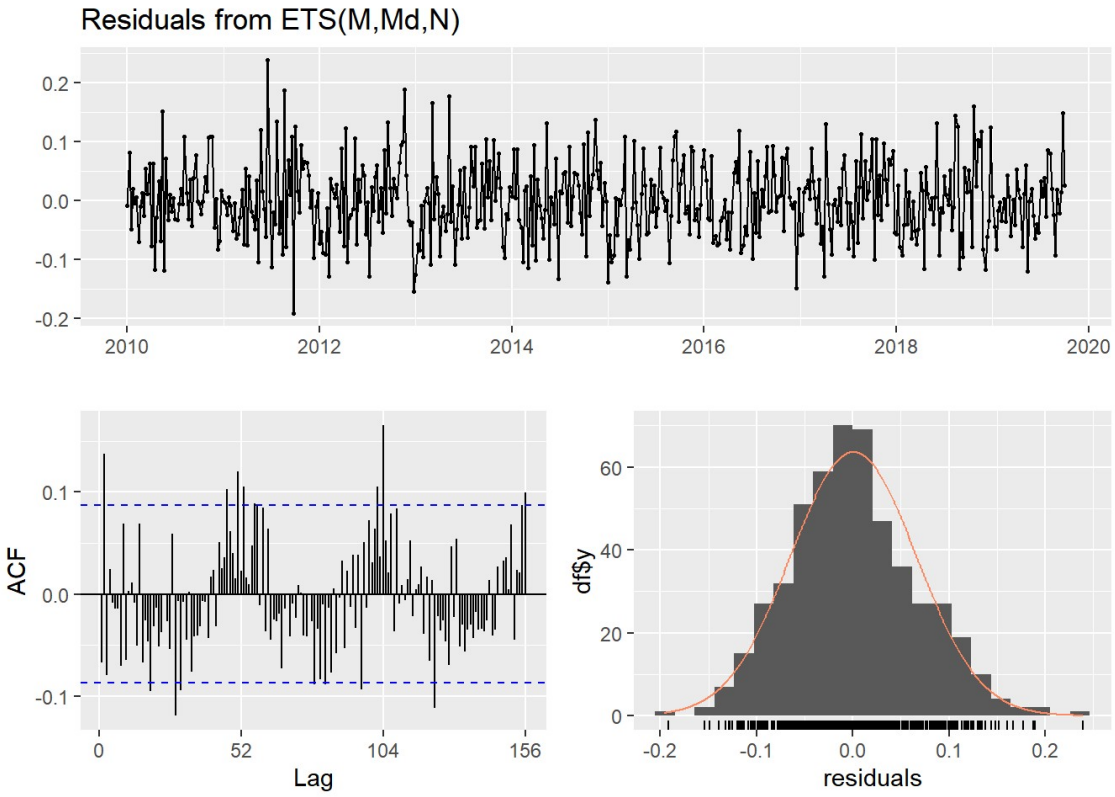



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,A,N)
## Q* = 192.28, df = 102, p-value = 1.655e-07
##
## Model df: 0.   Total lags used: 102
```

```
etsMM = ets(mortality, model="MMN")
summary(etsMM)
```

```
## ETS(M,Md,N)
##
## Call:
##  ets(y = mortality, model = "MMN")
##
##   Smoothing parameters:
##     alpha = 0.4431
##     beta  = 0.0363
##     phi   = 0.8
##
##   Initial states:
##     l = 100.788
##     b = 0.9747
##
##   sigma:  0.0657
##
##      AIC     AICc      BIC
## 4957.923 4958.091 4983.306
##
## Training set error measures:
##                       ME     RMSE      MAE         MPE     MAPE      MASE
## Training set -0.06310061 5.884085 4.599372 -0.3702585 5.163763 0.6887639
##                     ACF1
## Training set -0.02809165
```

```
checkresiduals(etsMM)
```

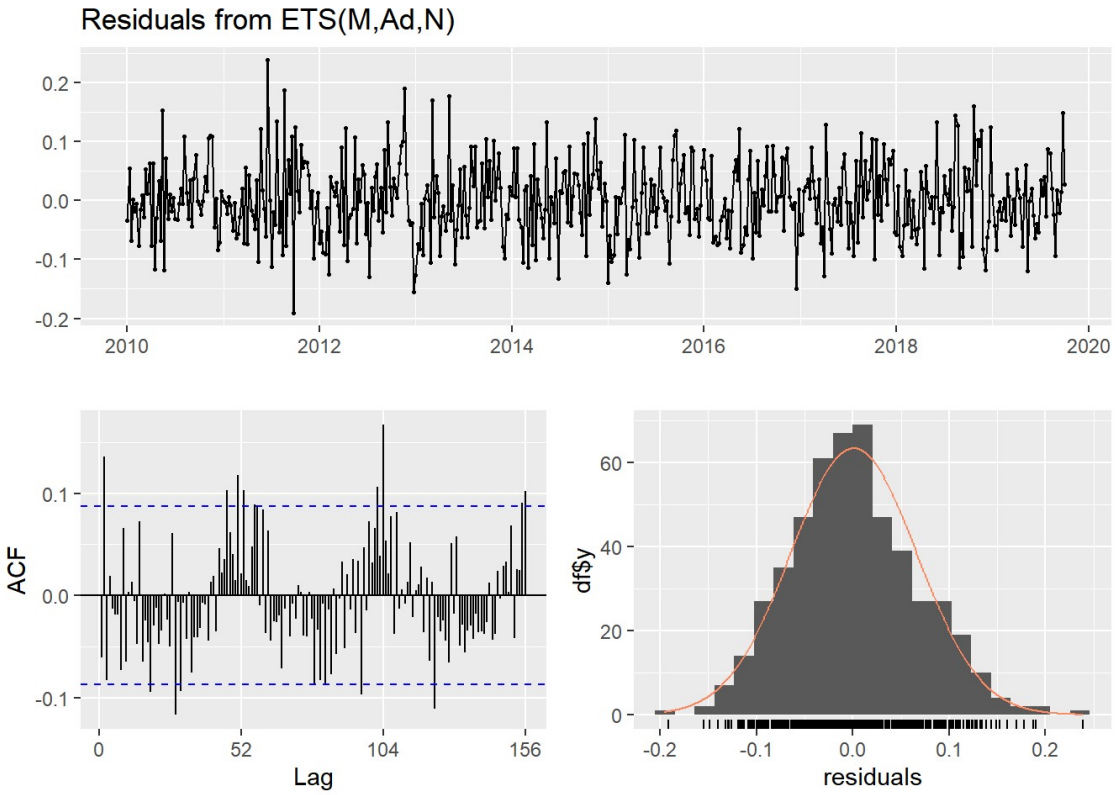


```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,Md,N)
## Q* = 163.48, df = 102, p-value = 0.0001077
##
## Model df: 0.   Total lags used: 102
```

```
etsMA = ets(mortality, model="MAN")
summary(etsMA)
```

```
## ETS(M,Ad,N)
##
## Call:
##  ets(y = mortality, model = "MAN")
##
##   Smoothing parameters:
##     alpha = 0.4311
##     beta  = 0.0441
##     phi   = 0.8
##
##   Initial states:
##     l = 101.9204
##     b = -0.7466
##
##   sigma:  0.0657
##
##      AIC     AICc      BIC
## 4957.818 4957.986 4983.201
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.041505 5.883233 4.604242 -0.3409889 5.167291 0.6894931
##                      ACF1
## Training set -0.02166677
```

```
checkresiduals(etsMA)
```



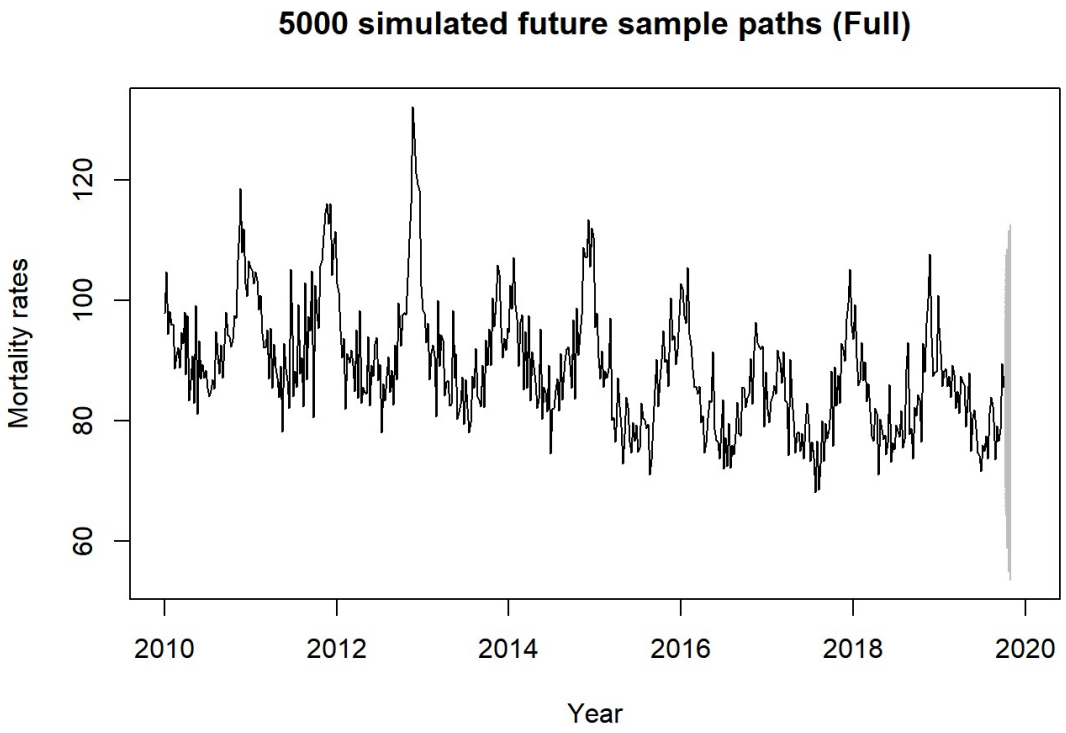

```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,Ad,N)
## Q* = 160.36, df = 102, p-value = 0.0002007
##
## Model df: 0.   Total lags used: 102
```

Model Summary

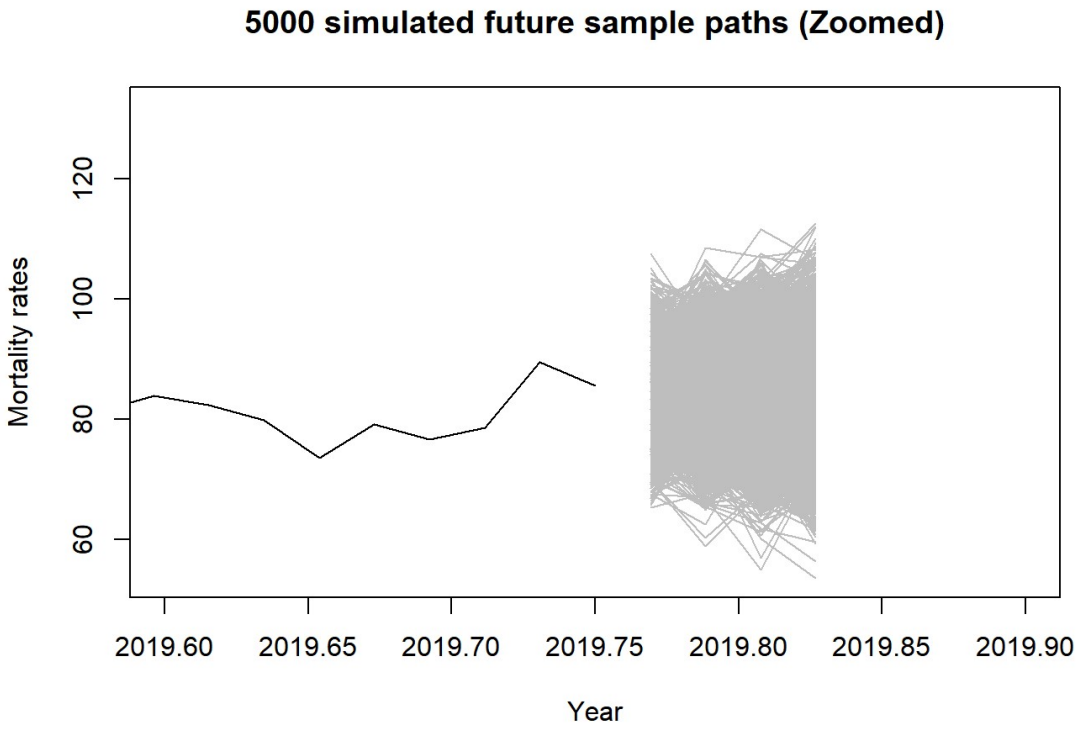
Model	AIC	BIC	MASE	Ljung-Box p-value	Residual Normality
ANN State Space	4971.256	4983.947	0.6871475	1.967e-07	Normal
MNN State Space	4954.111	4966.803	0.6879431	3.201e-09	Non-normal
AAN State Space	4975.460	4996.612	0.6874132	1.655e-07	Normal
MMN State Space	4957.923	4983.306	0.6887639	0.0001077	Normal
MAN State Space	4957.818	4983.201	0.6894931	0.0002007	Normal

The test statistics from the models show very similar values for all the reported metrics, indicating similar performance across all models. The MNN state space model seems to have slightly right skewed residuals and Keun be rejected. All models have a low p-value for the Ljung-Box test, indicating serial correlation in the data for all models due to the null hypothesis being rejected. Despite all models having similar MASE values, it is possible to conclude that the ANN state space model is the best based on its MASE.

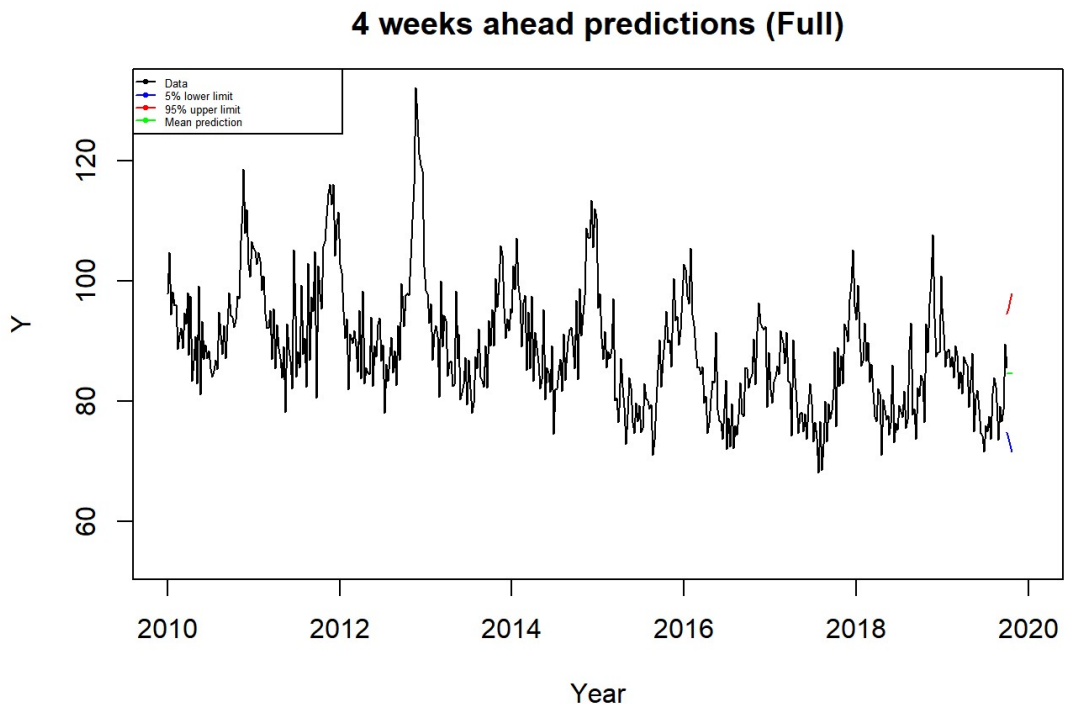
```
A = ts(matrix(NA,4,5000),start=c(2019,41),frequency = 52)

M = 5000
for (i in 1:M){
  A[,i] = simulate(etsAN , initstate = etsAN$states[509,] , nsim=4)
}

plot(mortality , ylim=range(mortality,A) , xlim=c(2010,2020),
     ylab="Mortality rates" , xlab="Year", main="5000 simulated future sample paths (Full)")
for(i in 1:M){
  lines(A[,i],col="gray")
}
```



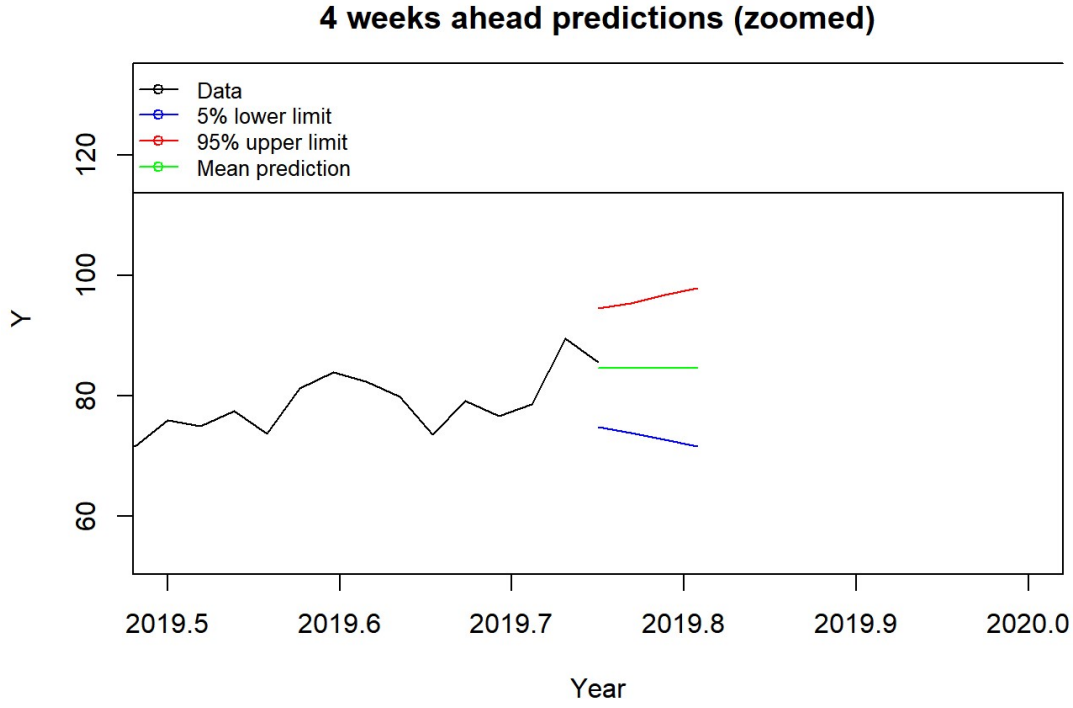

```
plot(mortality , ylim=range(mortality,A) , xlim=c(2019.6,2019.9),
     ylab="Mortality rates" , xlab="Year", main="5000 simulated future sample paths (Zoomed)")
for(i in 1:M){
  lines(A[,i],col="gray")
}
```




```
# interval estimates and bounds

N = 4
xlim=c(2010,2020)

Pi = array(NA, dim=c(N,2))
avrg = array(NA, N)

# Calculate the interval estimates and mid point
for (i in 1:N){
  Pi[i,] = quantile(A[i,], type=8, prob=c(.05,.95))
  avrg[i] = mean(A[i,]) # This would be median as well
}

# Create ts objects for plotting
Pi.lb = ts(Pi[,1],start=end(mortality),f=52)
Pi.ub = ts(Pi[,2],start=end(mortality),f=52)
avrg.pred = ts(avrg,start=end(mortality),f=52)

plot(mortality,xlim=xlim , ylim=range(mortality,A),ylab="Y",xlab="Year", main="
4 weeks ahead predictions (Full)")
lines(Pi.lb,col="blue", type="l")
lines(Pi.ub,col="red", type="l")
lines(avrg.pred,col="green", type="l")
legend("topleft", lty=1, pch=1, col=c("black","blue","red","green"), text.width =2, cex=0.4,
       c("Data","5% lower limit","95% upper limit","Mean prediction"))
```




```
plot(mortality,xlim=c(2019.5,2020) , ylim=range(mortality,A),ylab="Y",xlab="Year", main="
4 weeks ahead predictions (zoomed)")
lines(Pi.lb,col="blue", type="l")
lines(Pi.ub,col="red", type="l")
lines(avrg.pred,col="green", type="l")
legend("topleft", lty=1, pch=1, col=c("black","blue","red","green"), text.width =2, cex=0.8,
       c("Data","5% lower limit","95% upper limit","Mean prediction"))
```



5000 simulated future sample parts are plotted along with the four weeks ahead predictions for the ANN state space model. Zoomed views of each plot are also plotted to be able to analyze the forecast better. The forecast predicts that mortality will remain roughly constant at around 84.6.

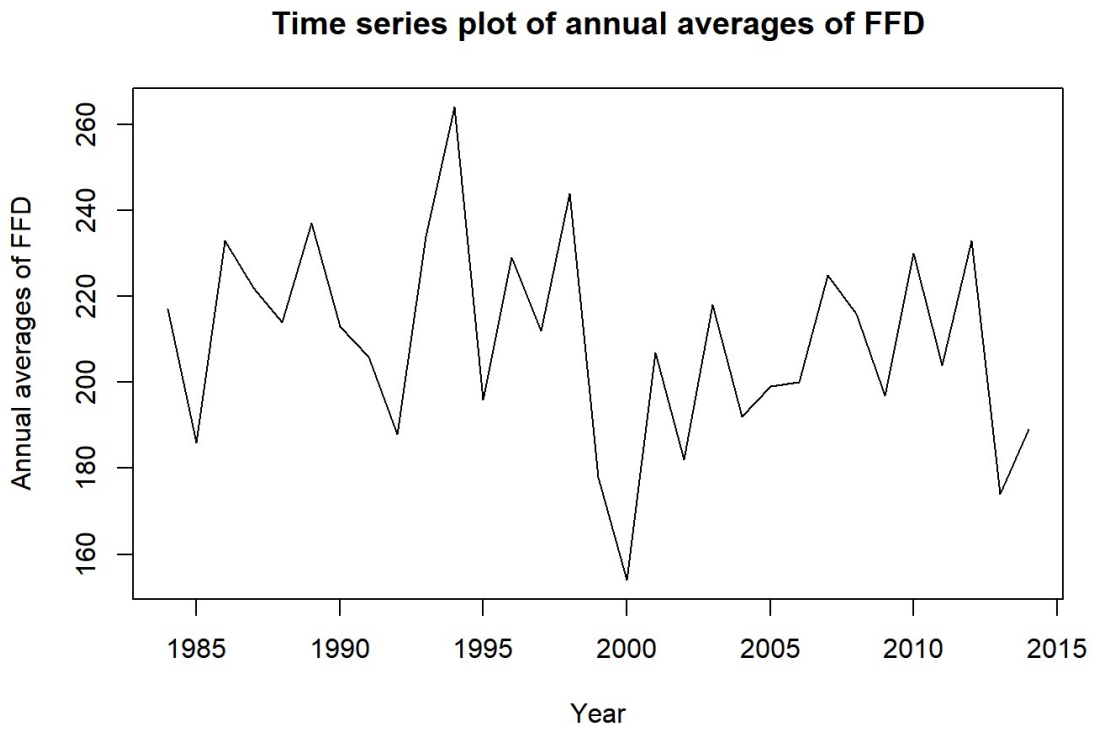
Task 2

The objective of this task is to create 4 weeks ahead forecasts for the first flowering day (FFD) based on multiple univariate models, including time series regression, exponential smoothing, and state space models. The plots of the FFD series and its predictors are given below:

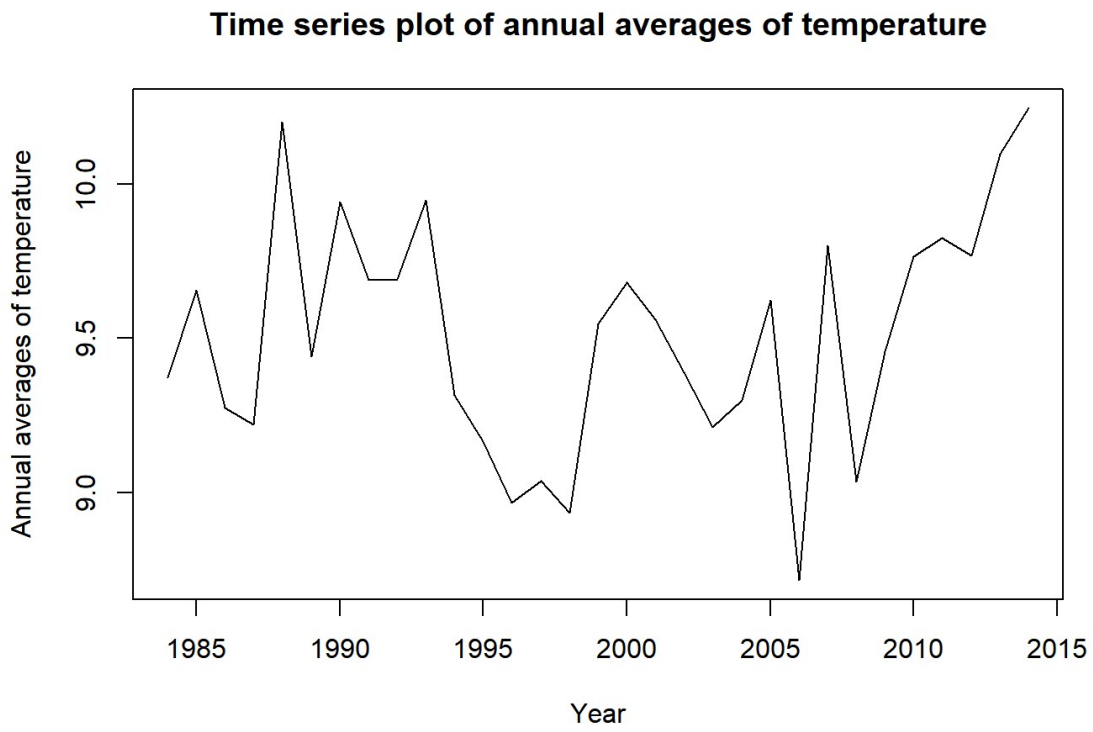
```
ffddata <- read.csv("FFD .csv")
head(ffddata)
```

```
##   Year Temperature Rainfall Radiation RelHumidity FFD
## 1 1984    9.371585 2.489344  14.87158    93.92650 217
## 2 1985    9.656164 2.475890  14.68493    94.93589 186
## 3 1986    9.273973 2.421370  14.51507    94.09507 233
## 4 1987    9.219178 2.319726  14.67397    94.49699 222
## 5 1988   10.202186 2.465301  14.74863    94.08142 214
## 6 1989    9.441096 2.735890  14.78356    96.08685 237
```

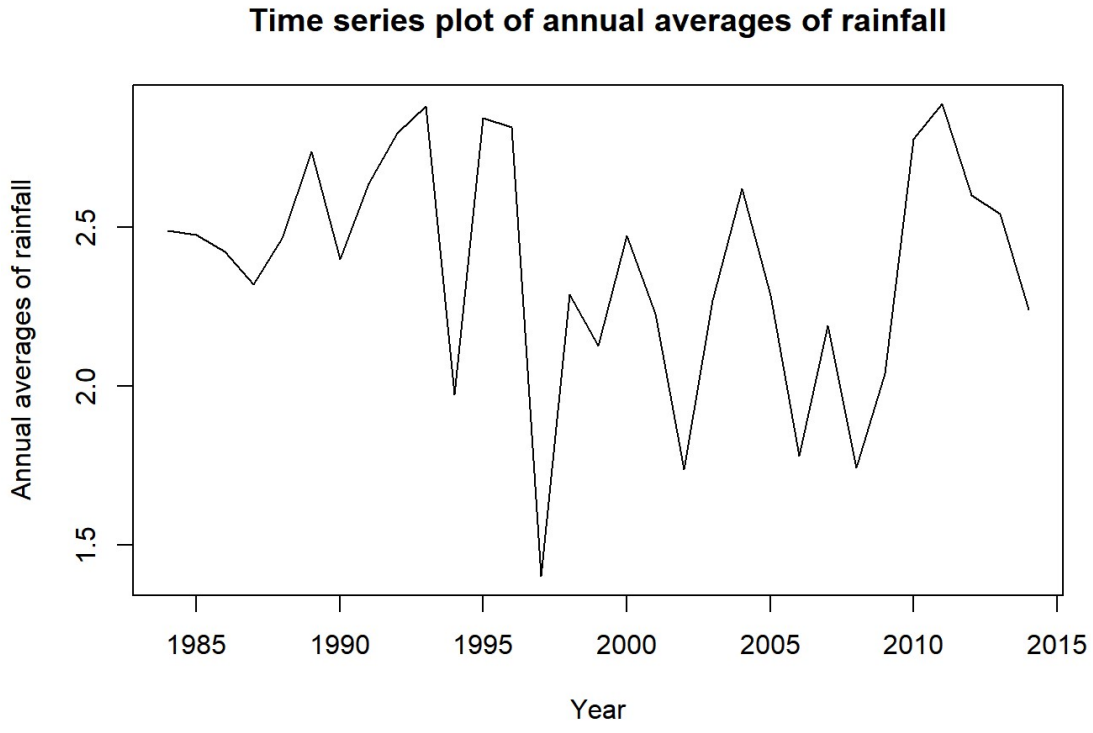
```
ffd <- ts(ffddata$FFD, start=c(1984,1), frequency=1)
temp <- ts(ffddata$Temperature, start=c(1984,1), frequency=1)
rain <- ts(ffddata$Rainfall, start=c(1984,1), frequency=1)
radiation <- ts(ffddata$Radiation, start=c(1984,1), frequency=1)
relhumid <- ts(ffddata$RelHumidity, start=c(1984,1), frequency=1)

plot(ffd, ylab='Annual averages of FFD',xlab='Year',
     main = "Time series plot of annual averages of FFD")
```



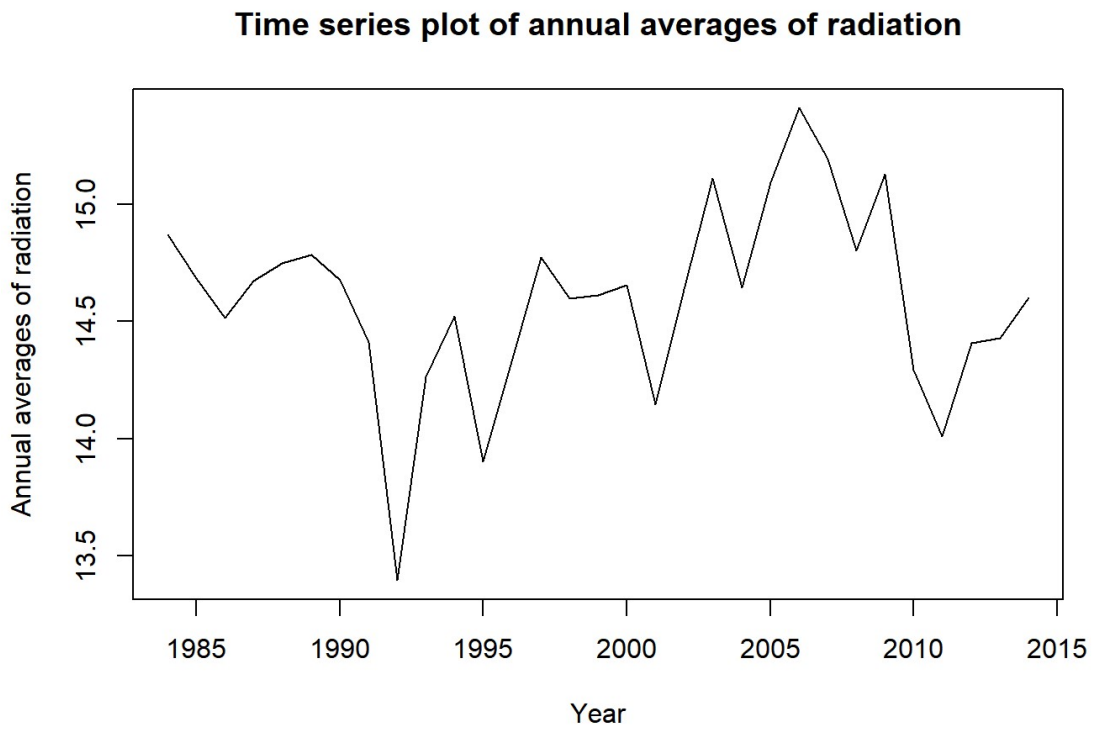

```
plot(temp, ylab='Annual averages of temperature',xlab='Year',
     main = "Time series plot of annual averages of temperature")
```



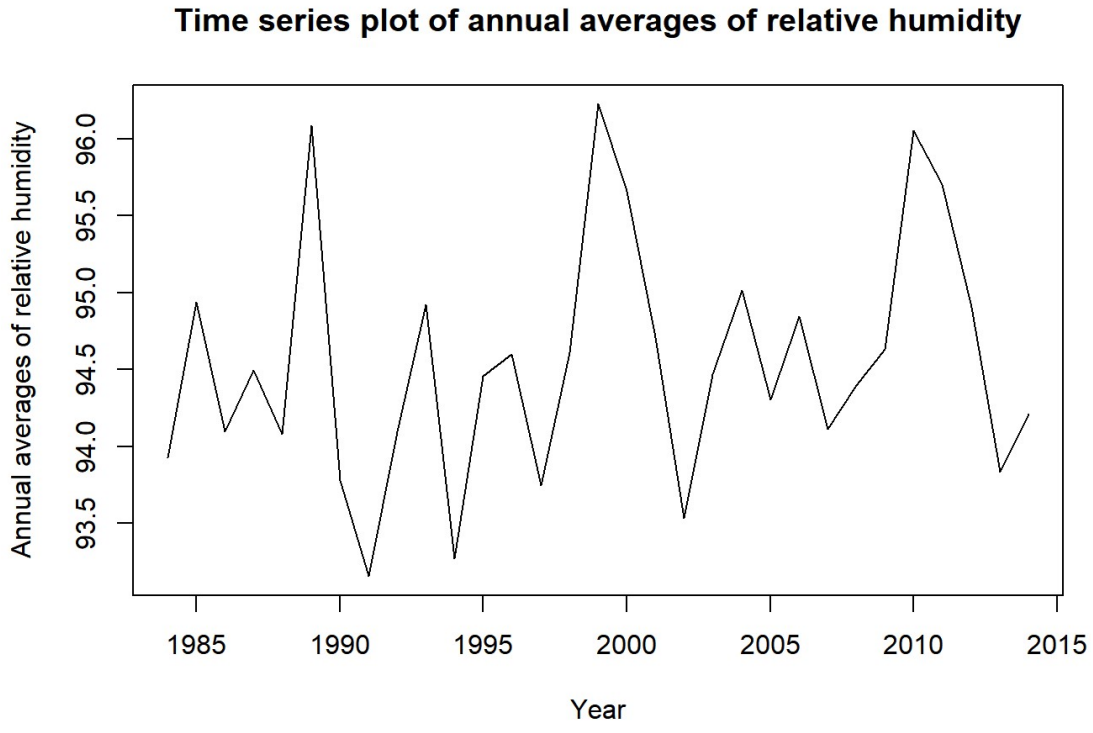

```
plot(rain, ylab='Annual averages of rainfall',xlab='Year',
     main = "Time series plot of annual averages of rainfall")
```




```
plot(radiation, ylab='Annual averages of radiation',xlab='Year',
     main = "Time series plot of annual averages of radiation")
```



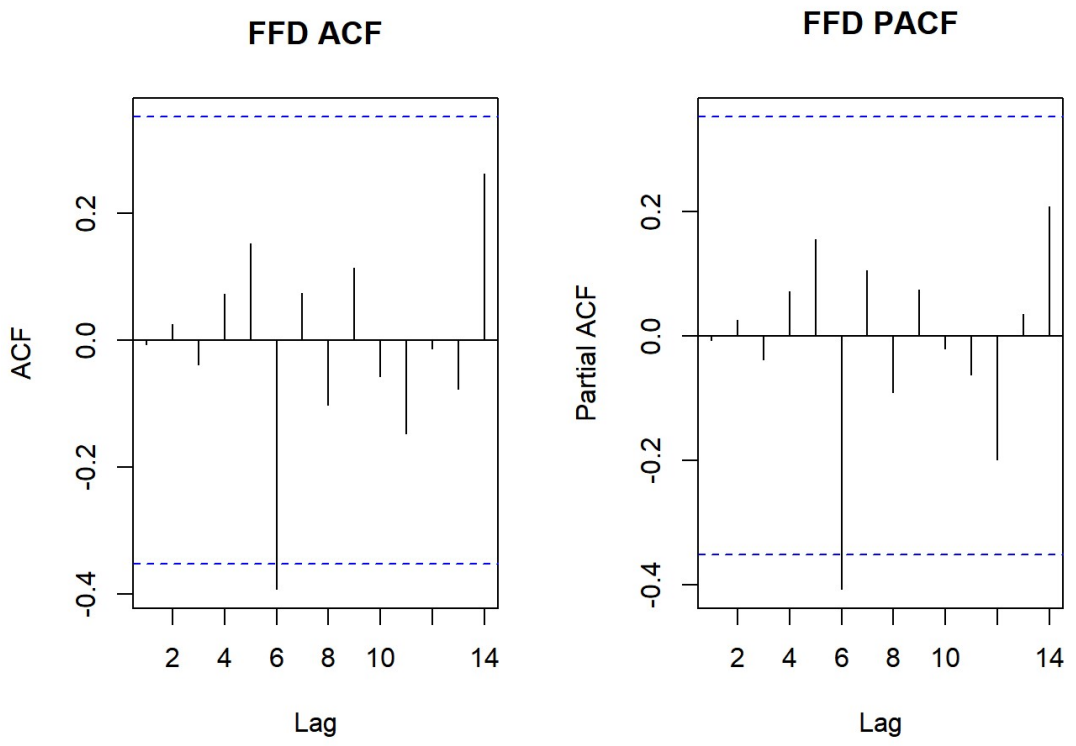

```
plot(relhumid, ylab='Annual averages of relative humidity',xlab='Year',
     main = "Time series plot of annual averages of relative humidity")
```



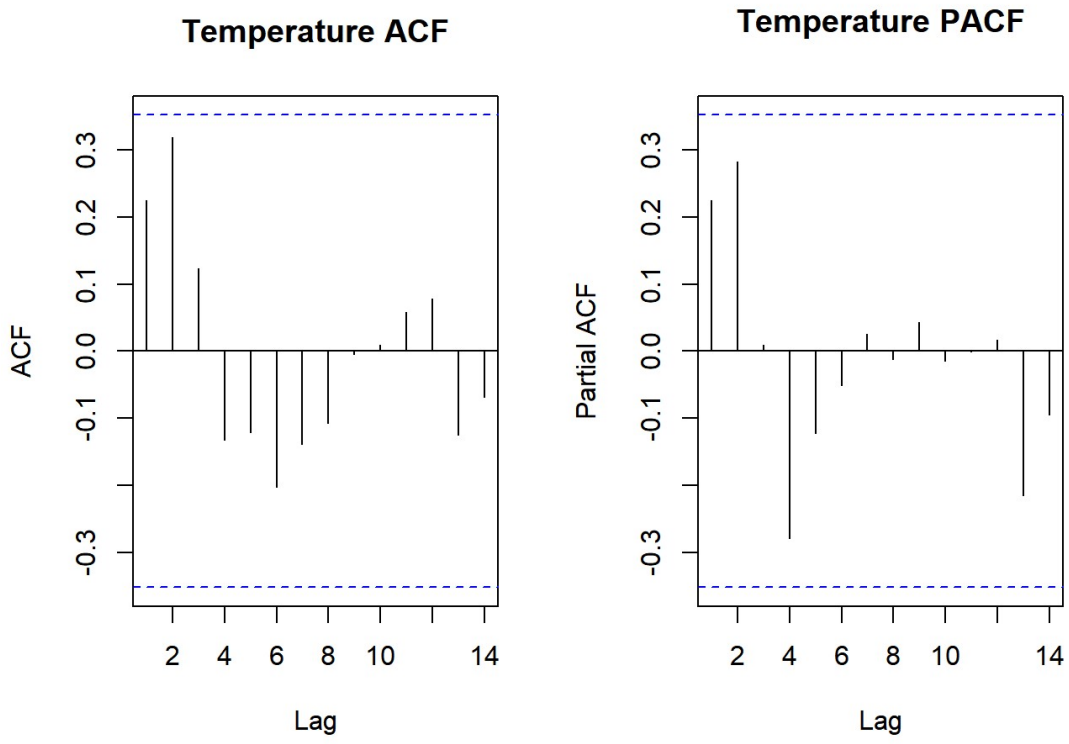
Exploring Stationarity of Variables

Plotting the ACF and PACF for FFD and its predictors demonstrates similar results for all variables. All the ACF and PACF plots have peaks that are not particularly significant, with no obvious seasonal components, Indicating that the series is stationary.

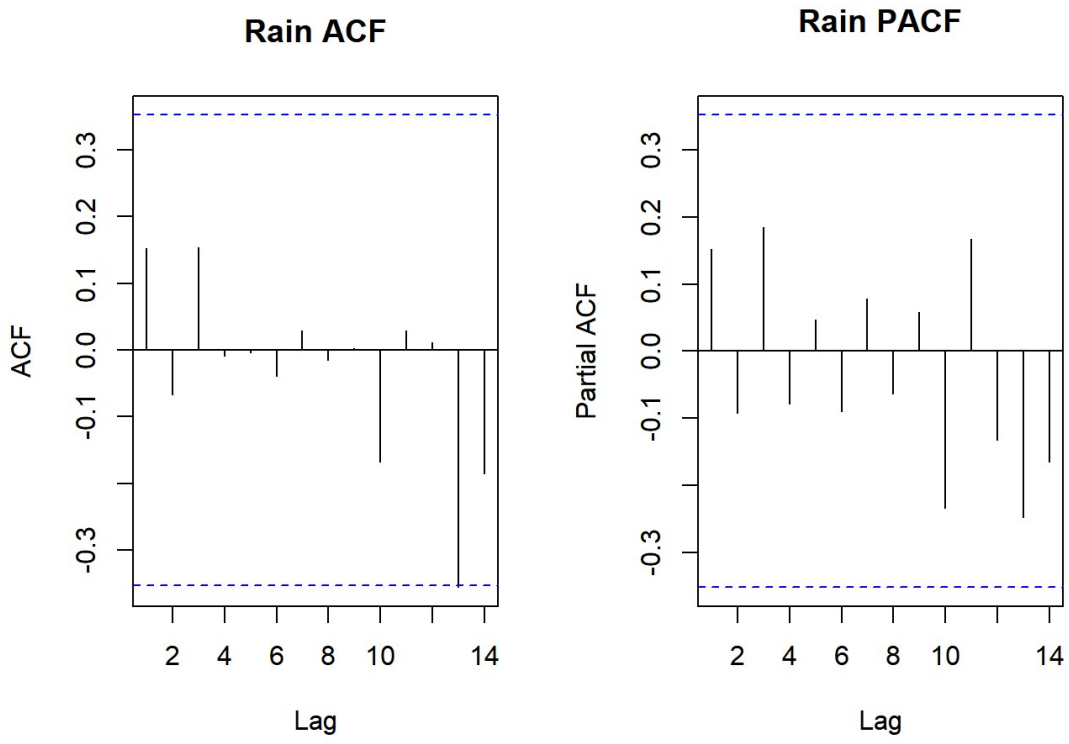
```
par(mfrow=c(1,2))
acf(ffd, max.lag = 24, main = "FFD ACF")
pacf(ffd, max.lag = 24, main = "FFD PACF")
```



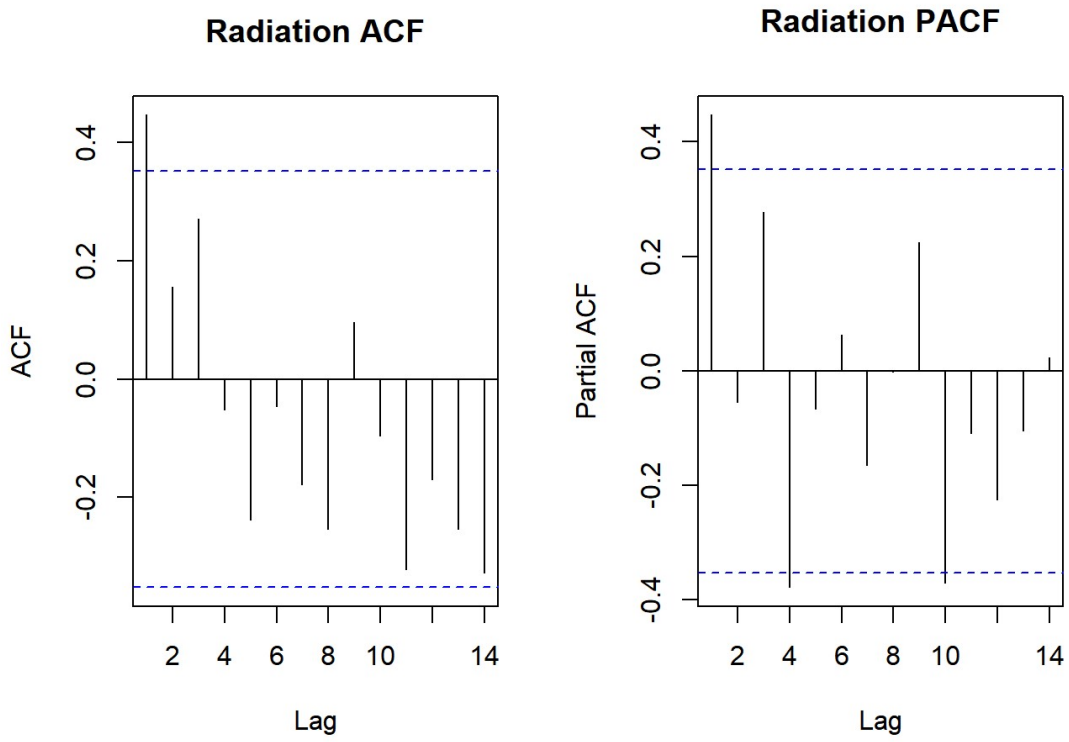

```
par(mfrow=c(1,2))
acf(temp, max.lag = 24, main="Temperature ACF")
pacf(temp, max.lag = 24, main = "Temperature PACF")
```



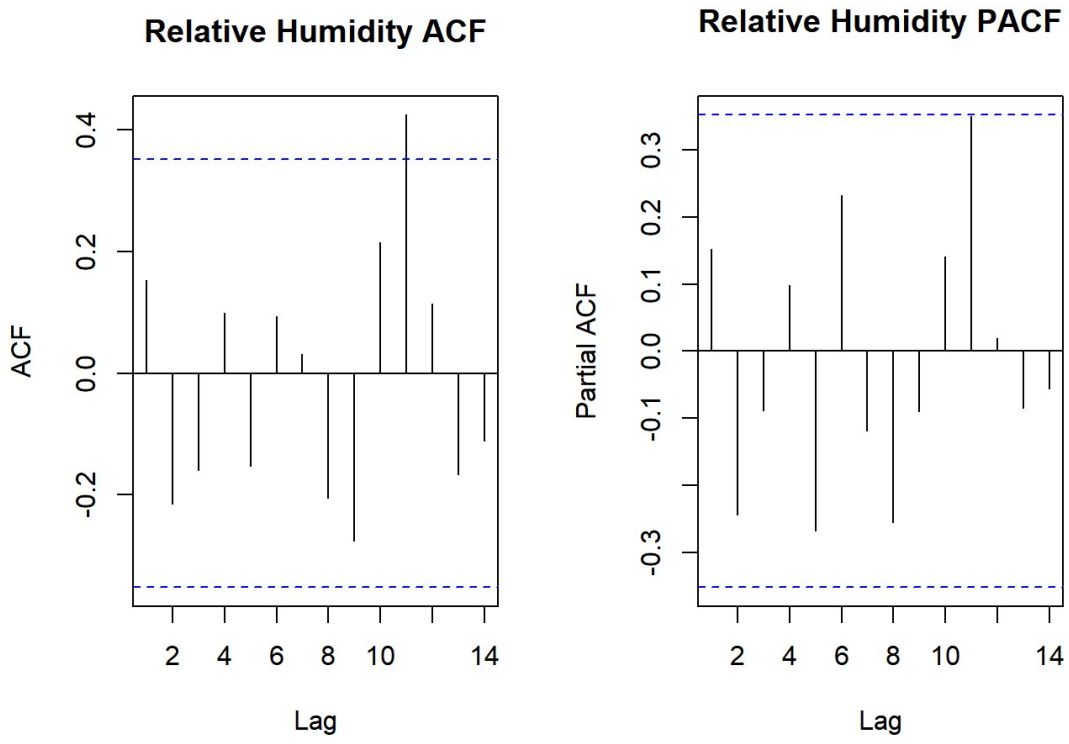

```
par(mfrow=c(1,2))
acf(rain, max.lag = 24, main = "Rain ACF")
pacf(rain, max.lag = 24, main = "Rain PACF")
```




```
par(mfrow=c(1,2))
acf(radiation, max.lag = 24, main = "Radiation ACF")
pacf(radiation, max.lag = 24, main = "Radiation PACF")
```




```
par(mfrow=c(1,2))
acf(relhumid, max.lag = 24, main = "Relative Humidity ACF")
pacf(relhumid, max.lag = 24, main = "Relative Humidity PACF")
```



For further confirmation of non-stationarity in the series, an Augmented Dicky-Fuller test and Phillips Perron unit root test is performed on the series. The augmented Dickey fuller test produces less extreme test statistics compared to the critical values at all significance levels for the FFD and all the predictor series. This implies that the null hypothesis can be rejected and the series are stationary. This is contrasted by the Phillips perron unit root test, which produces a

more extreme test statistic compared to its critical values. Therefore, the null hypothesis cannot be rejected implying that the series are non-stationary. These conflicting results make it difficult to conclude whether the series are truly stationary or not.

```
adf.ffd = ur.df(ffd, type = "none", lags = 1, selectlags = "AIC")
summary(adf.ffd)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -57.274 -14.617   2.381  24.385  56.281
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.008629   0.025397  -0.340  0.73665
## z.diff.lag -0.527429   0.160435  -3.288  0.00281 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.86 on 27 degrees of freedom
## Multiple R-squared:  0.2899, Adjusted R-squared:  0.2373
## F-statistic: 5.512 on 2 and 27 DF,  p-value: 0.009831
##
##
## Value of test-statistic is: -0.3398
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.62 -1.95 -1.61
```

```
pp.ffd = ur.pp(ffd, type = "Z-tau", lags = "short")
summary(pp.ffd)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -55.416 -16.070   0.175  18.712  54.961
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 210.614583  40.412896   5.212 1.56e-05 ***
## y.l1         -0.006732   0.191168  -0.035    0.972
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.19 on 28 degrees of freedom
## Multiple R-squared:  4.429e-05,  Adjusted R-squared:  -0.03567
## F-statistic: 0.00124 on 1 and 28 DF,  p-value: 0.9722
##
##
## Value of test-statistic, type: Z-tau  is: -5.2692
##
##          aux. Z statistics
## Z-tau-mu            5.2148
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.665967 -2.962656 -2.620011
```

```
adf.temp = ur.df(temp, type = "none", lags = 1, selectlags = "AIC")
summary(adf.temp)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.74882 -0.24560 -0.06356  0.26178  0.91921
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## z.lag.1     0.003385   0.007510   0.451 0.655777
## z.diff.lag -0.594821   0.154274  -3.856 0.000648 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3829 on 27 degrees of freedom
## Multiple R-squared:  0.3554, Adjusted R-squared:  0.3076
## F-statistic: 7.443 on 2 and 27 DF,  p-value: 0.002663
##
##
## Value of test-statistic is: 0.4507
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.62 -1.95 -1.61
```

```
app.temp = ur.pp(temp, type = "Z-tau", lags = "short")
summary(pp.temp)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.6204  -4.6804  -0.0509   4.6633  23.7841
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 29.54652    2.65853   11.11   <2e-16 ***
## y.l1         0.60211    0.03554   16.94   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.211 on 505 degrees of freedom
## Multiple R-squared:  0.3624, Adjusted R-squared:  0.3612
## F-statistic: 287.1 on 1 and 505 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic, type: Z-tau  is: -12.0726
##
##          aux. Z statistics
## Z-tau-mu           11.9834
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.445446 -2.867533 -2.569954
```

```
adf.rain = ur.df(rain, type = "none", lags = 1, selectlags = "AIC")
summary(adf.rain)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3753 -0.1936  0.1544  0.3340  0.8803
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.01662    0.03684  -0.451   0.6555
## z.diff.lag -0.36299    0.18035  -2.013   0.0542 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4736 on 27 degrees of freedom
## Multiple R-squared:  0.1428, Adjusted R-squared:  0.07929
## F-statistic: 2.249 on 2 and 27 DF,  p-value: 0.1249
##
##
## Value of test-statistic is: -0.4511
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.62 -1.95 -1.61
```

```
pp.rain = ur.pp(rain, type = "Z-tau", lags = "short")
summary(pp.rain)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0306 -0.1530  0.0554  0.2690  0.5381
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.0033     0.4491   4.461 0.000121 ***
## y.l1          0.1529     0.1868   0.818 0.420073
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3814 on 28 degrees of freedom
## Multiple R-squared:  0.02336,    Adjusted R-squared:  -0.01152
## F-statistic: 0.6697 on 1 and 28 DF,  p-value: 0.4201
##
##
## Value of test-statistic, type: Z-tau  is: -4.5072
##
##          aux. Z statistics
## Z-tau-mu            4.4335
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.665967 -2.962656 -2.620011
```

```
adf.radiation = ur.df(radiation, type = "none", lags = 1, selectlags = "AIC")
summary(adf.radiation)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.06546 -0.27928  0.06367  0.34134  0.64395
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.0007715  0.0054907  -0.141    0.889
## z.diff.lag -0.2325212  0.1870728  -1.243    0.225
##
## Residual standard error: 0.4312 on 27 degrees of freedom
## Multiple R-squared:  0.05456,    Adjusted R-squared:  -0.01548
## F-statistic: 0.779 on 2 and 27 DF,  p-value: 0.4689
##
##
## Value of test-statistic is: -0.1405
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.62 -1.95 -1.61
```

```
pp.radiation = ur.pp(radiation, type = "Z-tau", lags = "short")
summary(pp.radiation)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.10563 -0.09934  0.05929  0.19569  0.60439
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.0639     2.4425   3.302  0.00263 **
## y.l1          0.4467     0.1673   2.670  0.01249 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3705 on 28 degrees of freedom
## Multiple R-squared:  0.2029, Adjusted R-squared:  0.1745
## F-statistic: 7.128 on 1 and 28 DF,  p-value: 0.01249
##
##
## Value of test-statistic, type: Z-tau  is: -3.2491
##
##          aux. Z statistics
## Z-tau-mu             3.244
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.665967 -2.962656 -2.620011
```

```
adf.relhumid = ur.df(relhumid, type = "none", lags = 1, selectlags = "AIC")
summary(adf.relhumid)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7309 -0.7848  0.1133  0.6584  1.9218
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.0003151  0.0019849  -0.159    0.875
## z.diff.lag -0.2724937  0.1822047  -1.496    0.146
##
## Residual standard error: 1.011 on 27 degrees of freedom
## Multiple R-squared:  0.07733,    Adjusted R-squared:  0.008987
## F-statistic: 1.131 on 2 and 27 DF,  p-value: 0.3374
##
##
## Value of test-statistic is: -0.1587
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.62 -1.95 -1.61
```

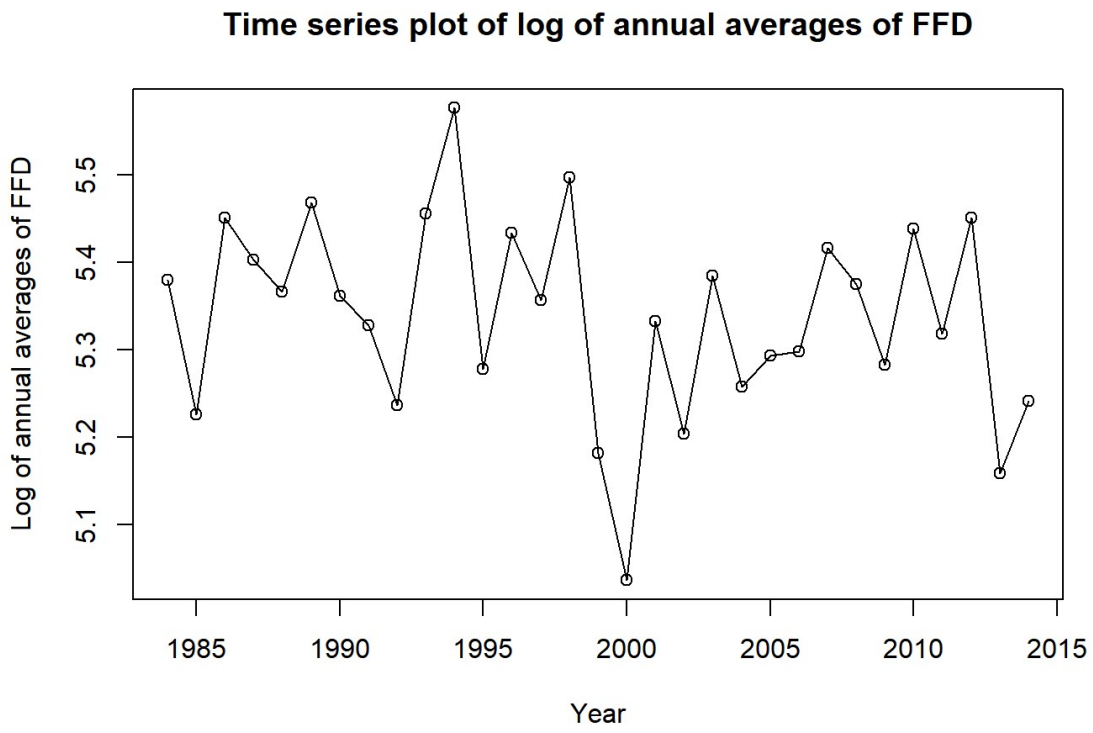
```
pp.relhumid = ur.pp(relhumid, type = "Z-tau", lags = "short")
summary(pp.relhumid)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.35103 -0.49219  0.02665  0.39864  1.65341
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  80.0768    17.5215   4.570 8.97e-05 ***
## y.l1          0.1532     0.1853   0.827    0.415
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7954 on 28 degrees of freedom
## Multiple R-squared:  0.02384,    Adjusted R-squared:  -0.01103
## F-statistic: 0.6837 on 1 and 28 DF,  p-value: 0.4153
##
##
## Value of test-statistic, type: Z-tau  is: -4.5348
##
##          aux. Z statistics
## Z-tau-mu            4.5352
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.665967 -2.962656 -2.620011
```

Applying Transformations to FFD

A log transformation, a Box Cox transformation, and differencing are performed to the FFD series to see if any improvements occur to the Phillips Perron test. However, similar results are obtained with a more extreme test statistic being produced compared to each test's respective critical values, implying that according to the Phillips Perron test, the series is non-stationary.

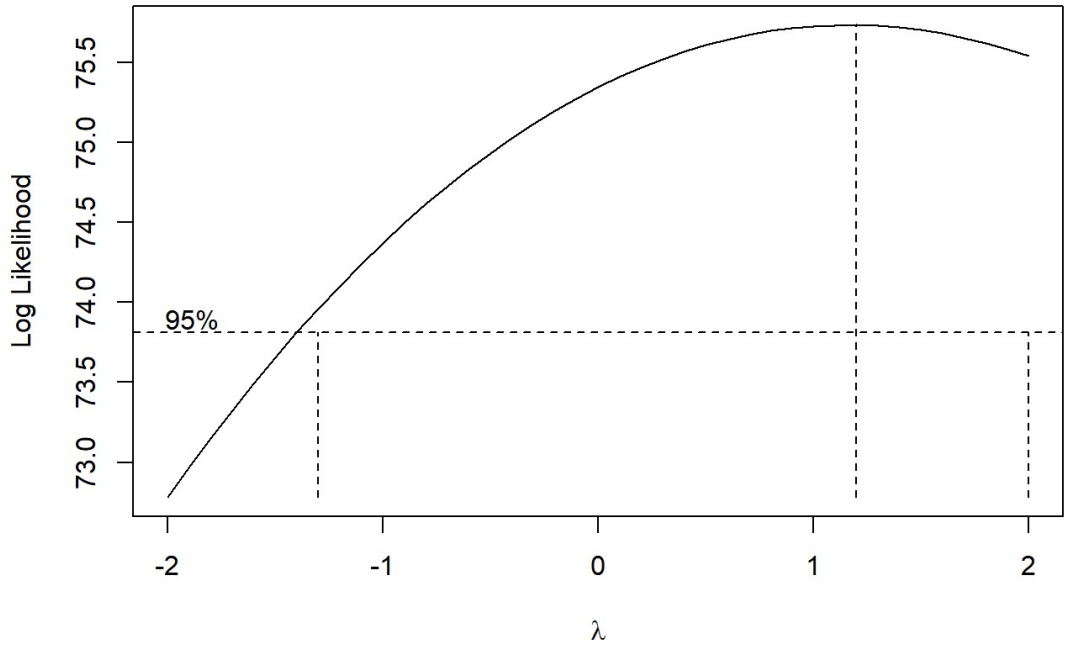
```
# Trying a log transformation to make the series stationary
log.ffd = log(ffd)
plot(log.ffd,ylab='Log of annual averages of FFD',xlab='Year',type='o',
     main = "Time series plot of log of annual averages of FFD")
```




```
pp.log = ur.pp(log.ffd, type = "Z-tau", lags = "short")
summary(pp.log)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.29788 -0.07546  0.00912  0.09316  0.23754
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.26701    1.01998   5.164 1.77e-05 ***
## y.l1         0.01309    0.19091   0.069    0.946
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1176 on 28 degrees of freedom
## Multiple R-squared:  0.0001678,  Adjusted R-squared:  -0.03554
## F-statistic: 0.0047 on 1 and 28 DF,  p-value: 0.9458
##
##
## Value of test-statistic, type: Z-tau  is: -5.1745
##
##          aux. Z statistics
## Z-tau-mu            5.1689
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.665967 -2.962656 -2.620011
```

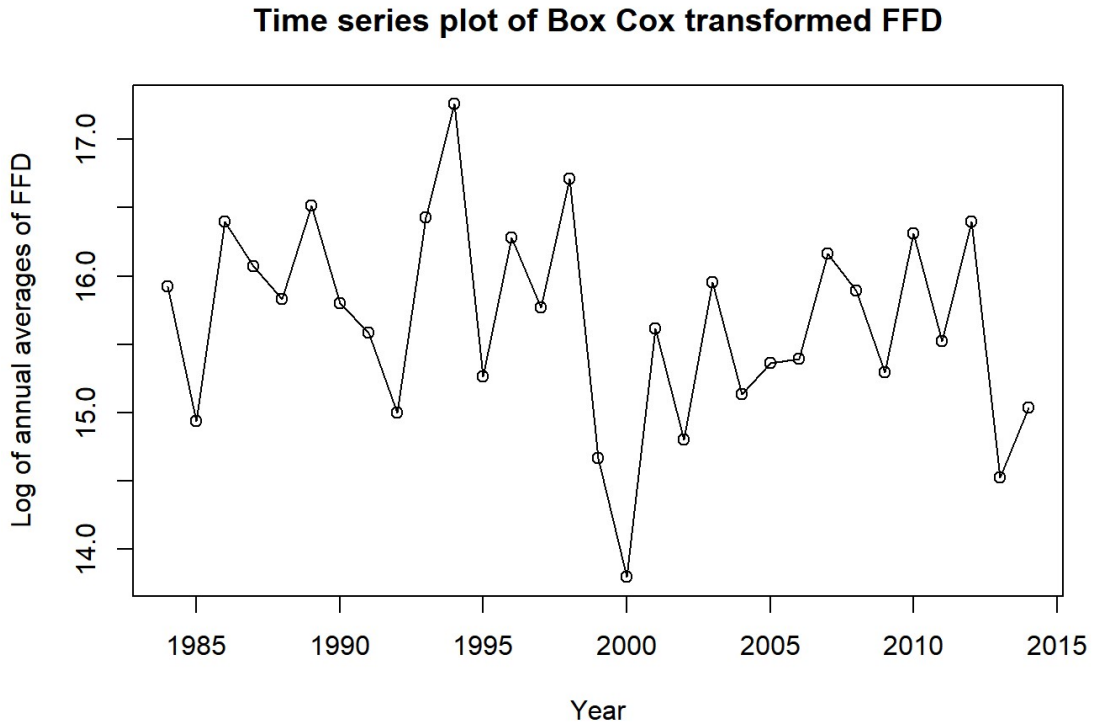
```
# Trying a Box Cox transformation to make the series stationary
bc.ffd = BoxCox.ar(ffd)
```




```
bc.ffd$ci
```

```
## [1] -1.3  2.0
```

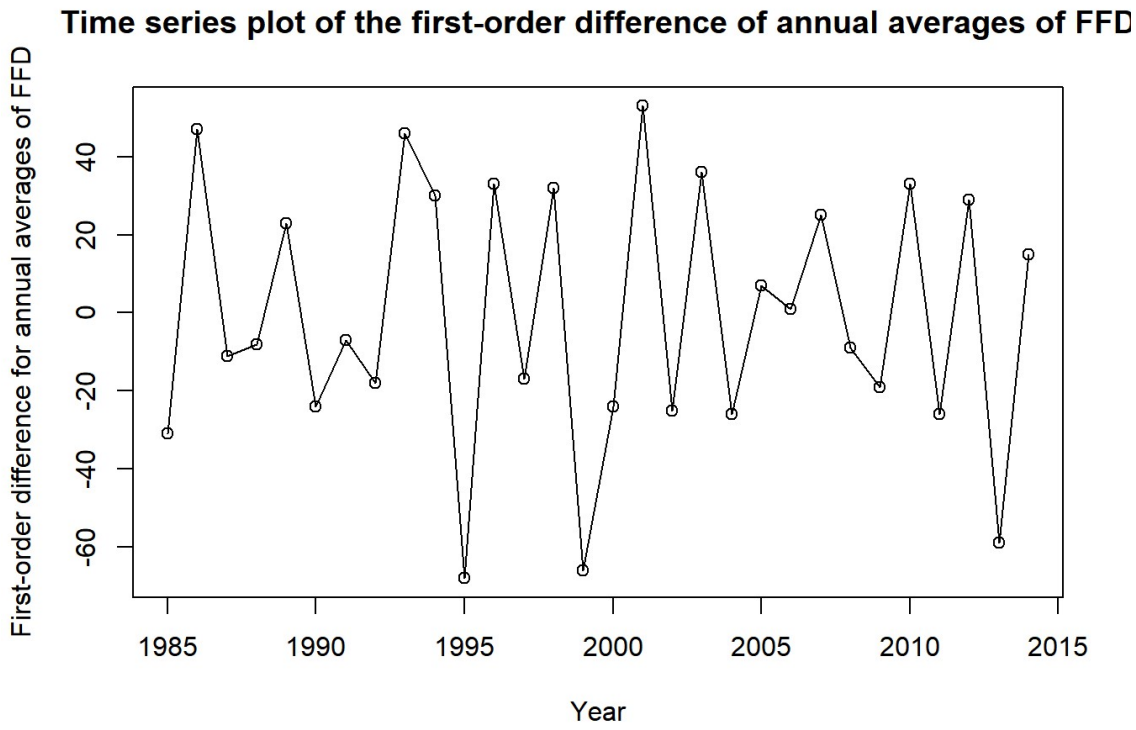
```
bc.ffd = BoxCox(ffd, lambda = 0.35)
plot(bc.ffd, ylab='Log of annual averages of FFD',xlab='Year',type='o',
     main = "Time series plot of Box Cox transformed FFD")
```




```
pp.bc = ur.pp(bc.ffd, type = "Z-tau", lags = "short")
summary(pp.bc)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.85235 -0.49265  0.03951  0.59580  1.59535
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 15.565034   2.999591   5.189 1.65e-05 ***
## y.l1         0.005859   0.191018   0.031    0.976
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.757 on 28 degrees of freedom
## Multiple R-squared:  3.36e-05,   Adjusted R-squared:  -0.03568
## F-statistic: 0.0009408 on 1 and 28 DF,  p-value: 0.9757
##
##
## Value of test-statistic, type: Z-tau  is: -5.2087
##
##          aux. Z statistics
## Z-tau-mu            5.1935
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.665967 -2.962656 -2.620011
```

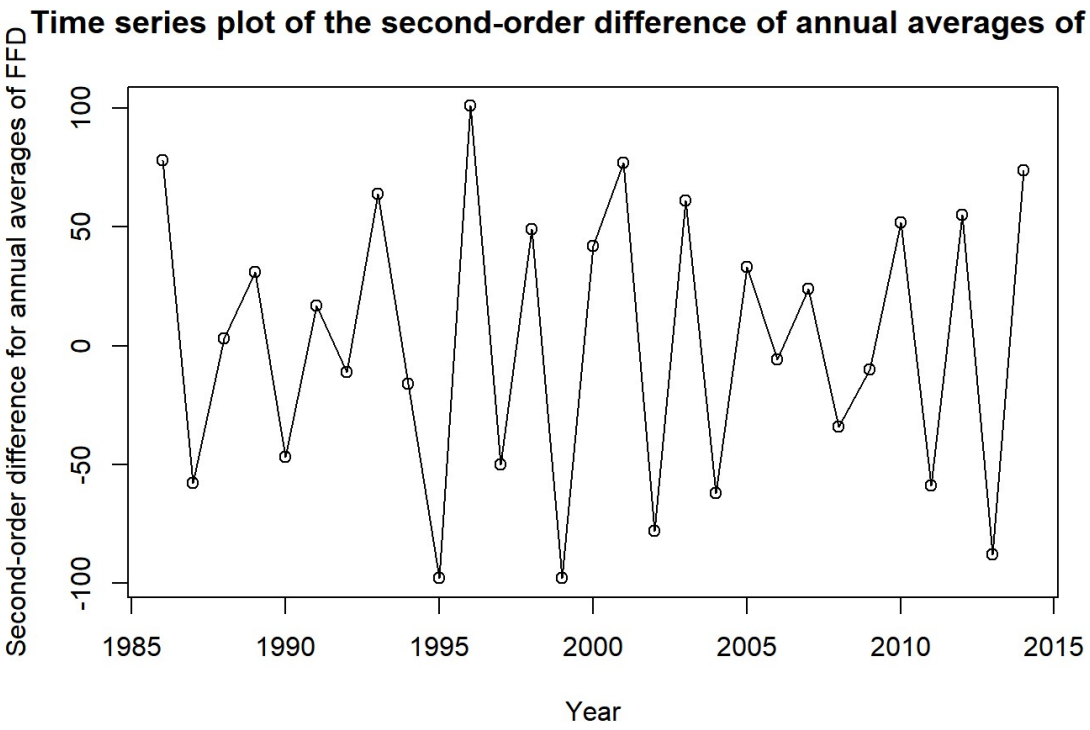
```
# Trying a first-order differencing transformation to make the series stationary
diff1.ffd = diff(ffd, differences = 1, lag = 1)
plot(diff1.ffd, ylab='First-order difference for annual averages of FFD', xlab='Year', type='o',
     main = "Time series plot of the first-order difference of annual averages of FFD")
```




```
pp.diff1 = ur.pp(diff1.ffd, type = "Z-tau", lags = "short")
summary(pp.diff1)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -58.324 -15.611   1.187  23.423  55.082
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.6830     5.3752  -0.127  0.89983
## y.l1         -0.5304     0.1608  -3.299  0.00273 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.92 on 27 degrees of freedom
## Multiple R-squared:  0.2873, Adjusted R-squared:  0.2609
## F-statistic: 10.88 on 1 and 27 DF,  p-value: 0.002726
##
##
## Value of test-statistic, type: Z-tau  is: -10.7898
##
##          aux. Z statistics
## Z-tau-mu             -0.16
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.675142 -2.966454 -2.622013
```

```
# Trying a second-order differencing transformation to make the series stationary
diff2.ffd = diff(ffd, differences = 2, lag = 1)
plot(diff2.ffd, ylab='Second-order difference for annual averages of FFD', xlab='Year', type='o',
     main = "Time series plot of the second-order difference of annual averages of FFD")
```



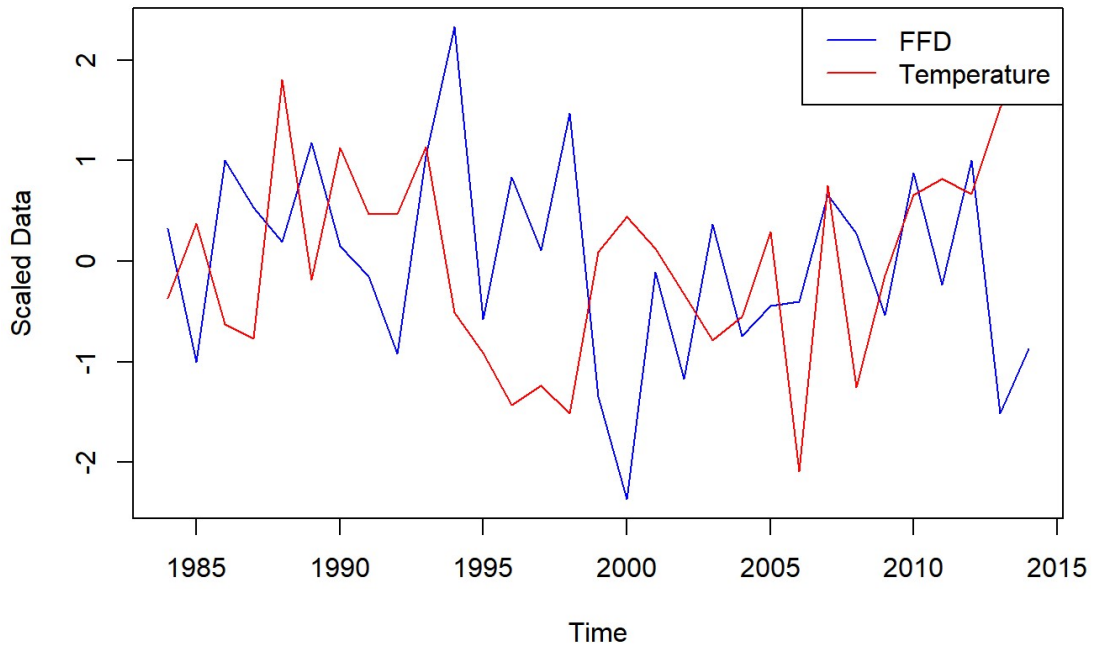

```
pp.diff2 = ur.pp(diff2.ffd, type = "Z-tau", lags = "short")
summary(pp.diff2)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -107.498  -21.934    1.047   22.072  108.646
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.8522     7.9111  -0.234    0.817
## y.l1         -0.7094     0.1373  -5.166 2.17e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 41.86 on 26 degrees of freedom
## Multiple R-squared:  0.5066, Adjusted R-squared:  0.4876
## F-statistic: 26.69 on 1 and 26 DF,  p-value: 2.168e-05
##
##
## Value of test-statistic, type: Z-tau  is: -18.2389
##
##          aux. Z statistics
## Z-tau-mu            -0.341
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.685059 -2.970549 -2.624171
```

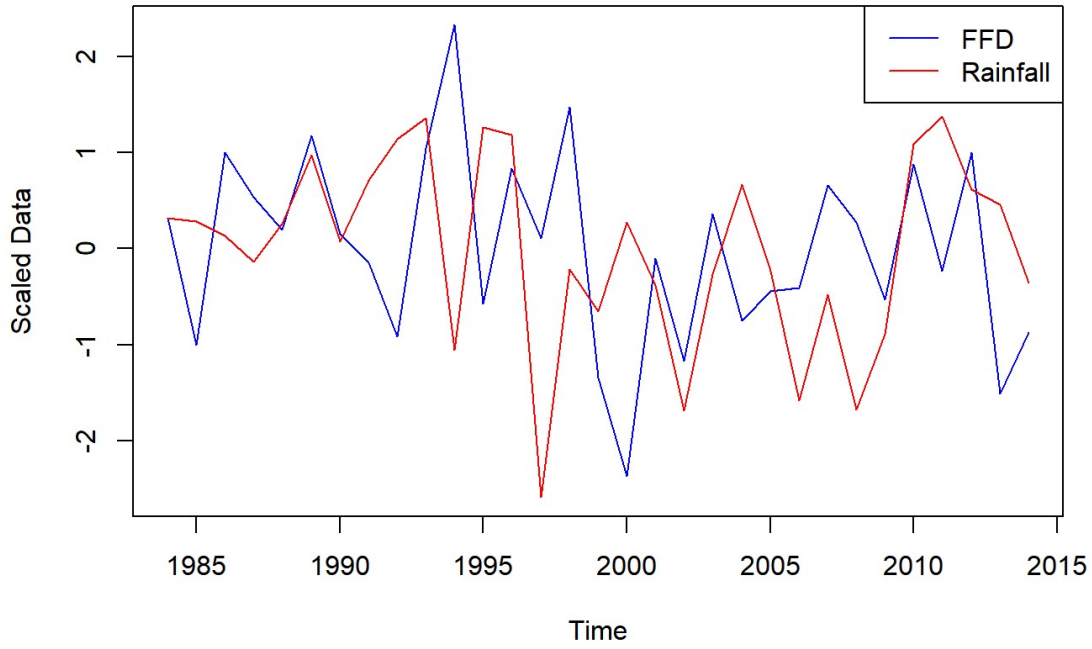
Correlating Predictor Variables with Dependent Variable

Inspecting the scaled plots of FFD versus its predictors is challenging due to the lack of data. However, all the predictors seem to follow the overall trend of FFD except for a significant peak in relative humidity around the year 2000 which contrasts the significant dip in the FFD series.

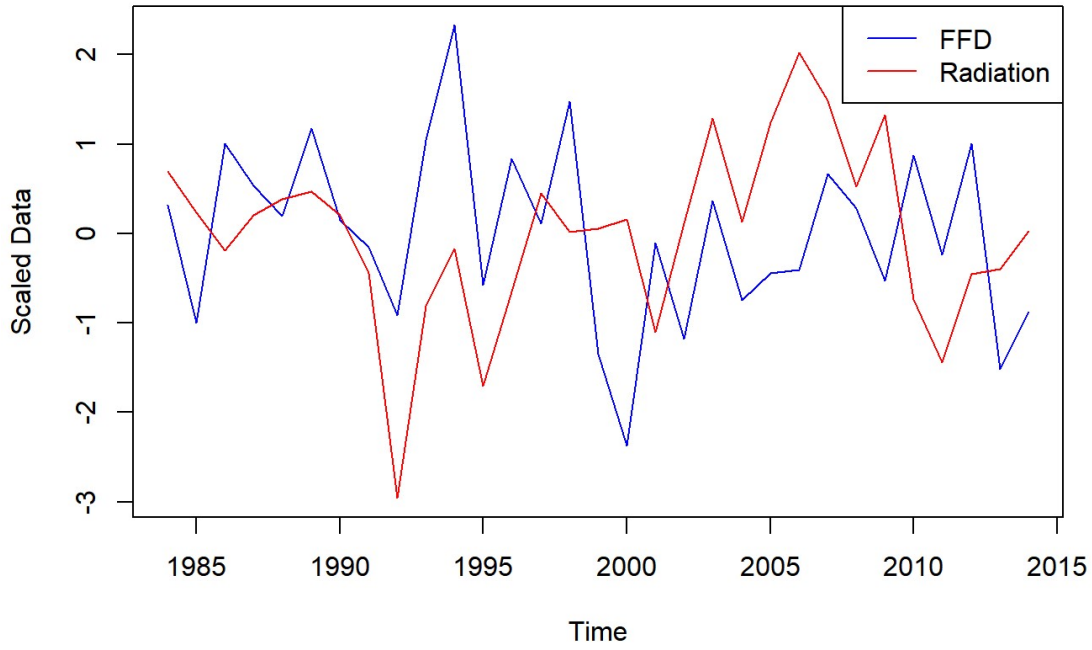
```
ffddata.ts <- ts(ffddata[,c(6,2)], start=c(1984,1), frequency=1)
ffddata_scaled = scale(ffddata.ts)
plot(ffddata_scaled, plot.type="s", col=c("blue", "red"), ylab="Scaled Data")
legend("topright", lty=1, col=c("blue", "red"), c("FFD", "Temperature"))
```



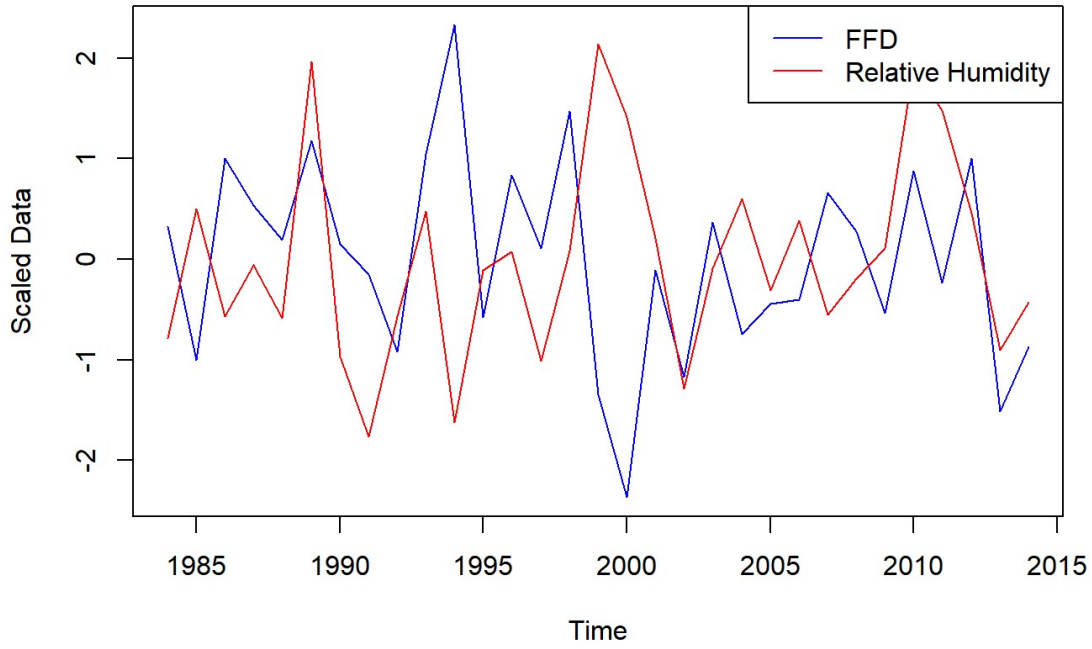

```
ffddata.ts <- ts(ffddata[,c(6,3)], start=c(1984,1), frequency=1)
ffddata_scaled = scale(ffddata.ts)
plot(ffddata_scaled, plot.type="s", col=c("blue", "red"), ylab="Scaled Data")
legend("topright", lty=1, col=c("blue", "red"), c("FFD", "Rainfall"))
```




```
ffddata.ts <- ts(ffddata[,c(6,4)], start=c(1984,1), frequency=1)
ffddata_scaled = scale(ffddata.ts)
plot(ffddata_scaled, plot.type="s", col=c("blue", "red"), ylab="Scaled Data")
legend("topright", lty=1, col=c("blue", "red"), c("FFD", "Radiation"))
```




```
ffddata.ts <- ts(ffddata[,c(6,5)], start=c(1984,1), frequency=1)
ffddata_scaled = scale(ffddata.ts)
plot(ffddata_scaled, plot.type="s", col=c("blue", "red"), ylab="Scaled Data")
legend("topright", lty=1, col=c("blue", "red"), c("FFD", "Relative Humidity"))
```



Since a visual confirmation of the correlation between predictor and dependent variables is insufficient, the variables can be correlated using the cor() function in order to obtain a numerical representation of correlation. Obtaining the correlation values shows that the most correlated predictor for the FFD series is temperature, with a negative correlation of -0.25. Overall however, none of the predictors seem to be highly correlated with FFD.

```
#correlate variables
ffddata.ts <- ts(ffddata[,2:6], start=c(1984,1), frequency=1)
cor(ffddata.ts)
```

```
##              Temperature   Rainfall   Radiation  RelHumidity         FFD
## Temperature  1.000000000  0.3933255 -0.24096625  0.009646021 -0.24793371
## Rainfall     0.393325545  1.0000000 -0.58131610  0.338461007  0.05069110
## Radiation   -0.240966245 -0.5813161  1.00000000 -0.055209652  0.04677758
## RelHumidity  0.009646021  0.3384610 -0.05520965  1.000000000 -0.12850244
## FFD         -0.247933708  0.0506911  0.04677758 -0.128502440  1.00000000
```

Fitting Distributed Lag Models

Polynomial DLM

Polynomial DLM models are first fitted using the polyDlm() function, using each predictor variable to predict FFD. Prior to fitting, the ideal lag value is found using the finiteDLMauto() function, which is made to generate multiple models with q ranging from 1 to 10 and uses AIC error to determine the suitability of each model. The results of finiteDLMauto() suggest that the ideal value for q is 10, and a polynomial order of 2.

```
#PolyDLM
finiteDLMauto(x = as.vector(temp), y = as.vector(ffd), q.min = 1, q.max = 10,
              model.type = "poly", error.type = "AIC", trace = TRUE)
```

```
##     q - k    MASE      AIC      BIC   GMRAE    MBRAE R.Adj.Sq Ljung-Box
## 10 10 - 2 0.64611 201.1493 206.3719 0.62565  5.94451  0.00982 0.2298487
## 9   9 - 2 0.63784 208.9273 214.3825 0.62023  0.86646  0.08087 0.4114741
## 8   8 - 2 0.64195 218.7617 224.4392 0.66309 -1.44874  0.03961 0.3026767
## 7   7 - 2 0.61771 225.7006 231.5909 0.58135  0.63816  0.08959 0.3502603
## 6   6 - 2 0.63263 234.0051 240.0995 0.63589 -5.92662  0.08508 0.2914524
## 5   5 - 2 0.65485 243.9958 250.2863 0.65033  0.64858  0.07077 0.3689050
## 4   4 - 2 0.64935 254.8350 261.3141 0.55368  0.97253 -0.02563 0.7664009
## 3   3 - 2 0.64680 262.6923 269.3533 0.59497  1.27728 -0.00240 0.8948984
## 2   2 - 2 0.67881 272.7512 279.5877 0.61846 -1.47353 -0.03404 0.9864821
## 1   1 - 2 0.66448 279.6699 285.2747 0.60244 -0.73174  0.01369 0.9600714
```

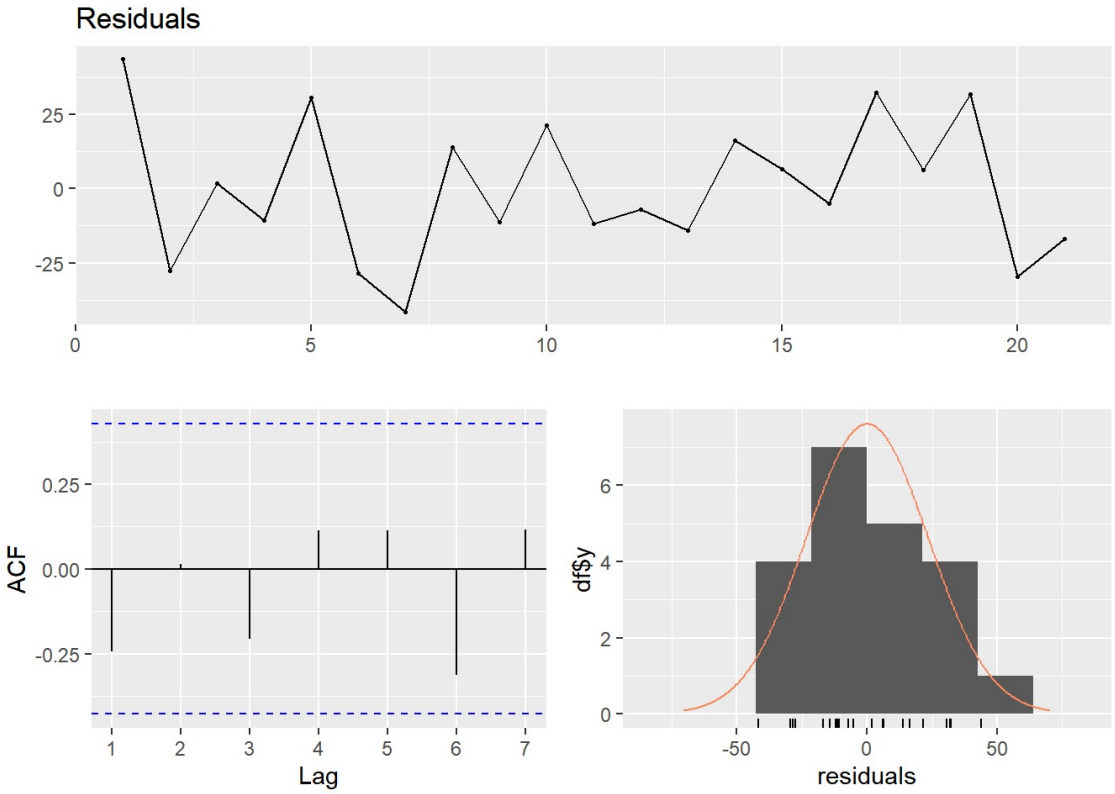
```
model.poly1 = polyDlm(x = as.vector(temp), y = as.vector(ffd), q = 10, k = 2, show.beta = TRUE)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error  t value P(>|t|)
## beta.0   -6.2400       9.09 -0.68600   0.507
## beta.1   -0.6310       6.86 -0.09200   0.928
## beta.2    3.7500       5.93  0.63100   0.541
## beta.3    6.8900       5.90  1.17000   0.267
## beta.4    8.8100       6.10  1.44000   0.177
## beta.5    9.5000       6.15  1.55000   0.151
## beta.6    8.9700       5.96  1.51000   0.160
## beta.7    7.2000       5.73  1.26000   0.235
## beta.8    4.2000       6.03  0.69700   0.500
## beta.9   -0.0216       7.49 -0.00289   0.998
## beta.10  -5.4700      10.30 -0.53200   0.605
```

```
summary(model.poly1)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -41.641 -14.230  -5.104  16.274  43.700
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -141.0618   533.0958  -0.265    0.794
## z.t0          -6.2363     9.0912  -0.686    0.502
## z.t1           6.2197     3.9083   1.591    0.130
## z.t2          -0.6143     0.3942  -1.559    0.138
##
## Residual standard error: 25.48 on 17 degrees of freedom
## Multiple R-squared:  0.1583, Adjusted R-squared:  0.009816
## F-statistic: 1.066 on 3 and 17 DF,  p-value: 0.3896
```

```
checkresiduals(model.poly1$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 7
##
## data:  Residuals
## LM test = 6.2838, df = 7, p-value = 0.507
```

```
MASE(model.poly1)
```

```
##                  MASE
## model.poly1 0.6461061
```

```
finiteDLMauto(x = as.vector(rain), y = as.vector(ffd), q.min = 1, q.max = 10,
              model.type = "poly", error.type = "AIC", trace = TRUE)
```

```
##     q - k    MASE      AIC      BIC   GMRAE    MBRAE R.Adj.Sq Ljung-Box
## 10 10 - 2 0.62489 201.0167 206.2393 0.57674  0.29011  0.01605 0.1599825
## 9   9 - 2 0.64986 211.4687 216.9239 0.60771  0.50737 -0.03168 0.5366235
## 8   8 - 2 0.65351 221.8263 227.5038 0.59957  2.23108 -0.09728 0.6860296
## 7   7 - 2 0.64075 230.1033 235.9935 0.60783  0.52048 -0.09372 0.7573639
## 6   6 - 2 0.63920 238.2623 244.3567 0.61624  0.51243 -0.08477 0.7461894
## 5   5 - 2 0.63124 247.2862 253.5766 0.49357  1.10159 -0.05459 0.8947076
## 4   4 - 2 0.63527 254.6528 261.1320 0.53498  0.67596 -0.01873 0.7504302
## 3   3 - 2 0.63320 263.1955 269.8565 0.50458  0.43263 -0.02058 0.9460660
## 2   2 - 2 0.67527 273.8231 280.6596 0.54561 -1.34330 -0.07297 0.9002345
## 1   1 - 2 0.67589 281.9769 287.5817 0.63528  4.41731 -0.06515 0.9475115
```

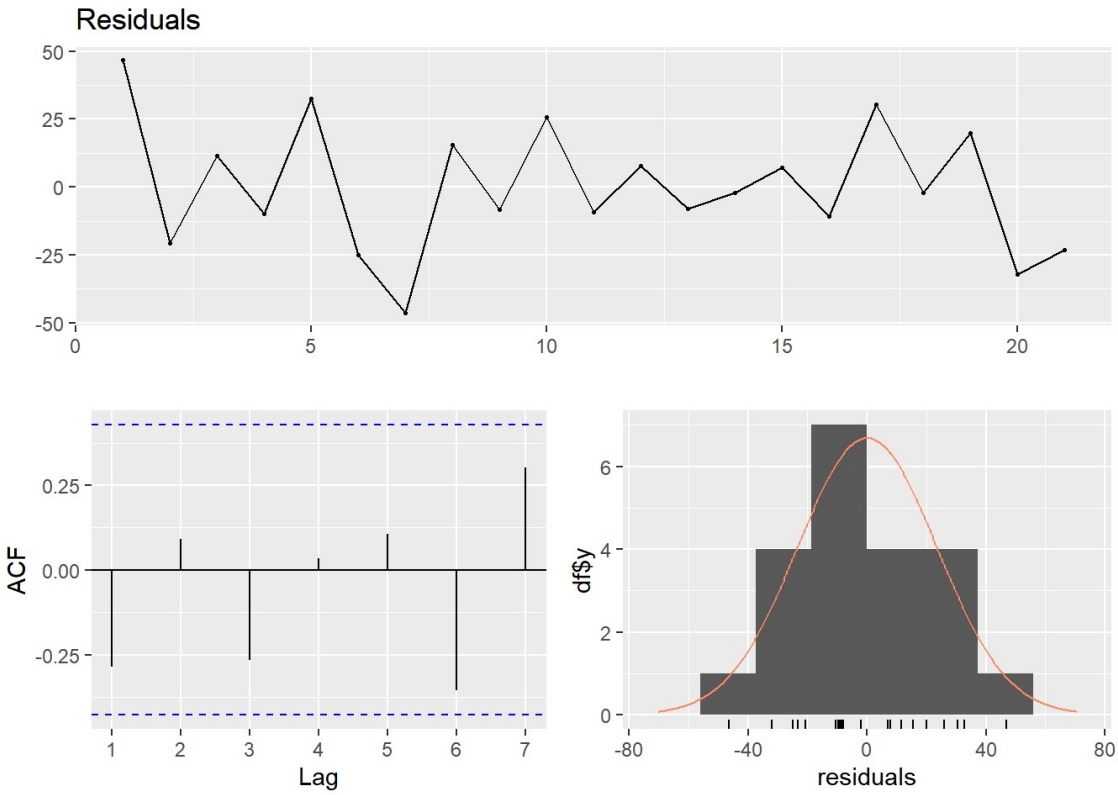
```
model.poly2 = polyDlm(x = as.vector(rain), y = as.vector(ffd), q = 10, k = 2, show.beta = TRUE)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value P(>|t|)
## beta.0     -3.23      10.90  -0.297   0.772
## beta.1      2.47       7.22   0.341   0.739
## beta.2      6.48       5.64   1.150   0.275
## beta.3      8.82       5.69   1.550   0.149
## beta.4      9.47       6.06   1.560   0.146
## beta.5      8.44       5.99   1.410   0.187
## beta.6      5.73       5.35   1.070   0.308
## beta.7      1.33       4.64   0.287   0.779
## beta.8     -4.74       5.38  -0.882   0.397
## beta.9    -12.50       8.55  -1.460   0.171
## beta.10   -21.90      13.60  -1.620   0.134
```

```
summary(model.poly2)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.494 -10.638  -2.134  15.543  46.891
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 209.0810   100.8835   2.073   0.0537 .
## z.t0         -3.2319    10.8654  -0.297   0.7697
## z.t1          6.5390     5.4708   1.195   0.2484
## z.t2         -0.8410     0.5707  -1.474   0.1589
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.4 on 17 degrees of freedom
## Multiple R-squared:  0.1636, Adjusted R-squared:  0.01605
## F-statistic: 1.109 on 3 and 17 DF,  p-value: 0.3729
```

```
checkresiduals(model.poly2$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 7
##
## data:  Residuals
## LM test = 14.791, df = 7, p-value = 0.03878
```

```
MASE(model.poly2)
```

```
##                  MASE
## model.poly2 0.6248857
```

```
finiteDLMauto(x = as.vector(radiation), y = as.vector(ffd), q.min = 1, q.max = 10,
              model.type = "poly", error.type = "AIC", trace = TRUE)
```

```
##     q - k    MASE      AIC      BIC   GMRAE    MBRAE R.Adj.Sq Ljung-Box
## 10 10 - 2 0.64637 203.4282 208.6508 0.57315  0.95439 -0.10369 0.5486715
## 9   9 - 2 0.66971 212.7091 218.1644 0.66270  0.22871 -0.09152 0.7944352
## 8   8 - 2 0.66297 222.2164 227.8938 0.65463  0.78710 -0.11604 0.7860092
## 7   7 - 2 0.64440 230.7400 236.6303 0.65229 -1.11535 -0.12313 0.8915555
## 6   6 - 2 0.64881 239.3461 245.4405 0.59796  0.54349 -0.13283 0.8795683
## 5   5 - 2 0.65041 246.0710 252.3614 0.60886 -1.73648 -0.00643 0.7387404
## 4   4 - 2 0.60270 254.5356 261.0148 0.46170  0.49120 -0.01432 0.8450362
## 3   3 - 2 0.62286 263.5785 270.2395 0.48732  0.66921 -0.03463 0.8989931
## 2   2 - 2 0.66726 273.2458 280.0823 0.51313 -1.85015 -0.05182 0.7148337
## 1   1 - 2 0.65796 281.5710 287.1758 0.56645  1.41699 -0.05083 0.9681143
```

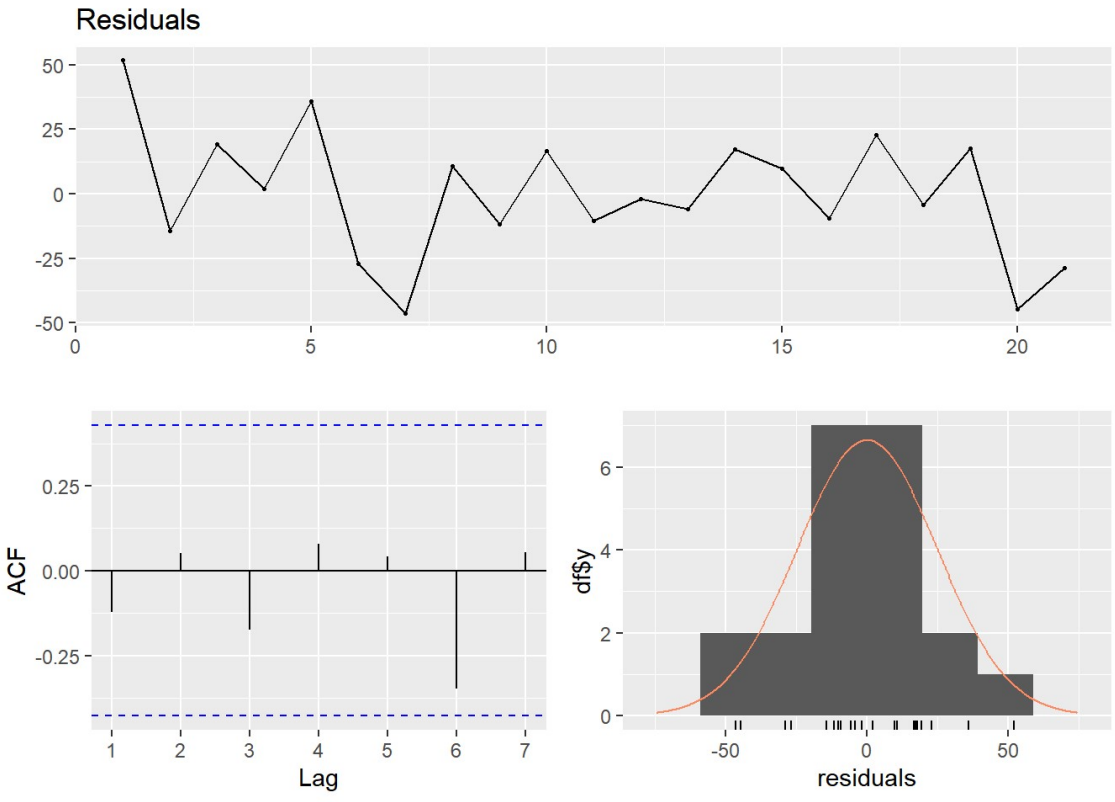
```
model.poly3 = polyDlm(x = as.vector(radiation), y = as.vector(ffd), q = 10, k = 2, show.beta = TRUE)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error  t value P(>|t|)
## beta.0   -0.4200       8.60 -0.04880   0.962
## beta.1   -0.6790       5.50 -0.12300   0.904
## beta.2   -0.6930       3.82 -0.18200   0.859
## beta.3   -0.4620       3.58 -0.12900   0.900
## beta.4    0.0143       3.90  0.00367   0.997
## beta.5    0.7360       4.03  0.18300   0.858
## beta.6    1.7000       3.80  0.44800   0.663
## beta.7    2.9100       3.57  0.81500   0.432
## beta.8    4.3700       4.28  1.02000   0.329
## beta.9    6.0700       6.50  0.93300   0.371
## beta.10   8.0200      10.00  0.80000   0.440
```

```
summary(model.poly3)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.417 -11.818  -1.942  17.330  51.949
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -106.9869   505.1443  -0.212    0.835
## z.t0          -0.4198     8.6001  -0.049    0.962
## z.t1          -0.3816     4.1407  -0.092    0.928
## z.t2           0.1225     0.4193   0.292    0.774
##
## Residual standard error: 26.9 on 17 degrees of freedom
## Multiple R-squared:  0.06187,    Adjusted R-squared:  -0.1037
## F-statistic: 0.3737 on 3 and 17 DF,  p-value: 0.773
```

```
checkresiduals(model.poly3$model)
```



```
##
##  Breusch-Godfrey test for serial correlation of order up to 7
##
## data:  Residuals
## LM test = 6.0156, df = 7, p-value = 0.5379
```

```
MASE(model.poly3)
```

```
##                  MASE
## model.poly3 0.6463697
```

```
finiteDLMauto(x = as.vector(relhumid), y = as.vector(ffd), q.min = 1, q.max = 10,
              model.type = "poly", error.type = "AIC", trace = TRUE)
```

```
##     q - k    MASE      AIC      BIC   GMRAE    MBRAE R.Adj.Sq Ljung-Box
## 10 10 - 2 0.65210 200.4011 205.6237 0.69016  0.92894  0.04447 0.2332720
## 9   9 - 2 0.61667 206.7468 212.2020 0.65685 -0.47485  0.16760 0.3878827
## 8   8 - 2 0.60011 218.1006 223.7781 0.52223  0.15688  0.06682 0.2153014
## 7   7 - 2 0.61830 227.0295 232.9198 0.65552  0.12682  0.03776 0.3453723
## 6   6 - 2 0.63052 235.0038 241.0982 0.71060  0.33281  0.04779 0.2942177
## 5   5 - 2 0.63760 244.2742 250.5647 0.64546  0.24934  0.06077 0.3692789
## 4   4 - 2 0.61984 252.2204 258.6996 0.65948  0.26626  0.06903 0.3441396
## 3   3 - 2 0.60545 258.9412 265.6022 0.59503  0.27634  0.12328 0.3553404
## 2   2 - 2 0.66698 273.1762 280.0127 0.56106  0.65586 -0.04931 0.5471083
## 1   1 - 2 0.67483 281.6547 287.2595 0.53295  1.04821 -0.05377 0.8579014
```

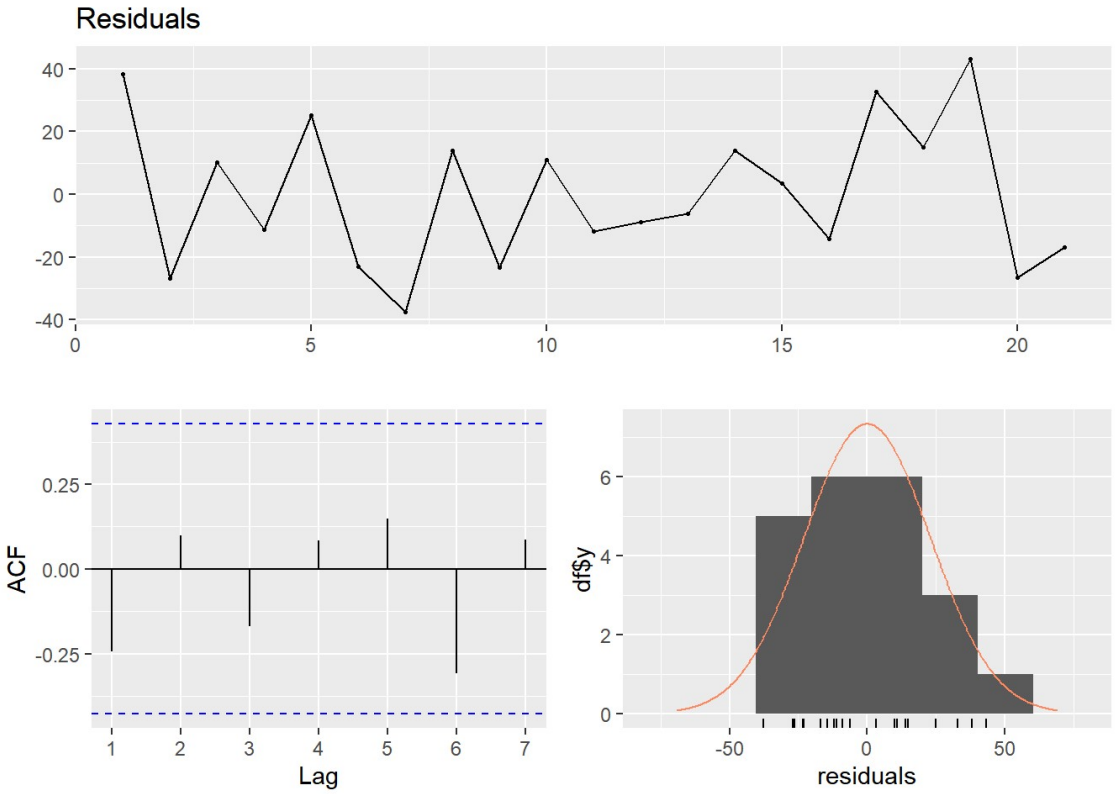
```
model.poly4 = polyDlm(x = as.vector(relhumid), y = as.vector(ffd), q = 10, k = 2, show.beta = TRUE)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value P(>|t|)
## beta.0   -8.8300       5.30 -1.6600  0.1240
## beta.1   -7.1000       3.82 -1.8600  0.0898
## beta.2   -5.5600       3.13 -1.7800  0.1030
## beta.3   -4.2000       3.07 -1.3700  0.1990
## beta.4   -3.0300       3.24 -0.9330  0.3710
## beta.5   -2.0400       3.35 -0.6080  0.5560
## beta.6   -1.2300       3.32 -0.3720  0.7170
## beta.7   -0.6160       3.25 -0.1900  0.8530
## beta.8   -0.1830       3.43 -0.0534  0.9580
## beta.9    0.0648       4.21  0.0154  0.9880
## beta.10   0.1270       5.74  0.0222  0.9830
```

```
summary(model.poly4)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -37.56 -16.91  -6.12  13.94  43.19
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3290.68298 2900.02326   1.135    0.272
## z.t0          -8.83169    5.30493  -1.665    0.114
## z.t1           1.82201    2.32542   0.784    0.444
## z.t2          -0.09261    0.22708  -0.408    0.688
##
## Residual standard error: 25.03 on 17 degrees of freedom
## Multiple R-squared:  0.1878, Adjusted R-squared:  0.04447
## F-statistic:  1.31 on 3 and 17 DF,  p-value: 0.3036
```

```
checkresiduals(model.poly4$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 7
##
## data:  Residuals
## LM test = 5.8819, df = 7, p-value = 0.5536
```

```
MASE(model.poly4)
```

```
##                 MASE
## model.poly4 0.652098
```

Polynomial Model Summary

Model	MASE	Model p-value	Model R-squared value	Breusch-Godfrey p-value

Model	MASE	Model p-value	Model R-squared value	Breusch-Godfrey p-value
Temperature	0.646	0.3896	0.009816	0.507
Rain	0.625	0.3729	0.01605	0.03878
Radiation	0.646	0.773	-0.1037	0.5379
Relative Humidity	0.652	0.3036	0.04447	0.5536

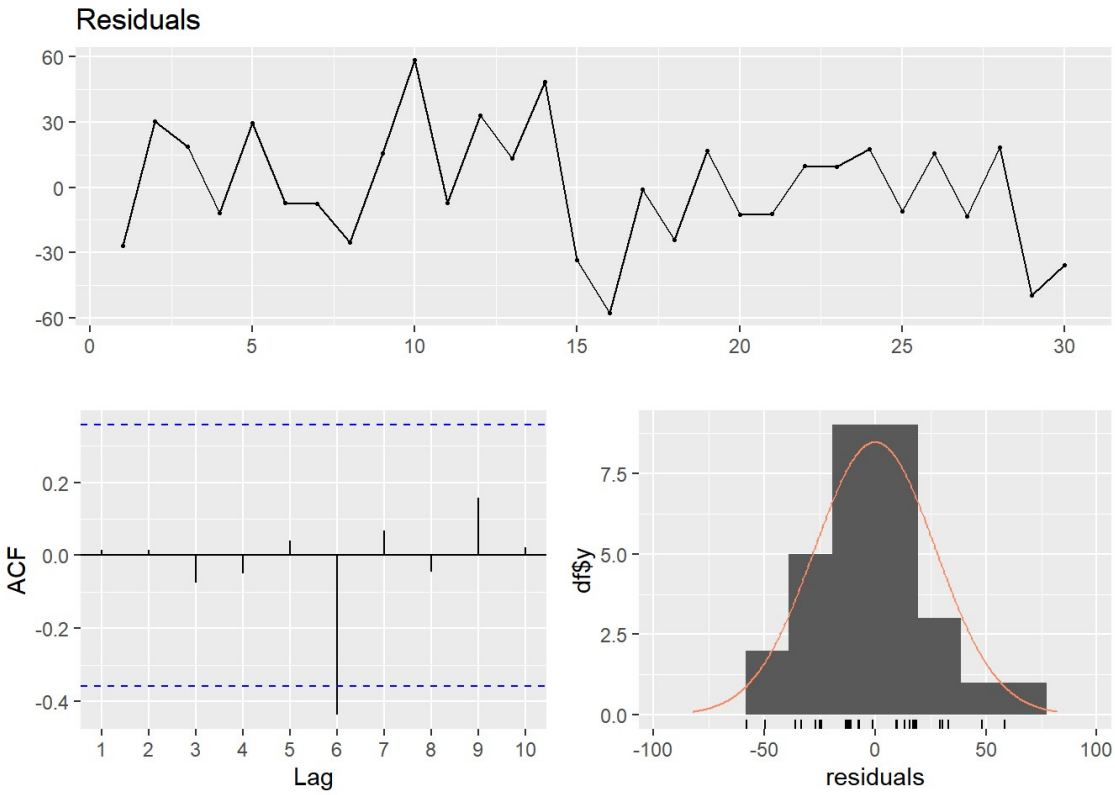
All the models are not significant, with p-values larger than 0.05 and very low R-squared values, implying that none of the models explained the data particularly well. All the models except for the rain model failed to reject the null hypothesis for the Breusch-Godfrey test due to high p-values over 0.05. This implies that the models do not have serial correlation. The MASE values decent since all the models produce relatively low values compared to earlier models. Despite poor performance by all the polynomial models, relative humidity is selected to be the best predictor from the polynomial models based on the results obtained.

Koyck (Geometric) DLM

```
# Kocyk
model.Koyck1 = koyckDlm(x = as.vector(temp), y = as.vector(ffd), intercept = TRUE)
summary(model.Koyck1)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -57.755 -12.972  -4.079  17.329  58.541
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -19.87216  608.22407  -0.033    0.974
## Y.1           0.03846    0.25384   0.152    0.881
## X.t          23.22042   61.08917   0.380    0.707
##
## Residual standard error: 28.38 on 27 degrees of freedom
## Multiple R-Squared: -0.3272, Adjusted R-squared: -0.4255
## Wald test: 0.07269 on 2 and 27 DF,  p-value: 0.9301
##
## Diagnostic tests:
## NULL
##
##                             alpha     beta       phi
## Geometric coefficients:  -20.66698 23.22042 0.0384583
```

```
checkresiduals(model.Koyck1$model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 8.0261, df = 6, p-value = 0.2362
##
## Model df: 0.   Total lags used: 6
```

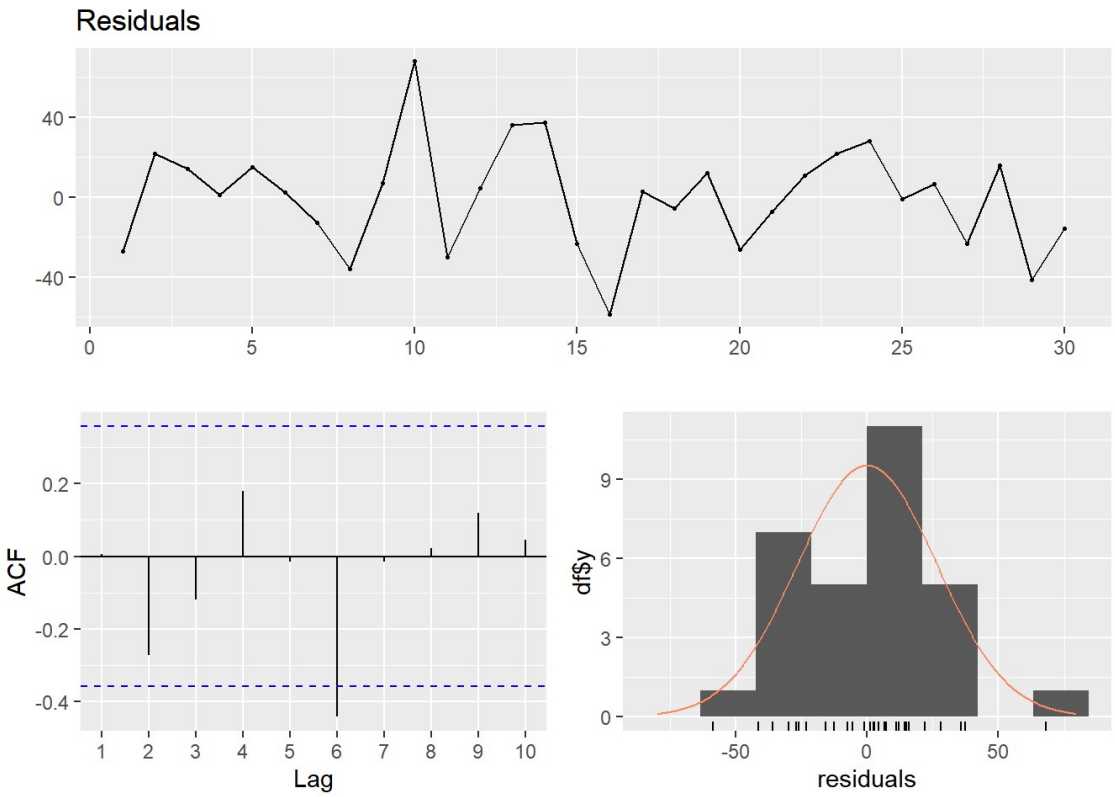
```
MASE(model.Koyck1)
```

```
##                   MASE
## model.Koyck1 0.7927338
```

```
model.Koyck2 = koyckDlm(x = as.vector(rain), y = as.vector(ffd), intercept = TRUE)
summary(model.Koyck2)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -58.691 -21.222   2.697  14.856  68.192
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.266e+02  2.185e+02   0.579    0.567
## Y.1         4.591e-03  2.196e-01   0.021    0.983
## X.t         3.448e+01  8.772e+01   0.393    0.697
##
## Residual standard error: 27.55 on 27 degrees of freedom
## Multiple R-Squared: -0.2505, Adjusted R-squared: -0.3431
## Wald test: 0.07773 on 2 and 27 DF,  p-value: 0.9254
##
## Diagnostic tests:
## NULL
##
##                              alpha     beta         phi
## Geometric coefficients:  127.2254 34.48095 0.004590669
```

```
checkresiduals(model.Koyck2$model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 12.111, df = 6, p-value = 0.05953
##
## Model df: 0.   Total lags used: 6
```

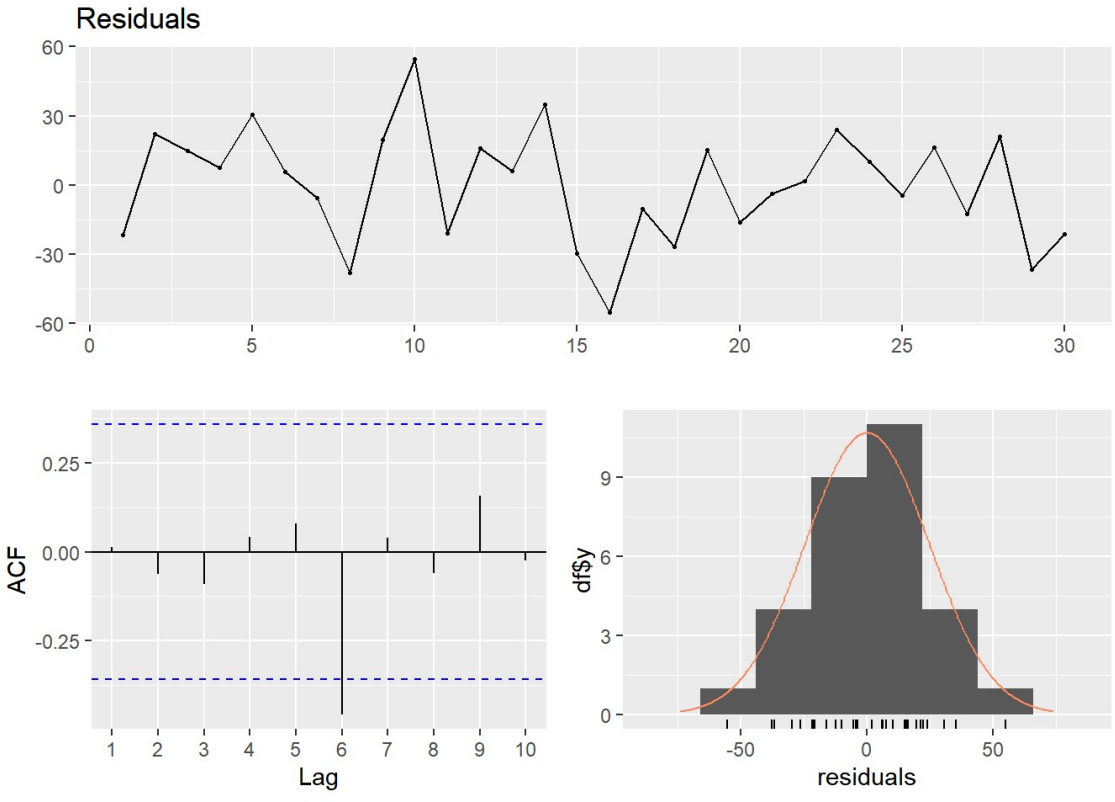
```
MASE(model.Koyck2)
```

```
##                   MASE
## model.Koyck2 0.7267872
```

```
model.Koyck3 = koyckDlm(x = as.vector(radiation), y = as.vector(ffd), intercept = TRUE)
summary(model.Koyck3)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -55.229 -19.662   3.956  16.232  54.756
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 418.31843  384.12678   1.089    0.286
## Y.1          -0.03254    0.20729  -0.157    0.876
## X.t         -13.87153   25.49529  -0.544    0.591
##
## Residual standard error: 25.54 on 27 degrees of freedom
## Multiple R-Squared: -0.07436,    Adjusted R-squared: -0.1539
## Wald test: 0.1486 on 2 and 27 DF,  p-value: 0.8626
##
## Diagnostic tests:
## NULL
##
##                              alpha      beta         phi
## Geometric coefficients:  405.1353 -13.87153 -0.03254005
```

```
checkresiduals(model.Koyck3$model)
```



```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 9.1322, df = 6, p-value = 0.1663
##
## Model df: 0.   Total lags used: 6
```

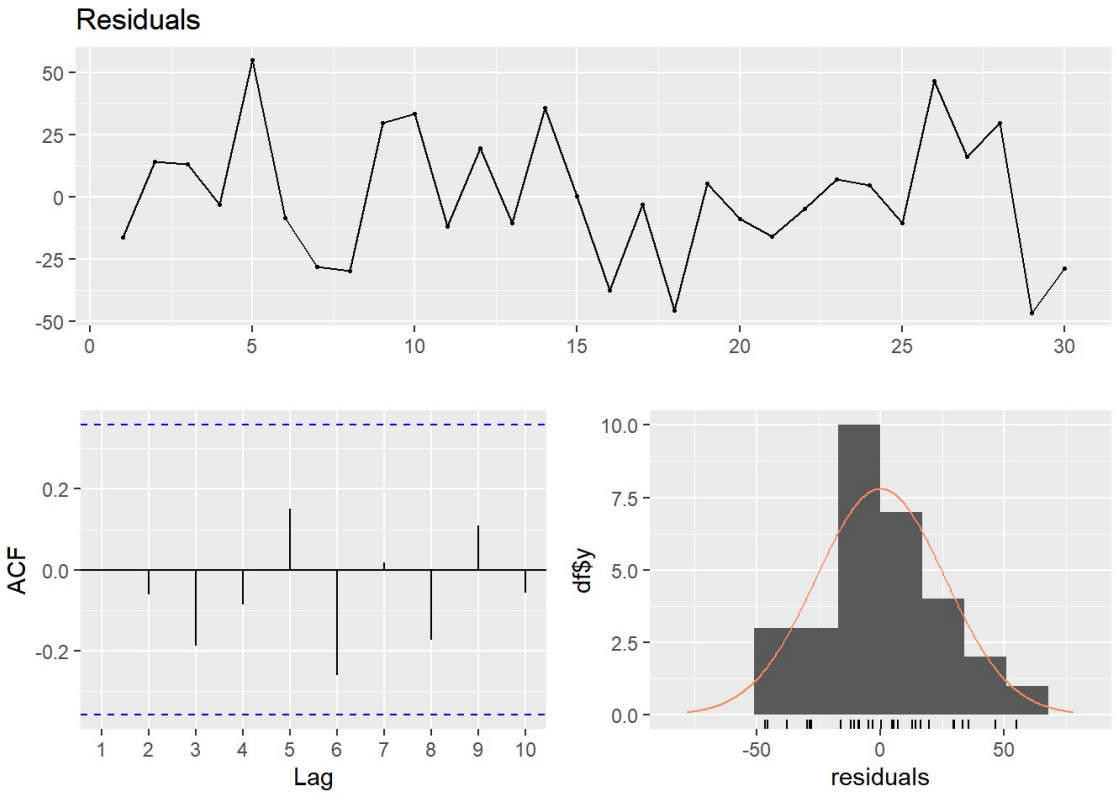
```
MASE(model.Koyck3)
```

```
##                   MASE
## model.Koyck3 0.7136301
```

```
model.Koyck4 = koyckDlm(x = as.vector(relhumid), y = as.vector(ffd), intercept = TRUE)
summary(model.Koyck4)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.787 -14.896  -3.024  15.673  55.019
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1898.18932 4211.29973   0.451    0.656
## Y.1           -0.05904    0.25016  -0.236    0.815
## X.t          -17.72952   44.24103  -0.401    0.692
##
## Residual standard error: 27 on 27 degrees of freedom
## Multiple R-Squared: -0.2016, Adjusted R-squared: -0.2906
## Wald test: 0.0808 on 2 and 27 DF,  p-value: 0.9226
##
## Diagnostic tests:
## NULL
##
##                             alpha      beta         phi
## Geometric coefficients:  1792.364 -17.72952 -0.05904222
```

```
checkresiduals(model.Koyck4$model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 5.2406, df = 6, p-value = 0.5133
##
## Model df: 0.   Total lags used: 6
```

```
MASE(model.Koyck4)
```

```
##                   MASE
## model.Koyck4 0.7335956
```

Koyck Model Summary

Model	MASE	Model p-value	Model R-squared value	Ljung-Box p-value
Temperature	0.793	0.9301	-0.4255	0.2362
Rain	0.727	0.9254	-0.3431	0.05953
Radiation	0.714	0.8626	-0.1037	0.1663
Relative Humidity	0.734	0.9226	-0.2906	0.5133

The Koyck models are also not significant due to high p-values obtained for each model. The R-squared values obtained for the models are not desirable due to all of them being negative. MASE values are slightly higher than those obtained for the polynomial models. The best model from the Koyck models is chosen to be radiation based on the results obtained, however none of the models are particularly suitable to predict FFD.

Finite DLM

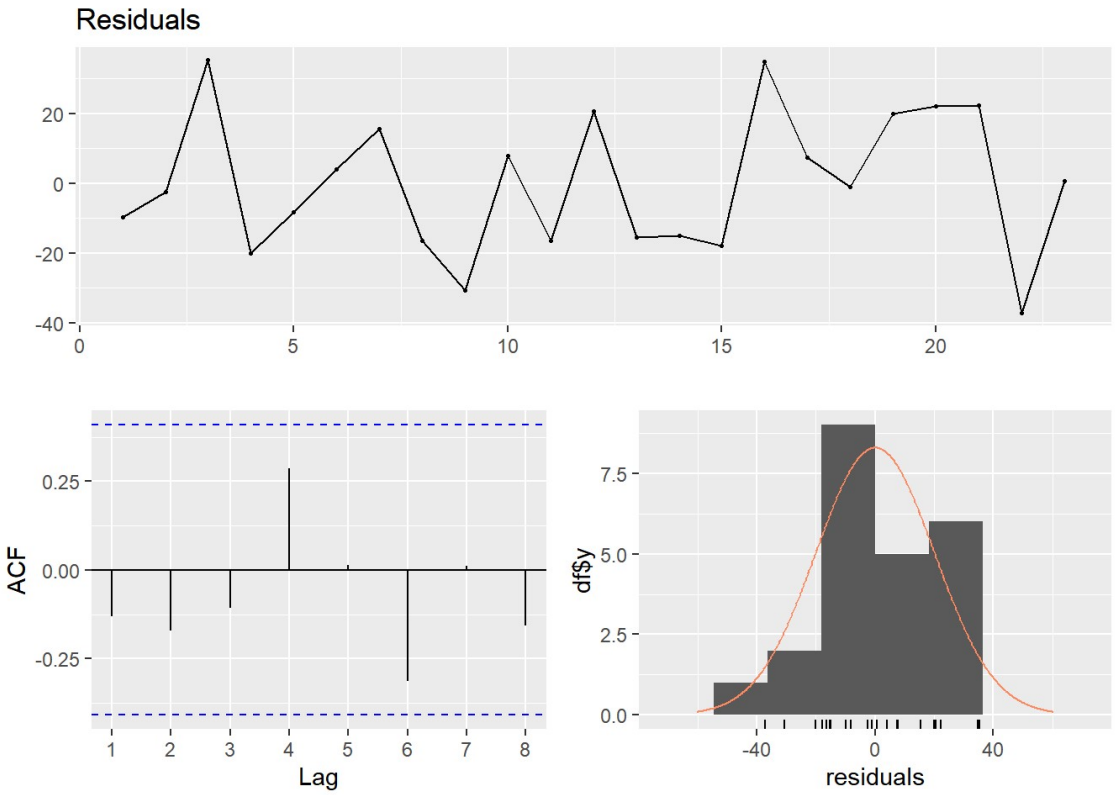
```
# Finite DLM
finiteDLMauto(x = as.vector(temp), y = as.vector(ffd), q.min = 1, q.max = 8,
              model.type = "dlm", error.type = "AIC", trace = TRUE)
```

```
##   q - k    MASE      AIC      BIC   GMRAE    MBRAE R.Adj.Sq Ljung-Box
## 8     8 0.53776 224.2942 236.7846 0.48954 -0.15363 -0.05959 0.5048786
## 7     7 0.55255 231.1518 242.9323 0.46908  0.33150 -0.00430 0.5149090
## 6     6 0.57949 238.7187 249.6886 0.58989  0.76443  0.00903 0.4142493
## 5     5 0.59596 247.1156 257.1804 0.56488  0.44083  0.03688 0.3059082
## 4     4 0.64069 258.0491 267.1200 0.52394 -0.01370 -0.09108 0.8481071
## 3     3 0.64396 264.2921 272.2853 0.50459  0.47305 -0.03114 0.9350955
## 2     2 0.67881 272.7512 279.5877 0.61846 -1.47353 -0.03404 0.9864821
## 1     1 0.66448 279.6699 285.2747 0.60244 -0.73174  0.01369 0.9600714
```

```
model.dlm1 = dlm(x = as.vector(temp), y = as.vector(ffd), q = 8)
summary(model.dlm1)
```

```
##
## Call:
## lm(formula = model.formula, data = design)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -37.275 -15.864  -1.115  17.803  35.433
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -234.5776   436.6363  -0.537    0.600
## x.t          -16.2546    16.9552  -0.959    0.355
## x.1           13.2675    19.2454   0.689    0.503
## x.2            6.9760    20.7470   0.336    0.742
## x.3            4.2206    21.1738   0.199    0.845
## x.4            0.3059    19.9592   0.015    0.988
## x.5           33.1797    19.4750   1.704    0.112
## x.6            9.0036    18.8281   0.478    0.640
## x.7          -17.7065    17.7672  -0.997    0.337
## x.8           13.9289    18.7092   0.744    0.470
##
## Residual standard error: 26.15 on 13 degrees of freedom
## Multiple R-squared:  0.3739, Adjusted R-squared:  -0.05959
## F-statistic: 0.8625 on 9 and 13 DF,  p-value: 0.578
##
## AIC and BIC values for the model:
##        AIC      BIC
## 1 224.2942 236.7846
```

```
checkresiduals(model.dlm1$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 13
##
## data:  Residuals
## LM test = 23, df = 13, p-value = 0.04168
```

```
MASE(model.dlm1)
```

```
##                MASE
## model.dlm1 0.5377635
```

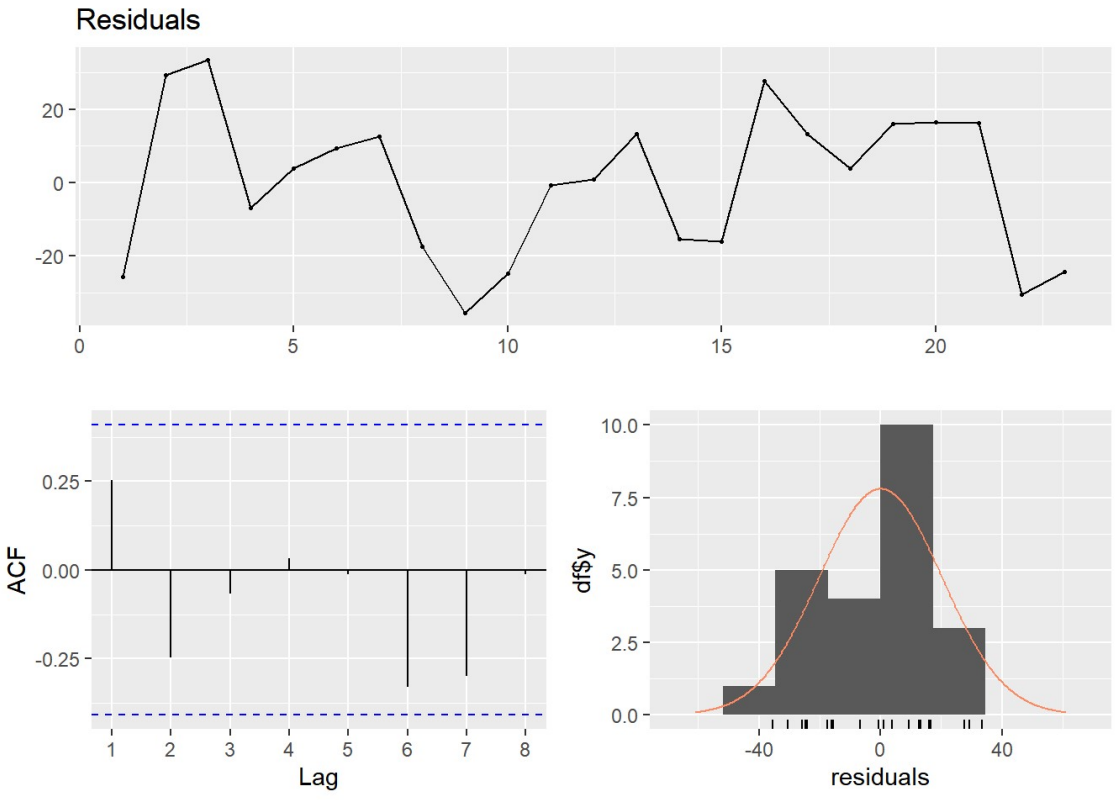
```
finiteDLMauto(x = as.vector(rain), y = as.vector(ffd), q.min = 1, q.max = 8,
              model.type = "dlm", error.type = "AIC", trace = TRUE)
```

```
##   q - k    MASE      AIC      BIC   GMRAE    MBRAE R.Adj.Sq Ljung-Box
## 8     8 0.55539 224.8048 237.2953 0.49990  0.33082 -0.08337 0.1968049
## 7     7 0.54478 231.4148 243.1954 0.49633  0.33181 -0.01537 0.1328802
## 6     6 0.55221 238.0915 249.0614 0.54542  0.29252  0.03358 0.2272468
## 5     5 0.58089 246.1244 256.1891 0.50429  0.45681  0.07290 0.3002973
## 4     4 0.63093 257.0860 266.1568 0.62526  0.34033 -0.05285 0.4414820
## 3     3 0.63266 265.1863 273.1795 0.49927  0.43813 -0.06460 0.9369154
## 2     2 0.67527 273.8231 280.6596 0.54561 -1.34330 -0.07297 0.9002345
## 1     1 0.67589 281.9769 287.5817 0.63528  4.41731 -0.06515 0.9475115
```

```
model.dlm2 = dlm(x = as.vector(rain), y = as.vector(ffd), q = 8)
summary(model.dlm2)
```

```
##
## Call:
## lm(formula = model.formula, data = design)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -35.580 -16.756   3.891  14.729  33.528
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  146.777     86.001   1.707   0.1116
## x.t           -5.535     14.077  -0.393   0.7005
## x.1           12.160     14.258   0.853   0.4092
## x.2           -1.927     14.492  -0.133   0.8962
## x.3           24.412     14.494   1.684   0.1160
## x.4          -24.788     15.346  -1.615   0.1302
## x.5           29.608     15.796   1.874   0.0835 .
## x.6           -9.355     14.926  -0.627   0.5417
## x.7           11.641     15.976   0.729   0.4791
## x.8          -10.249     16.204  -0.632   0.5380
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.45 on 13 degrees of freedom
## Multiple R-squared:  0.3598, Adjusted R-squared:  -0.08337
## F-statistic: 0.8119 on 9 and 13 DF,  p-value: 0.6149
##
## AIC and BIC values for the model:
##        AIC      BIC
## 1 224.8048 237.2953
```

```
checkresiduals(model.dlm2$model)
```



```
##
##  Breusch-Godfrey test for serial correlation of order up to 13
##
## data:  Residuals
## LM test = 23, df = 13, p-value = 0.04168
```

```
MASE(model.dlm2)
```

```
##                MASE
## model.dlm2 0.5553913
```

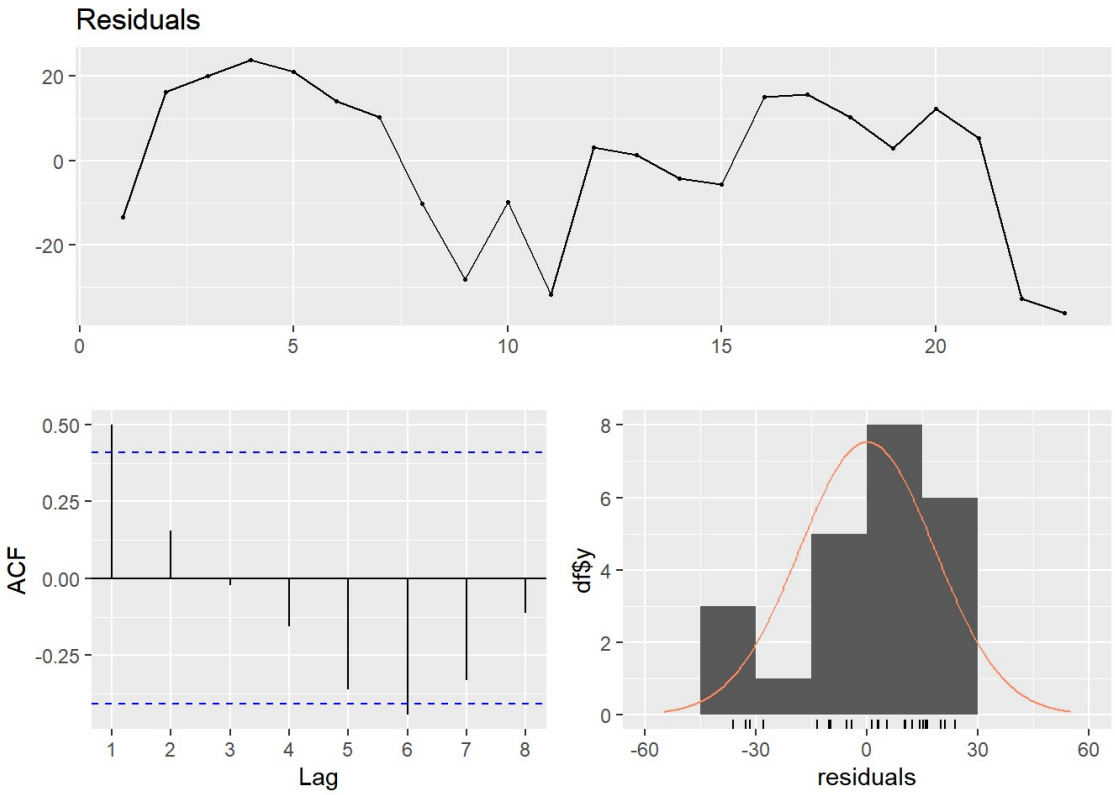
```
finiteDLMauto(x = as.vector(radiation), y = as.vector(ffd), q.min = 1, q.max = 8,
              model.type = "dlm", error.type = "AIC", trace = TRUE)
```

```
##   q - k    MASE      AIC      BIC   GMRAE    MBRAE R.Adj.Sq  Ljung-Box
## 8     8 0.48398 219.9335 232.4240 0.46750 -0.02805  0.12341 0.01076661
## 7     7 0.47918 225.7050 237.4856 0.44950 -0.02535  0.19961 0.01196956
## 6     6 0.52987 235.4683 246.4382 0.51395 -1.29322  0.12984 0.03143819
## 5     5 0.58709 249.3748 259.4396 0.48849  0.40993 -0.05056 0.47295184
## 4     4 0.59688 257.9491 267.0200 0.47646  0.65680 -0.08705 0.98795490
## 3     3 0.61648 264.6660 272.6592 0.46684  0.65345 -0.04500 0.88857402
## 2     2 0.66726 273.2458 280.0823 0.51313 -1.85015 -0.05182 0.71483368
## 1     1 0.65796 281.5710 287.1758 0.56645  1.41699 -0.05083 0.96811432
```

```
model.dlm3 = dlm(x = as.vector(radiation), y = as.vector(ffd), q = 8)
summary(model.dlm3)
```

```
##
## Call:
## lm(formula = model.formula, data = design)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -36.146 -10.029   3.224  14.631  23.891
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  146.258    380.404   0.384   0.7068
## x.t           13.348     14.496   0.921   0.3739
## x.1           -7.485     17.381  -0.431   0.6738
## x.2          -34.521     17.082  -2.021   0.0644 .
## x.3           30.360     17.407   1.744   0.1047
## x.4          -12.838     17.683  -0.726   0.4807
## x.5           40.876     18.641   2.193   0.0471 *
## x.6          -47.718     18.887  -2.527   0.0253 *
## x.7           20.825     18.818   1.107   0.2885
## x.8            1.233     17.323   0.071   0.9443
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23.79 on 13 degrees of freedom
## Multiple R-squared:  0.482,  Adjusted R-squared:  0.1234
## F-statistic: 1.344 on 9 and 13 DF,  p-value: 0.3042
##
## AIC and BIC values for the model:
##        AIC     BIC
## 1 219.9335 232.424
```

```
checkresiduals(model.dlm3$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 13
##
## data:  Residuals
## LM test = 23, df = 13, p-value = 0.04168
```

```
MASE(model.dlm3)
```

```
##                MASE
## model.dlm3 0.4839788
```

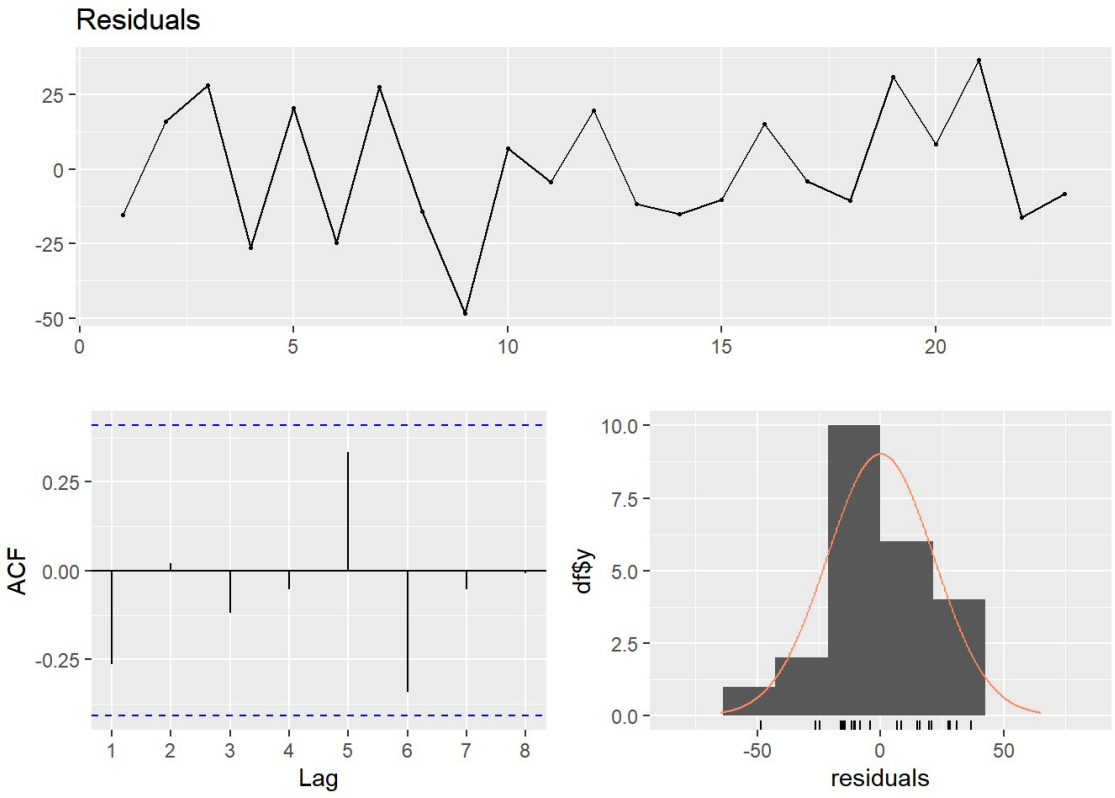
```
finiteDLMauto(x = as.vector(relhumid), y = as.vector(ffd), q.min = 1, q.max = 8,
              model.type = "dlm", error.type = "AIC", trace = TRUE)
```

```
##   q - k    MASE      AIC      BIC   GMRAE   MBRAE R.Adj.Sq Ljung-Box
## 8     8 0.59092 227.6823 240.1727 0.63485 0.35459 -0.22775 0.1790283
## 7     7 0.59196 234.5121 246.2927 0.58807 0.37936 -0.15523 0.3036238
## 6     6 0.59068 240.4575 251.4274 0.64573 0.38918 -0.06235 0.2955561
## 5     5 0.62473 248.6573 258.7220 0.64421 0.32410 -0.02196 0.3638097
## 4     4 0.61331 254.6378 263.7087 0.65999 0.33371  0.03842 0.3902598
## 3     3 0.61247 260.6544 268.6476 0.64605 0.20942  0.09449 0.3833163
## 2     2 0.66698 273.1762 280.0127 0.56106 0.65586 -0.04931 0.5471083
## 1     1 0.67483 281.6547 287.2595 0.53295 1.04821 -0.05377 0.8579014
```

```
model.dlm4 = dlm(x = as.vector(relhumid), y = as.vector(ffd), q = 8)
summary(model.dlm4)
```

```
##
## Call:
## lm(formula = model.formula, data = design)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -48.474 -14.718  -4.257  17.753  36.621
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2860.5403  2629.9855   1.088    0.296
## x.t           -6.4456    10.2036  -0.632    0.539
## x.1           -6.4440    10.5714  -0.610    0.553
## x.2           -1.7943    10.7161  -0.167    0.870
## x.3          -12.6097     9.7133  -1.298    0.217
## x.4           -2.9260     8.4179  -0.348    0.734
## x.5            0.5875     9.0176   0.065    0.949
## x.6           -2.3194     9.0132  -0.257    0.801
## x.7            0.5700     8.9481   0.064    0.950
## x.8            3.3369     9.2198   0.362    0.723
##
## Residual standard error: 28.15 on 13 degrees of freedom
## Multiple R-squared:  0.2745, Adjusted R-squared:  -0.2278
## F-statistic: 0.5465 on 9 and 13 DF,  p-value: 0.8164
##
## AIC and BIC values for the model:
##        AIC      BIC
## 1 227.6822 240.1727
```

```
checkresiduals(model.dlm4$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 13
##
## data:  Residuals
## LM test = 23, df = 13, p-value = 0.04168
```

```
MASE(model.dlm4)
```

```
##                 MASE
## model.dlm4 0.5909197
```

Finite DLM Model Summary

Model	MASE	Model p-value	Model R-squared value	Breusch-Godfrey p-value
Temperature	0.538	0.578	-0.05959	0.04168
Rain	0.555	0.6149	-0.08337	0.04168
Radiation	0.484	0.3042	0.04168	0.1234
Relative Humidity	0.591	0.8164	-0.2278	0.04168

A finite DLM is fitted using the dlm() function. Prior to fitting, the ideal lag value is found using the finiteDLMauto() function, which is made to generate multiple models with q ranging from 1 to 8 and uses AIC error to determine the suitability of each model. The results of finiteDLMauto() suggest that the ideal value for q is 8.

The results for the DLM models are similar to the polynomial and Koyck implementations as all the models have high p-values, indicating that the models are not significant. Low R-squared values indicate that the models do not describe the data particularly well, but the p-value obtained from the Breusch-Godfrey test implies that the models do not have serial correlation and they have the lowest MASE obtained thus far. Based on these results, radiation is chosen as the best predictor when implementing DLM models for FFD.

ARDL Model

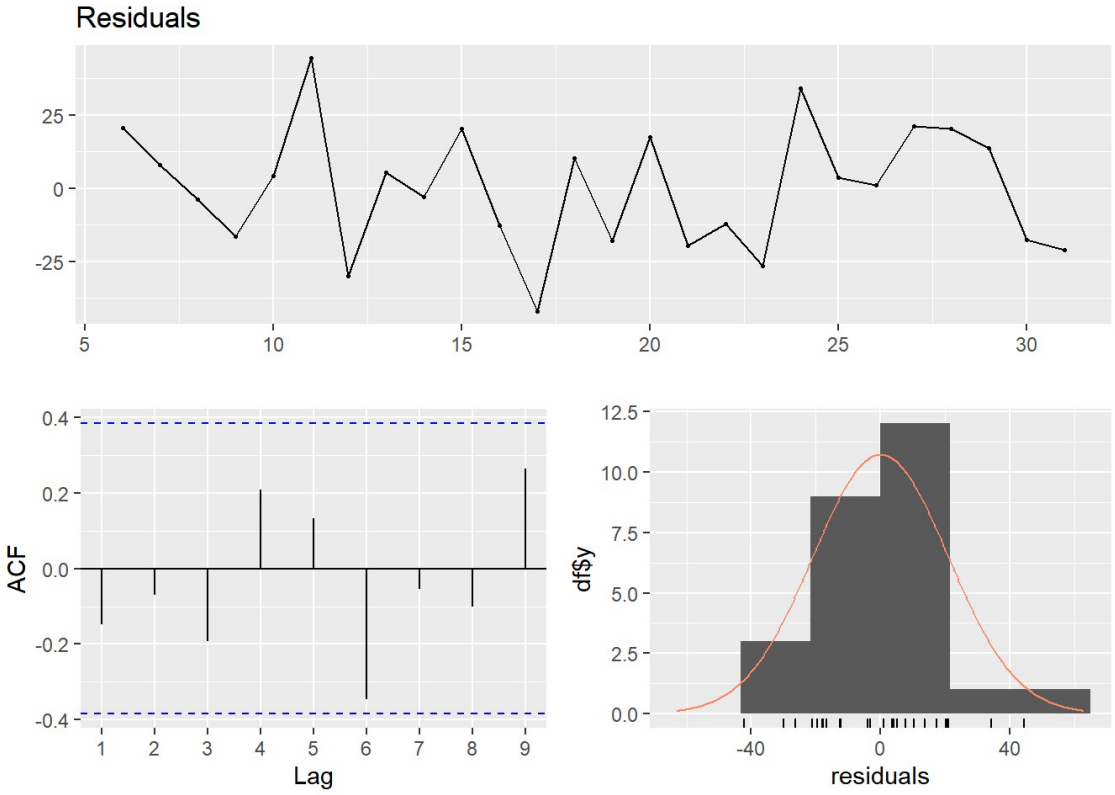
```
#ARDL

#temp
columns = c("p","q","AIC","BIC")
df = data.frame(matrix(nrow = 0, ncol = length(columns)))
colnames(df) = columns

for(i in 1:5){
  for(j in 1:5){
    model.ARDL = ardlDlm(x = as.vector(temp), y = as.vector(ffd), p = i, q = j)
    new_row = data.frame(p = i, q=j, AIC=AIC(model.ARDL$model), BIC=BIC(model.ARDL$model))
    df = bind_rows(df, new_row)
  }
}

df[order(df$AIC),]
```

```
##    p q      AIC      BIC
## 21 5 1 249.0170 260.3399
## 22 5 2 250.5602 263.1412
## 23 5 3 252.4929 266.3320
## 5  1 5 253.8875 265.2103
## 24 5 4 254.2659 269.3631
## 25 5 5 254.7620 271.1173
## 10 2 5 255.6945 268.2754
## 15 3 5 256.2379 270.0770
## 20 4 5 258.1505 273.2476
## 16 4 1 260.0445 270.4112
## 4  1 4 260.7609 271.1276
## 17 4 2 262.0215 273.6840
## 9  2 4 262.7310 274.3935
## 14 3 4 263.0578 276.0161
## 18 4 3 264.0057 276.9641
## 19 4 4 264.9661 279.2203
## 11 3 1 266.2915 275.6169
## 3  1 3 267.8377 277.1631
## 12 3 2 268.2817 278.9393
## 8  2 3 269.7687 280.4263
## 13 3 3 270.2780 282.2678
## 2  1 2 274.6571 282.8609
## 6  2 1 274.7477 282.9515
## 7  2 2 276.5497 286.1208
## 1  1 1 281.6674 288.6734
```

```
model.ARDL1 <- ardlDlm(x = as.vector(temp), y = as.vector(ffd), p = 5, q = 1)
summary(model.ARDL1)
```

```
##
## Time series regression with "ts" data:
## Start = 6, End = 31
##
## Call:
## dynlm(formula = as.formula(model.text), data = data, start = 1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -42.158 -17.389   2.224  16.392  44.444
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -109.00430  268.70831  -0.406   0.6898
## X.t          -17.03324   16.09102  -1.059   0.3038
## X.1           14.07942   15.46983   0.910   0.3748
## X.2            2.08246   16.60969   0.125   0.9016
## X.3            4.83937   15.69984   0.308   0.7614
## X.4            2.11481   15.65509   0.135   0.8940
## X.5           28.86539   15.76145   1.831   0.0836 .
## Y.1           -0.05743    0.21966  -0.261   0.7967
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.72 on 18 degrees of freedom
## Multiple R-squared:  0.2708, Adjusted R-squared:  -0.01279
## F-statistic: 0.9549 on 7 and 18 DF,  p-value: 0.4912
```

```
checkresiduals(model.ARDL1$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 11
##
## data:  Residuals
## LM test = 13.511, df = 11, p-value = 0.2612
```

```
MASE(model.ARDL1)
```

```
##                  MASE
## model.ARDL1 0.5906457
```

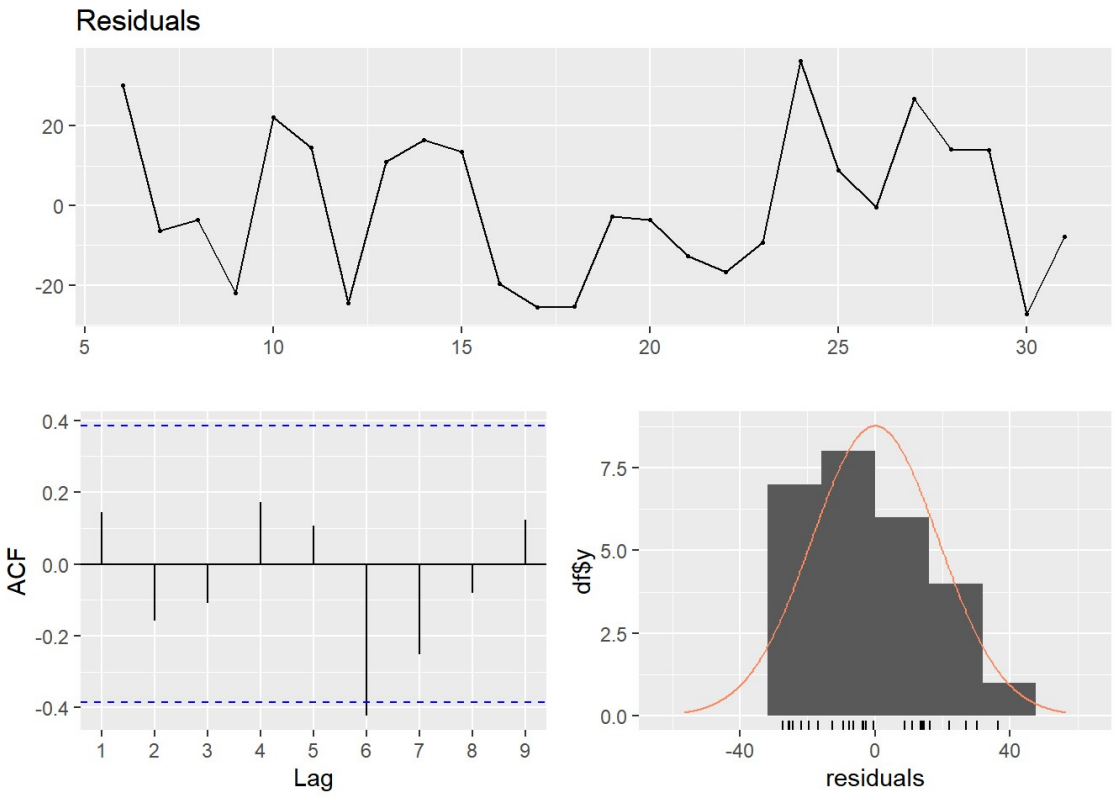
```
df = df[0,]

#rain
for(i in 1:5){
  for(j in 1:5){
    model.ARDL = ardlDlm(x = as.vector(rain), y = as.vector(ffd), p = i, q = j)
    new_row = data.frame(p = i, q=j, AIC=AIC(model.ARDL$model), BIC=BIC(model.ARDL$model))
    df = bind_rows(df, new_row)
  }
}

df[order(df$AIC),]
```

```
##    p q      AIC      BIC
## 23 5 3 247.4178 261.2568
## 21 5 1 247.6247 258.9476
## 24 5 4 248.4355 263.5326
## 22 5 2 249.3037 261.8846
## 25 5 5 250.0315 266.3868
## 5  1 5 255.5705 266.8934
## 10 2 5 257.2016 269.7826
## 20 4 5 257.7752 272.8724
## 15 3 5 258.2483 272.0874
## 16 4 1 259.0485 269.4152
## 17 4 2 260.6860 272.3485
## 18 4 3 262.1327 275.0910
## 4  1 4 262.6875 273.0542
## 19 4 4 263.7563 278.0105
## 9  2 4 264.2889 275.9514
## 14 3 4 264.4739 277.4323
## 11 3 1 267.1614 276.4868
## 12 3 2 269.1262 279.7838
## 3  1 3 269.3361 278.6615
## 8  2 3 270.6698 281.3274
## 13 3 3 270.8152 282.8051
## 6  2 1 275.8163 284.0201
## 2  1 2 276.7348 284.9385
## 7  2 2 277.8084 287.3795
## 1  1 1 283.9744 290.9804
```

```
model.ARDL2 <- ardlDlm(x = as.vector(rain), y = as.vector(ffd), p = 5, q = 3)
summary(model.ARDL2)
```

```
##
## 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
## -27.328 -15.753  -3.097  14.021  36.333
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 151.2305    76.3341   1.981   0.0650 .
## X.t           1.3743    14.8163   0.093   0.9273
## X.1          22.4130    15.2765   1.467   0.1617
## X.2          -6.5116    13.2735  -0.491   0.6304
## X.3          25.3618    12.8441   1.975   0.0658 .
## X.4         -21.2375    14.8494  -1.430   0.1719
## X.5          40.6213    16.0795   2.526   0.0225 *
## Y.1           0.1708     0.2173   0.786   0.4433
## Y.2          -0.1627     0.2959  -0.550   0.5900
## Y.3          -0.4236     0.2638  -1.606   0.1278
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23.54 on 16 degrees of freedom
## Multiple R-squared:  0.4121, Adjusted R-squared:  0.08137
## F-statistic: 1.246 on 9 and 16 DF,  p-value: 0.3355
```

```
checkresiduals(model.ARDL2$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 13
##
## data:  Residuals
## LM test = 22.571, df = 13, p-value = 0.04712
```

```
MASE(model.ARDL2)
```

```
##               MASE
## model.ARDL2 0.5484
```

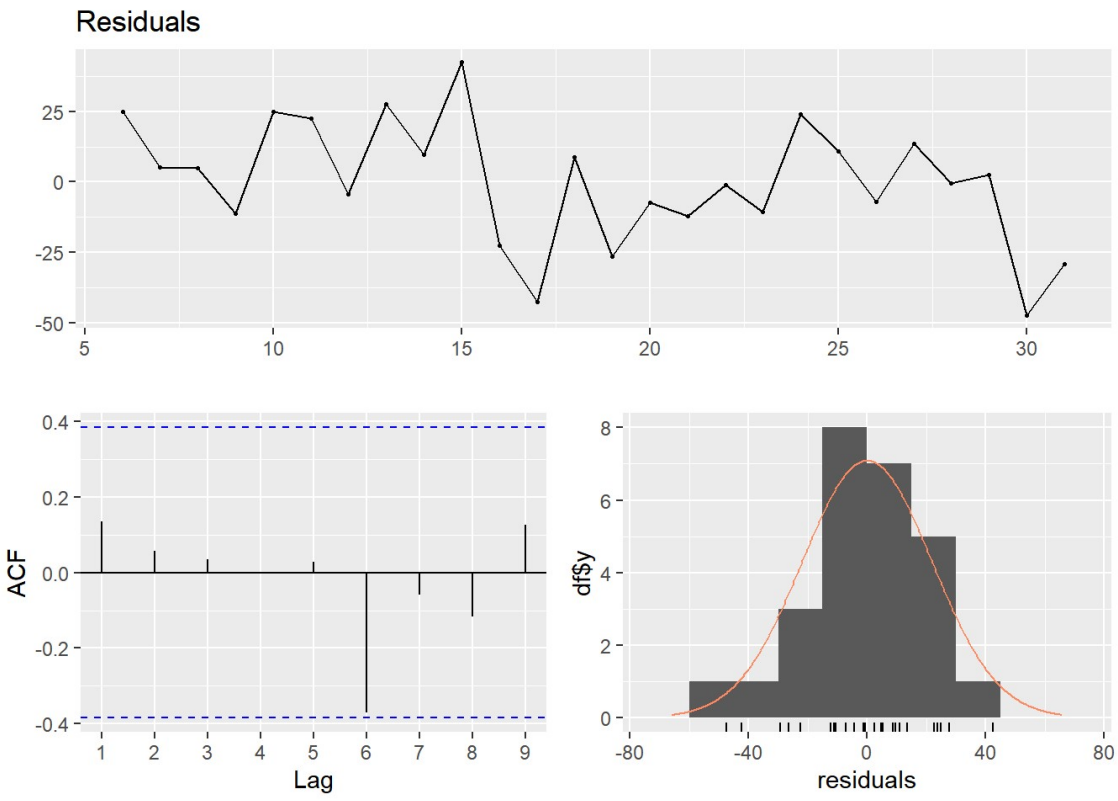
```
df = df[0,]

#radiation
for(i in 1:5){
  for(j in 1:5){
    model.ARDL = ardlDlm(x = as.vector(radiation), y = as.vector(ffd), p = i, q = j)
    new_row = data.frame(p = i, q=j, AIC=AIC(model.ARDL$model), BIC=BIC(model.ARDL$model))
    df = bind_rows(df, new_row)
  }
}

df[order(df$AIC),]
```

```
##    p q      AIC      BIC
## 21 5 1 251.3746 262.6975
## 22 5 2 252.9746 265.5556
## 23 5 3 254.7000 268.5391
## 5  1 5 255.6012 266.9241
## 24 5 4 255.7433 270.8405
## 10 2 5 256.4326 269.0136
## 15 3 5 257.3509 271.1899
## 25 5 5 257.5483 273.9036
## 20 4 5 258.7707 273.8679
## 16 4 1 259.9163 270.2830
## 17 4 2 261.3427 273.0052
## 4  1 4 262.6904 273.0571
## 9  2 4 262.9001 274.5626
## 18 4 3 263.3127 276.2711
## 14 3 4 263.6351 276.5935
## 19 4 4 265.2261 279.4803
## 11 3 1 266.5357 275.8611
## 12 3 2 268.3699 279.0276
## 3  1 3 269.1591 278.4845
## 8  2 3 269.5463 280.2040
## 13 3 3 270.1803 282.1701
## 6  2 1 275.2400 283.4438
## 2  1 2 276.5262 284.7300
## 7  2 2 277.1882 286.7593
## 1  1 1 283.5658 290.5718
```

```
model.ARDL3 <- ardlDlm(x = as.vector(radiation), y = as.vector(ffd), p = 5, q = 1)
summary(model.ARDL3)
```

```
##
## 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
## -47.412 -11.010   0.968  12.883  42.547
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  28.842428 324.397851   0.089    0.930
## X.t           7.276129  15.162165   0.480    0.637
## X.1           4.276663  17.673151   0.242    0.812
## X.2         -28.391845  17.135606  -1.657    0.115
## X.3          11.308921  17.279661   0.654    0.521
## X.4          -3.404791  18.014276  -0.189    0.852
## X.5          21.226029  15.784898   1.345    0.195
## Y.1          -0.002524   0.228188  -0.011    0.991
##
## Residual standard error: 25.87 on 18 degrees of freedom
## Multiple R-squared:  0.2016, Adjusted R-squared:  -0.1089
## F-statistic: 0.6492 on 7 and 18 DF,  p-value: 0.7106
```

```
checkresiduals(model.ARDL3$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 11
##
## data:  Residuals
## LM test = 16.745, df = 11, p-value = 0.1157
```

```
MASE(model.ARDL3)
```

```
##                MASE
## model.ARDL3 0.587024
```

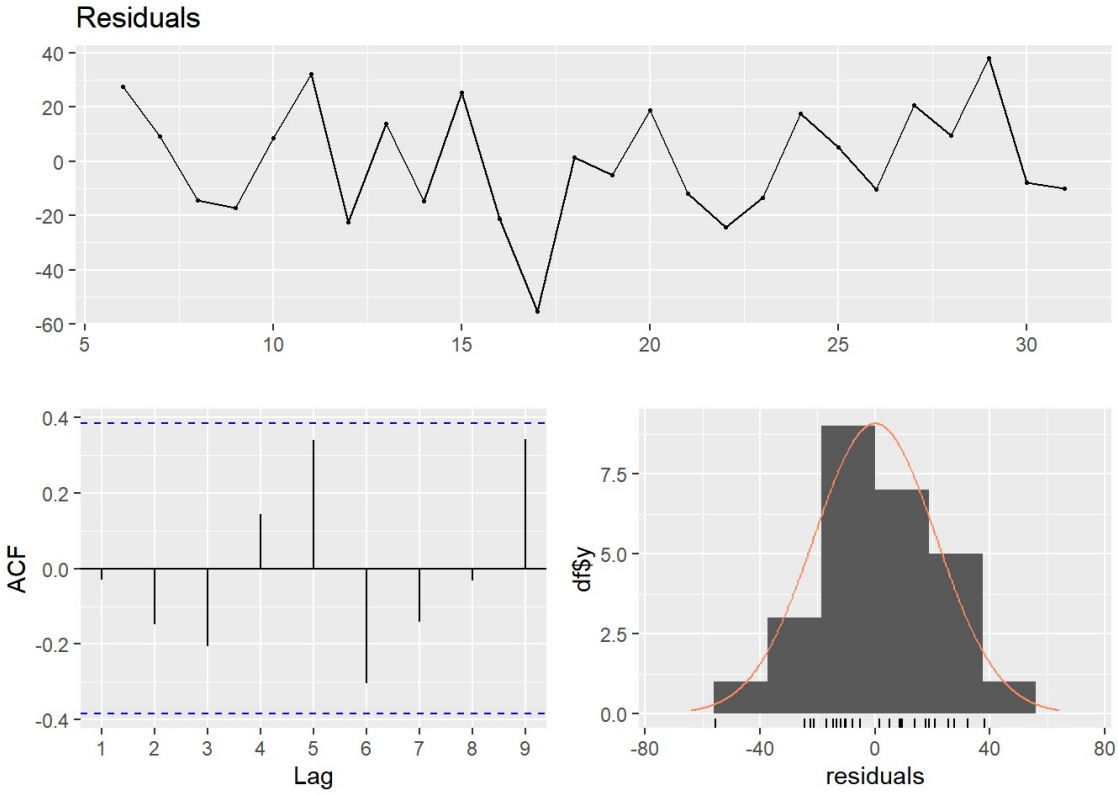
```
df = df[0,]

#relative humidity
for(i in 1:5){
  for(j in 1:5){
    model.ARDL = ardlDlm(x = as.vector(relhumid), y = as.vector(ffd), p = i, q = j)
    new_row = data.frame(p = i, q=j, AIC=AIC(model.ARDL$model), BIC=BIC(model.ARDL$model))
    df = bind_rows(df, new_row)
  }
}

df[order(df$AIC),]
```

```
##    p q      AIC      BIC
## 21 5 1 249.9992 261.3221
## 15 3 5 251.0833 264.9224
## 22 5 2 251.7412 264.3221
## 20 4 5 252.7653 267.8625
## 23 5 3 253.2512 267.0902
## 25 5 5 254.7429 271.0982
## 24 5 4 255.2458 270.3430
## 5  1 5 255.9194 267.2423
## 16 4 1 255.9681 266.3348
## 10 2 5 255.9861 268.5671
## 17 4 2 257.8014 269.4639
## 18 4 3 259.3579 272.3162
## 14 3 4 259.8661 272.8245
## 19 4 4 261.3174 275.5716
## 11 3 1 262.2897 271.6151
## 4  1 4 262.6906 273.0573
## 9  2 4 263.6474 275.3099
## 12 3 2 264.1836 274.8412
## 13 3 3 265.8767 277.8665
## 3  1 3 269.2856 278.6110
## 8  2 3 270.4134 281.0711
## 6  2 1 275.1189 283.3227
## 2  1 2 276.4240 284.6278
## 7  2 2 277.1178 286.6889
## 1  1 1 283.6338 290.6398
```

```
model.ARDL4 <- ardlDlm(x = as.vector(relhumid), y = as.vector(ffd), p = 5, q = 1)
summary(model.ARDL4)
```

```
##
## 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
## -55.410 -14.213  -1.856  16.775  38.095
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3326.7753  1794.9957   1.853   0.0803 .
## X.t           -4.9729     6.7908  -0.732   0.4734
## X.1           -5.9445     6.6754  -0.891   0.3849
## X.2           -4.3631     6.9986  -0.623   0.5408
## X.3          -13.3392     6.7739  -1.969   0.0645 .
## X.4           -3.2999     7.1864  -0.459   0.6516
## X.5           -0.6955     7.3031  -0.095   0.9252
## Y.1           -0.1601     0.2357  -0.679   0.5056
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 25.19 on 18 degrees of freedom
## Multiple R-squared:  0.2427, Adjusted R-squared:  -0.05177
## F-statistic: 0.8242 on 7 and 18 DF,  p-value: 0.5802
```

```
checkresiduals(model.ARDL4$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 11
##
## data:  Residuals
## LM test = 19.996, df = 11, p-value = 0.0454
```

```
MASE(model.ARDL4)
```

```
##                  MASE
## model.ARDL4 0.6035524
```

ARDL Model Summary

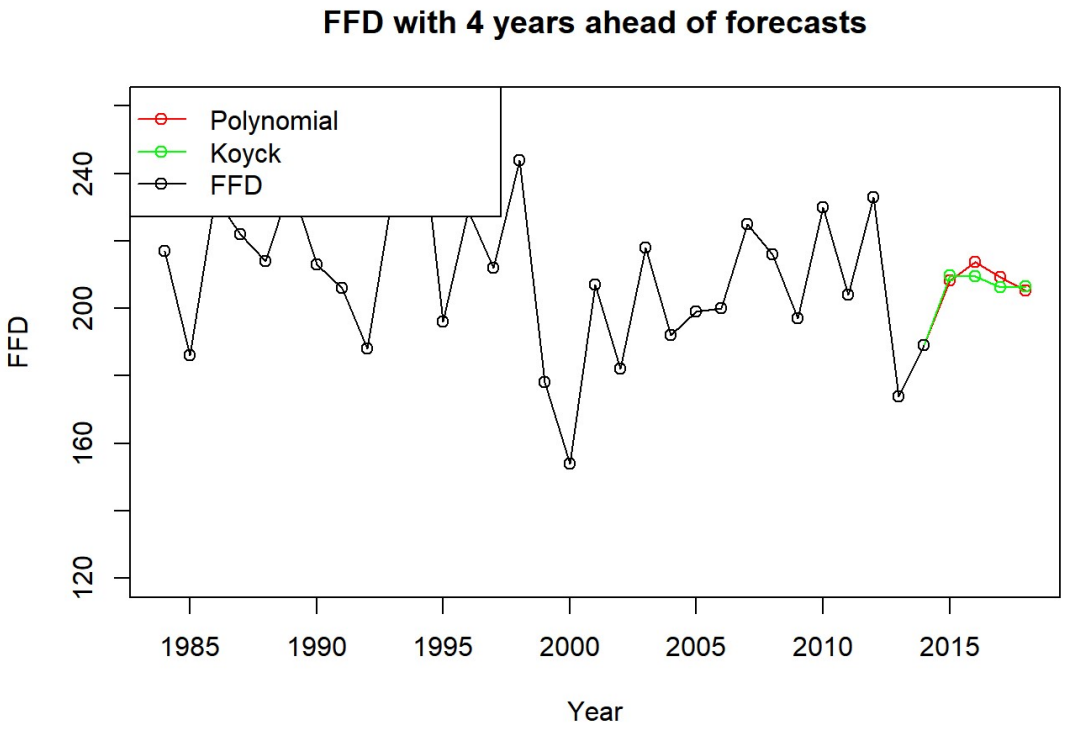
Model	MASE	Model p-value	Model R-squared value	Breusch-Godfrey p-value
Temperature	0.591	0.4912	-0.01279	0.2612
Rain	0.5484	0.3355	0.08137	0.04712
Radiation	0.587	0.7106	-0.1089	0.1157
Relative Humidity	0.603	0.5802	-0.05177	0.0454

Four different ARDL models are produced based on each predictor, one at a time. A process of iteration similar to the previous task is used to determine the ideal p and q values for each model and the best model is chosen based on the lowest AIC values. The summary and residuals of each of the models demonstrate that none of the models are significant due to high p-values and low R-squared values. All the models have no signs of serial correlation based on the high p-value obtained from the Breusch-Godfrey test. All the ARDL models have relatively low MASE compared to previous models. The ARDL model using rain as a predictor is chosen to be the best one based on its MASE, p-value, and R-squared value relative to other models.

Forecasting with Time Series Regression Models

Four years ahead forecasts are created for the polynomial and Koyck models. The resulting plot shows that Koyck and polynomial models produces somewhat similar forecasts, except the predictions of the Koyck model remain slightly lower than that pf the polynomial model until 2018, where their predictions align.

```
predictor <- read.csv("Covariate x-values for Task 2 .csv")
model.poly.Frc = dLagM::forecast(model.poly4, x = t(predictor$RelHumidity), h=4)
model.koyck.Frc = dLagM::forecast(model.Koyck3, x = t(predictor$Radiation), h=4)

plot(ts(c(as.vector(ffd), model.poly.Frc$forecasts), start = 1984, frequency = 1), type="o", col="red", ylim=
c(120, 260),
     ylab="FFD", xlab="Year", main="FFD with 4 years ahead of forecasts")
lines(ts(c(as.vector(ffd), model.koyck.Frc$forecasts), start = 1984, frequency = 1), col="green",type="o")
lines(ts(as.vector(ffd), start = 1984, frequency = 1),col="black",type="o")
legend("topleft",lty=1, pch = 1, text.width = 11, col=c( "red", "green", "black"),
       c("Polynomial", "Koyck", "FFD"))
```


FFD with 4 years ahead of forecasts
Polynomial
Koyck
FFD
FFD
120
160
200
240
1985
1990
1995
2000
2005
2010
2015
Year

Exponential Smoothing Models

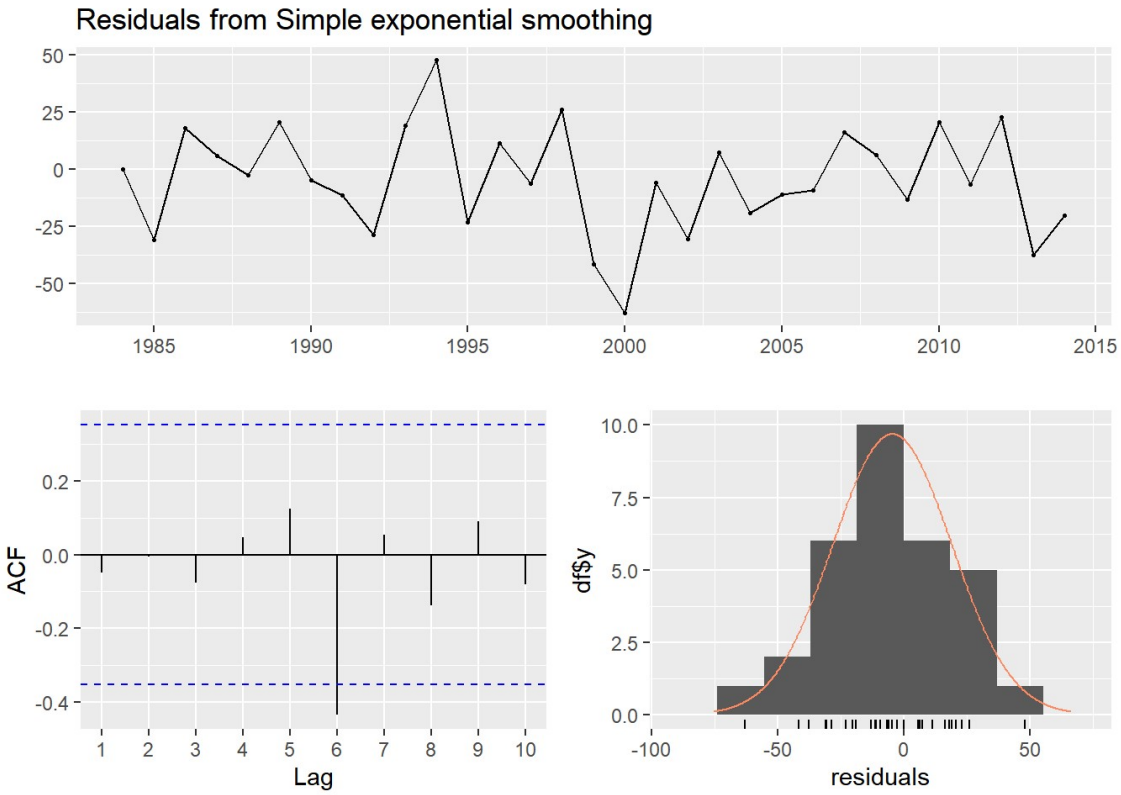
A simple exponential smoothing model, a Holt's linear trend model, and a Holt's linear trend model with additive damped trend are implemented using the FFD series. Models that take into account seasonal components are not considered due to the annual frequency of the time series data. The three above models are created for the FFD series using the ses() and holt() functions.

The results of the point forecasts show that the SES and the additive damped trend models produce a constant forecast at around the 200 mark. In contrast, the Holt's linear trend forecast starts at a lower FFD 183 and shows a constant decrease for the duration of the forecast.

```
fit1.ses <- ses(ffd, initial="simple", h=4)
summary(fit1.ses)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = ffd, h = 4, initial = "simple")
##
##   Smoothing parameters:
##     alpha = 0.0635
##
##   Initial states:
##     l = 217
##
##   sigma:  23.6628
## Error measures:
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -4.611341 23.66282 18.97233 -3.494419 9.522598 0.6711909
##                      ACF1
## Training set -0.04858377
##
## Forecasts:
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2015       207.9195 177.5944 238.2446 161.5412 254.2978
## 2016       207.9195 177.5332 238.3057 161.4477 254.3912
## 2017       207.9195 177.4723 238.3667 161.3545 254.4845
## 2018       207.9195 177.4114 238.4276 161.2614 254.5776
```

```
checkresiduals(fit1.ses)
```

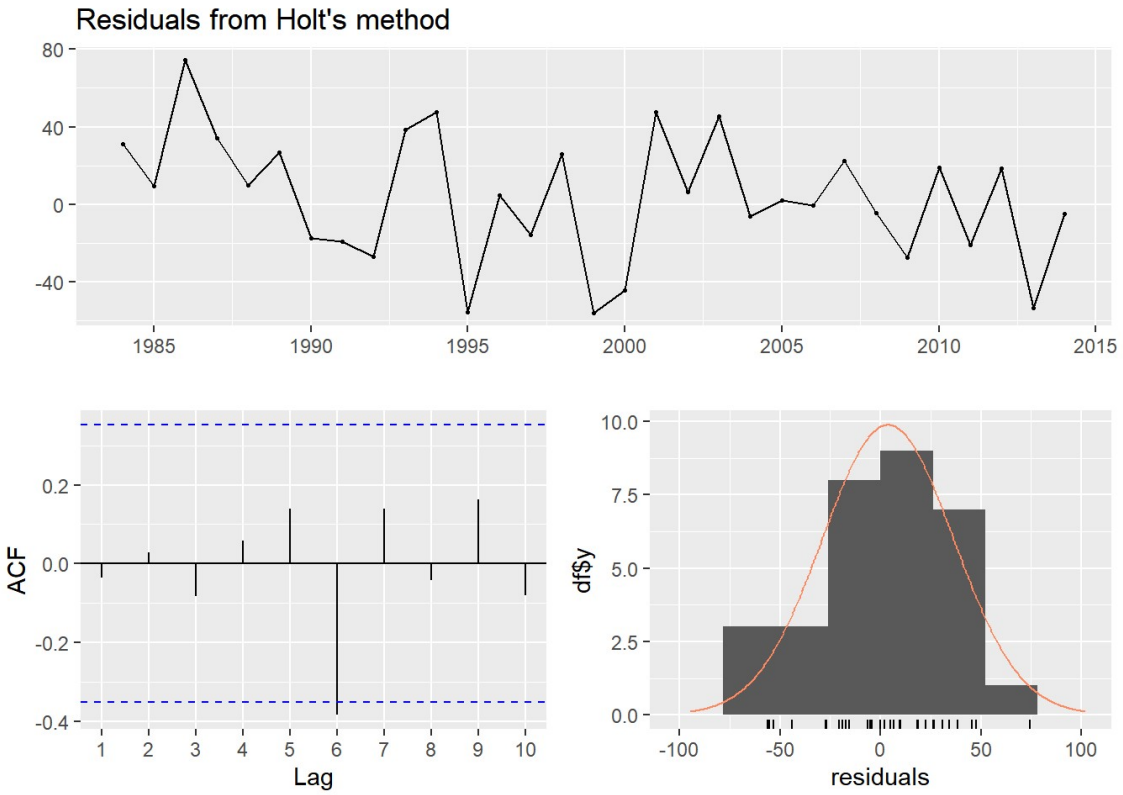



```
##
##  Ljung-Box test
##
## data:  Residuals from Simple exponential smoothing
## Q* = 8.7258, df = 6, p-value = 0.1896
##
## Model df: 0.   Total lags used: 6
```

```
fit2.holt <- holt(ffd, initial="simple", h=4)
summary(fit2.holt)
```

```
##
## Forecast method: Holt's method
##
## Model Information:
## Holt's method
##
## Call:
##  holt(y = ffd, h = 4, initial = "simple")
##
##   Smoothing parameters:
##     alpha = 0.4947
##     beta  = 0.4105
##
##   Initial states:
##     l = 217
##     b = -31
##
##   sigma:  32.4048
## Error measures:
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 3.643616 32.40477 26.33733 0.4670524 12.74033 0.9317453
##                      ACF1
## Training set -0.03620177
##
## Forecasts:
##      Point Forecast     Lo 80    Hi 80      Lo 95    Hi 95
## 2015       183.4402 141.91186 224.9686 119.928060 246.9524
## 2016       175.3764 119.36198 231.3909  89.709726 261.0431
## 2017       167.3126  89.06345 245.5618  47.640838 286.9844
## 2018       159.2488  53.12709 265.3705  -3.050347 321.5479
```

```
checkresiduals(fit2.holt)
```

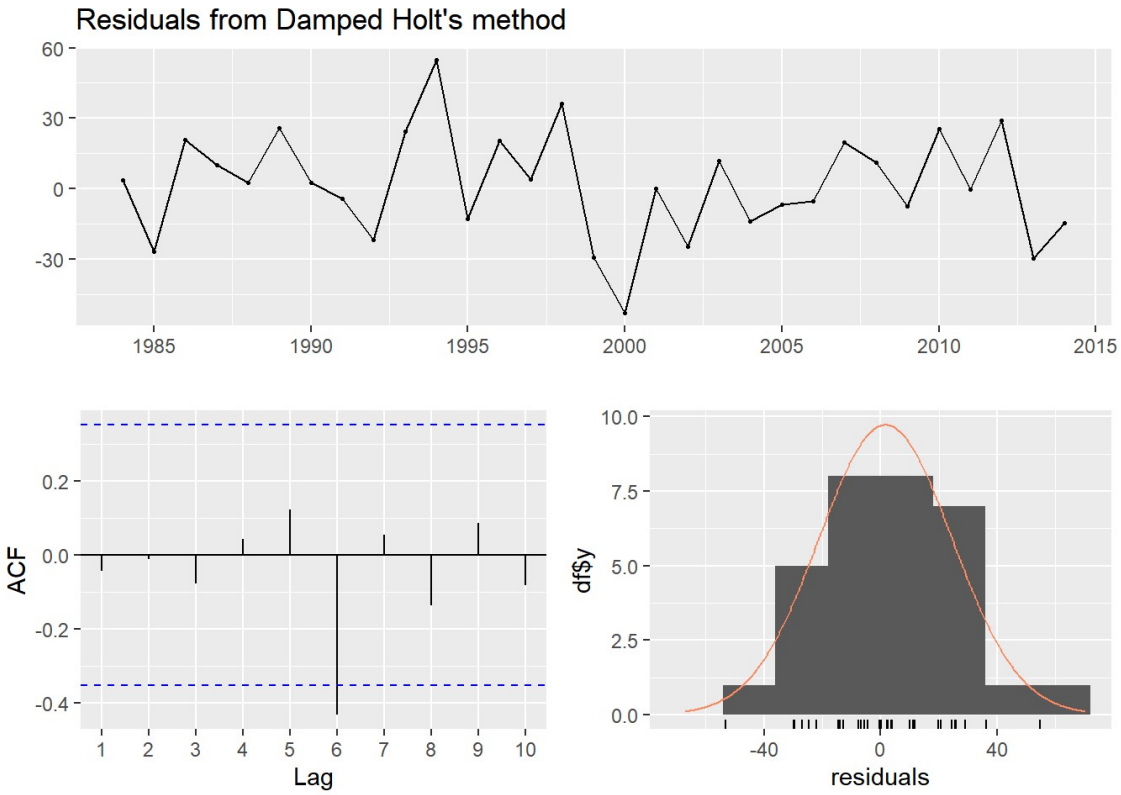


```
##
##  Ljung-Box test
##
## data:  Residuals from Holt's method
## Q* = 7.2593, df = 6, p-value = 0.2975
##
## Model df: 0.   Total lags used: 6
```

```
fit3.holt <- holt(ffd, damped=TRUE, initial="simple", h=4) # Fit with additive damped trend
summary(fit3.holt)
```

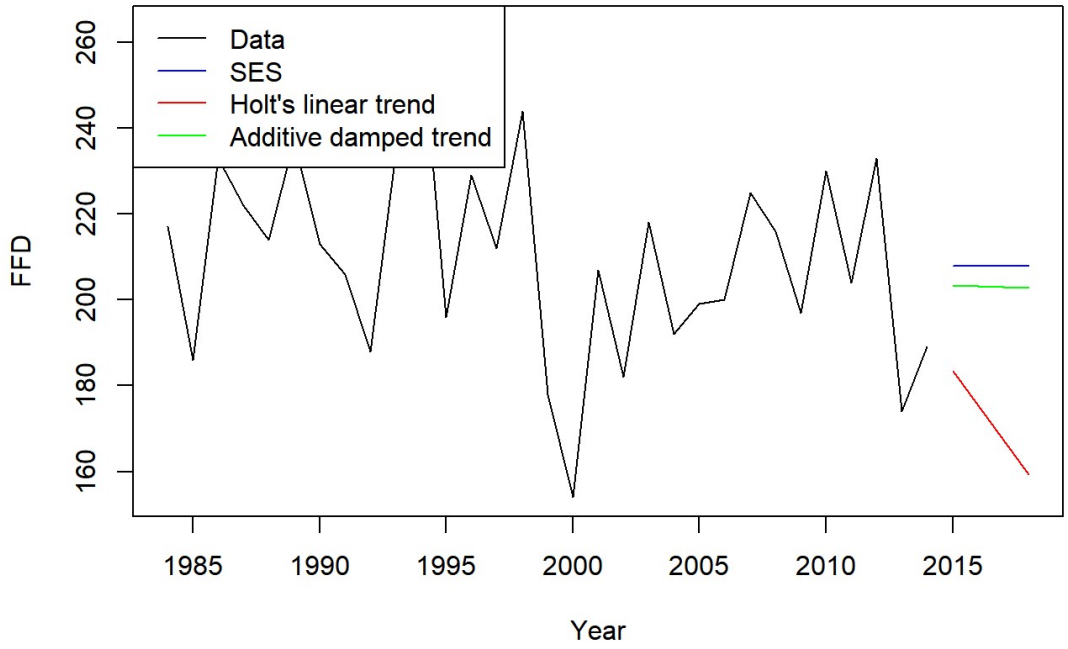
```
##
## Forecast method: Damped Holt's method
##
## Model Information:
## Damped Holt's method
##
## Call:
##  holt(y = ffd, h = 4, damped = TRUE, initial = "simple")
##
##   Smoothing parameters:
##     alpha = 1e-04
##     beta  = 1e-04
##     phi   = 0.9724
##
##   Initial states:
##     l = 213.8104
##     b = -0.5047
##
##   sigma:  24.6844
##
##      AIC     AICc      BIC
## 311.7836 315.2836 320.3875
##
## Error measures:
##                    ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 1.646287 22.60624 17.87068 -0.4309696 8.698589 0.6322173
##                    ACF1
## Training set -0.0440588
##
## Forecasts:
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2015       203.4051 171.7707 235.0394 155.0245 251.7856
## 2016       203.2069 171.5725 234.8412 154.8263 251.5874
## 2017       203.0142 171.3798 234.6485 154.6336 251.3947
## 2018       202.8268 171.1924 234.4611 154.4462 251.2073
```

```
checkresiduals(fit3.holt)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt's method
## Q* = 8.6078, df = 6, p-value = 0.1969
##
## Model df: 0.   Total lags used: 6
```

```
plot(ffd, type="l", ylab="FFD", xlab="Year", fcol="white", plot.conf=FALSE, xlim=c(1984,2018))
lines(fit1.ses$mean, col="blue", type="l")
lines(fit2.holt$mean, col="red", type="l")
lines(fit3.holt$mean, col="green", type="l")
legend("topleft", lty=1, col=c("black","blue","red","green"),
       c("Data","SES", "Holt's linear trend", "Additive damped trend"))
```



Model Summary

Model	AIC	BIC	MASE	Ljung-Box p-value	Residual Normality
Simple SES	NA	NA	0.6711909	0.1896	Normal
Simple Holt	NA	NA	0.9317453	0.2975	Non-normal
Additive Damped Trend Holt SES	311.7836	320.3875	0.6322173	0.1969	Non-normal

Checking the summaries and residuals for each model reveals that the Simple SES and Additive Damped Trend Holt SES have lower MASE values than the Simple Holt model. It is difficult to determine normality of residuals from the output provided as there are not many data points in the dataset; however, it is possible to assume that Simple SES has normal residuals based on the residuals plots. Considering all the metrics from the summary and residuals, the Additive Damped Trend Holt SES may be chosen as the best Simple exponential smoothing model.

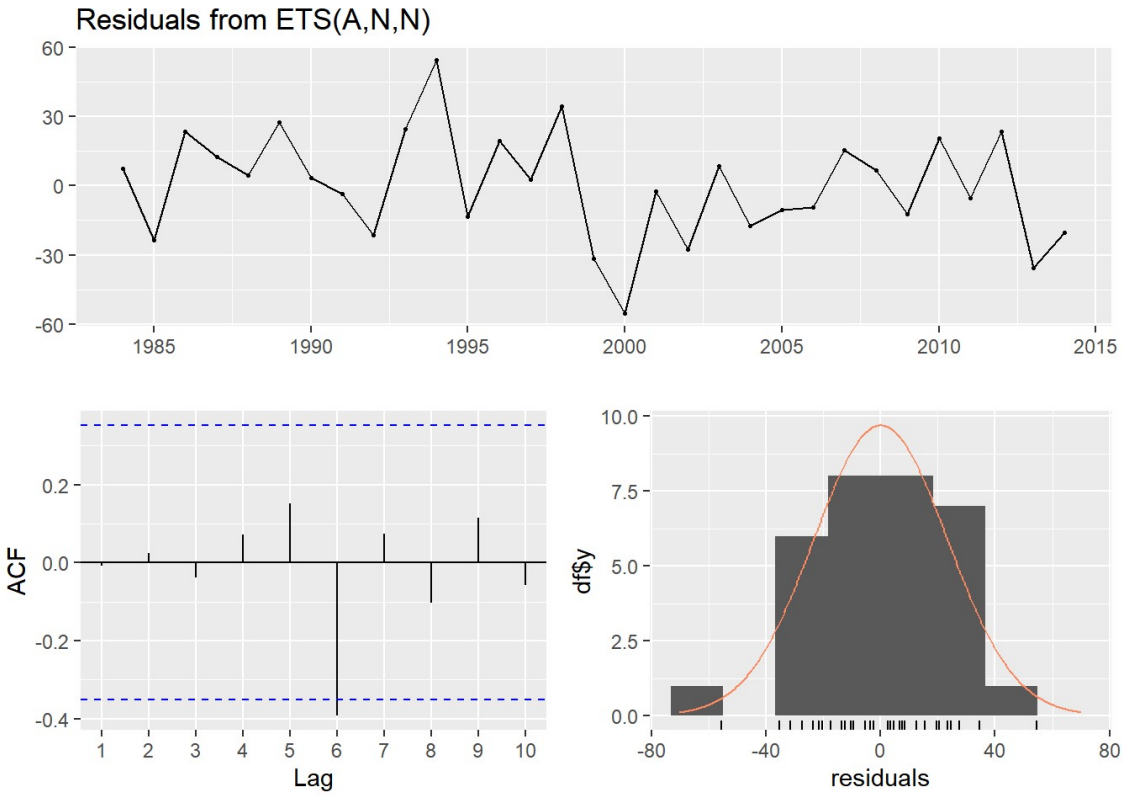
State Space Models

ANN, MNN, AAN, MMN, and MAN state space models are created for FFD and their residuals and summaries are analysed below.

```
etsAN = ets(ffd, model="ANN")
summary(etsAN)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = ffd, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 209.4516
##
##   sigma:  23.8149
##
##      AIC     AICc      BIC
## 306.9455 307.8343 311.2474
##
## Training set error measures:
##                       ME     RMSE      MAE       MPE     MAPE     MASE
## Training set -0.004800665 23.03386 18.69589 -1.265312 9.173703 0.661411
##                      ACF1
## Training set -0.006864147
```

```
checkresiduals(etsAN)
```

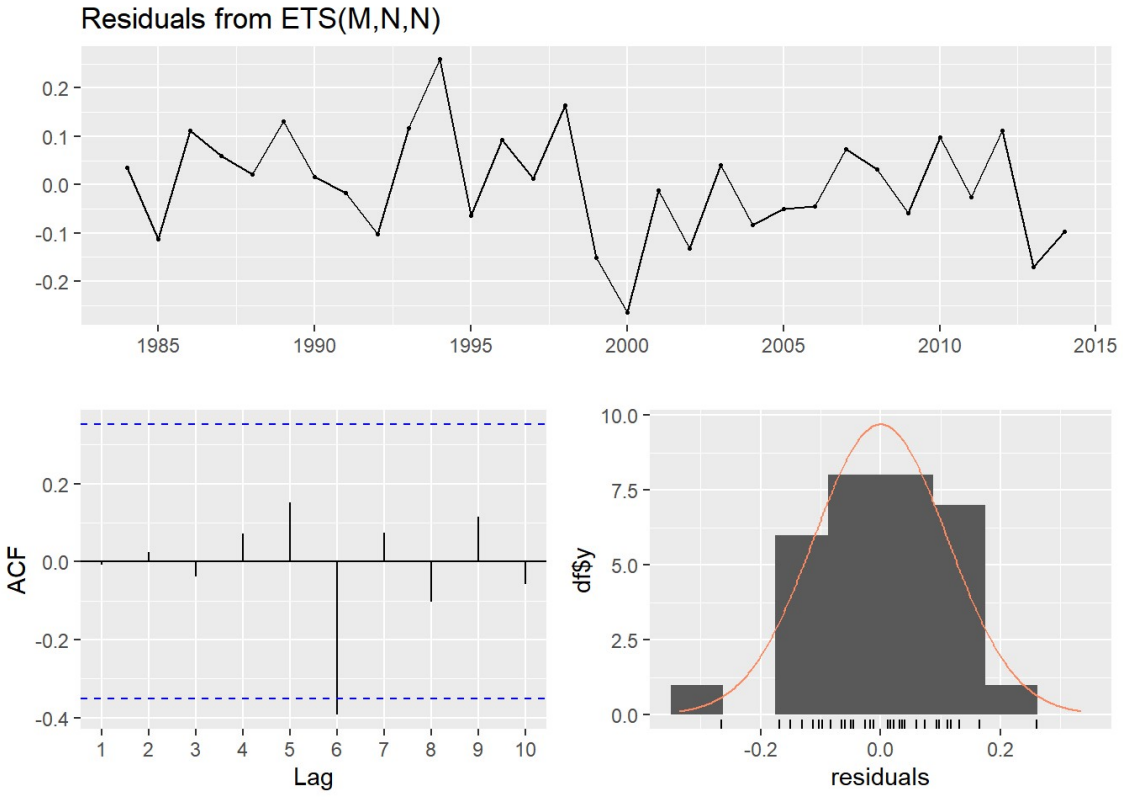


```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,N)
## Q* = 7.489, df = 6, p-value = 0.278
##
## Model df: 0.   Total lags used: 6
```

```
etsMN = ets(ffd, model="MNN")
summary(etsMN)
```

```
## ETS(M,N,N)
##
## Call:
##  ets(y = ffd, model = "MNN")
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 209.448
##
##   sigma:  0.1137
##
##      AIC     AICc      BIC
## 306.9448 307.8337 311.2468
##
## Training set error measures:
##                        ME     RMSE    MAE       MPE     MAPE      MASE
## Training set -0.001214657 23.03386 18.696 -1.263579 9.173601 0.6614151
##                      ACF1
## Training set -0.006864096
```

```
checkresiduals(etsMN)
```

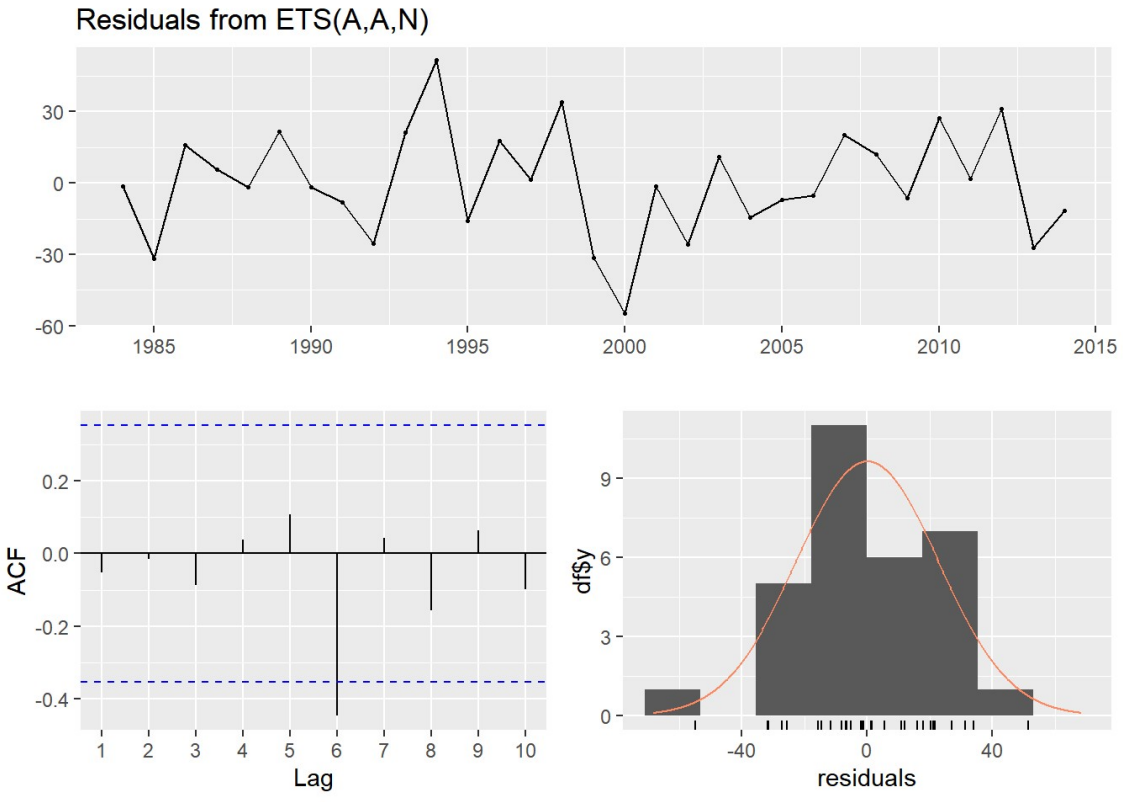


```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,N)
## Q* = 7.489, df = 6, p-value = 0.278
##
## Model df: 0.   Total lags used: 6
```

```
etsAA = ets(ffd, model="AAN")
summary(etsAA)
```

```
## ETS(A,A,N)
##
## Call:
##  ets(y = ffd, model = "AAN")
##
##   Smoothing parameters:
##     alpha = 1e-04
##     beta  = 1e-04
##
##   Initial states:
##     l = 218.839
##     b = -0.5898
##
##   sigma:  24.0066
##
##      AIC     AICc      BIC
## 309.2274 311.6274 316.3973
##
## Training set error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.04681346 22.40433 17.52335 -1.174704 8.595767 0.6199298
##                      ACF1
## Training set -0.05273509
```

```
checkresiduals(etsAA)
```

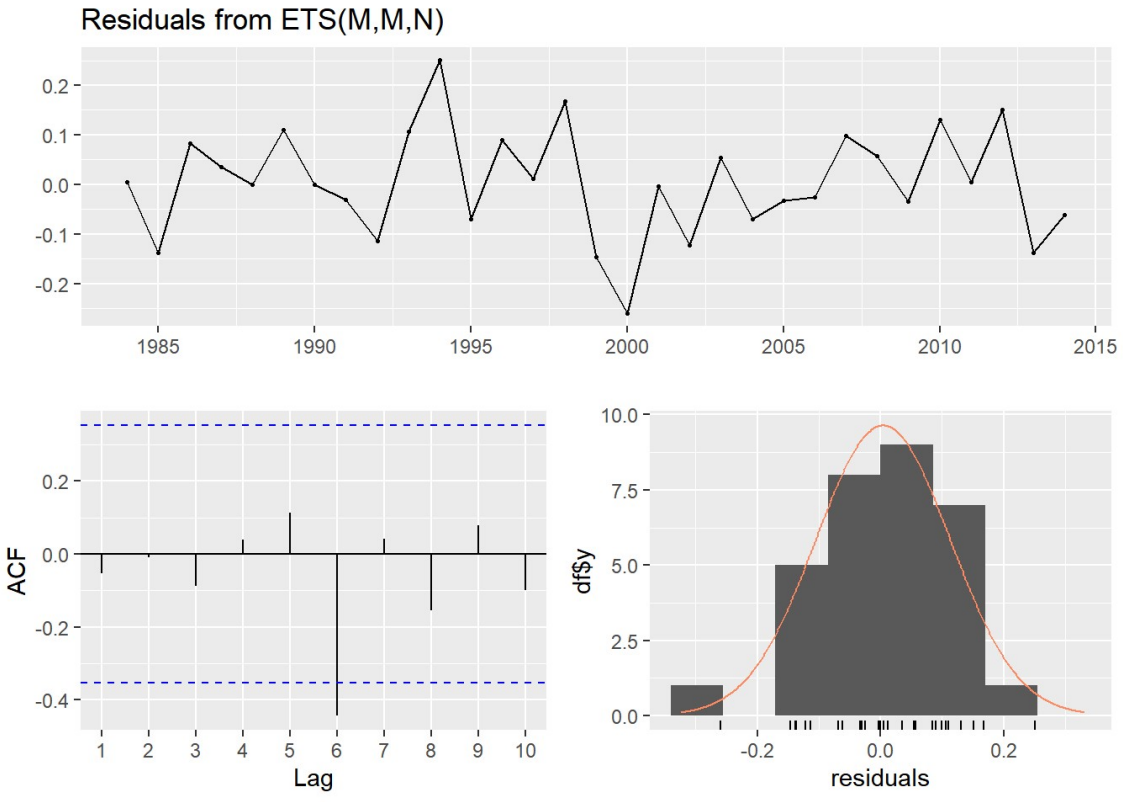


```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,A,N)
## Q* = 9.0429, df = 6, p-value = 0.1712
##
## Model df: 0.   Total lags used: 6
```

```
etsMM = ets(ffd, model="MMN")
summary(etsMM)
```

```
## ETS(M,M,N)
##
## Call:
##  ets(y = ffd, model = "MMN")
##
##   Smoothing parameters:
##     alpha = 1e-04
##     beta  = 1e-04
##
##   Initial states:
##     l = 216.4878
##     b = 0.9977
##
##   sigma:  0.1153
##
##      AIC     AICc      BIC
## 309.3333 311.7333 316.5032
##
## Training set error measures:
##                      ME    RMSE      MAE        MPE     MAPE      MASE
## Training set 0.8316197 22.443 17.48424 -0.8068944 8.548035 0.6185461
##                      ACF1
## Training set -0.05190951
```

```
checkresiduals(etsMM)
```

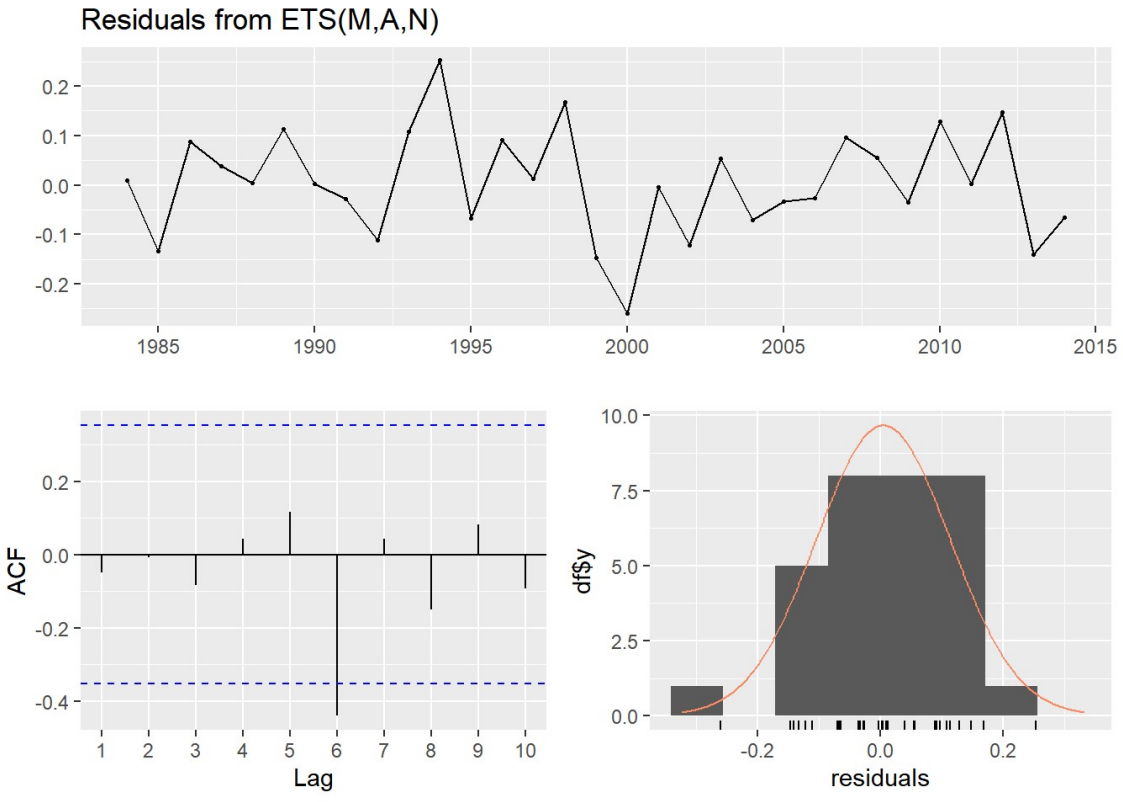


```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,M,N)
## Q* = 8.9792, df = 6, p-value = 0.1748
##
## Model df: 0.   Total lags used: 6
```

```
etsMA = ets(ffd, model="MAN")
summary(etsMA)
```

```
## ETS(M,A,N)
##
## Call:
##  ets(y = ffd, model = "MAN")
##
##   Smoothing parameters:
##     alpha = 1e-04
##     beta  = 1e-04
##
##   Initial states:
##     l = 215.3797
##     b = -0.4305
##
##   sigma:  0.1155
##
##      AIC     AICc      BIC
## 309.4179 311.8179 316.5879
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 0.9196759 22.47637 17.60976 -0.7708524 8.605156 0.6229869
##                      ACF1
## Training set -0.04986601
```

```
checkresiduals(etsMA)
```



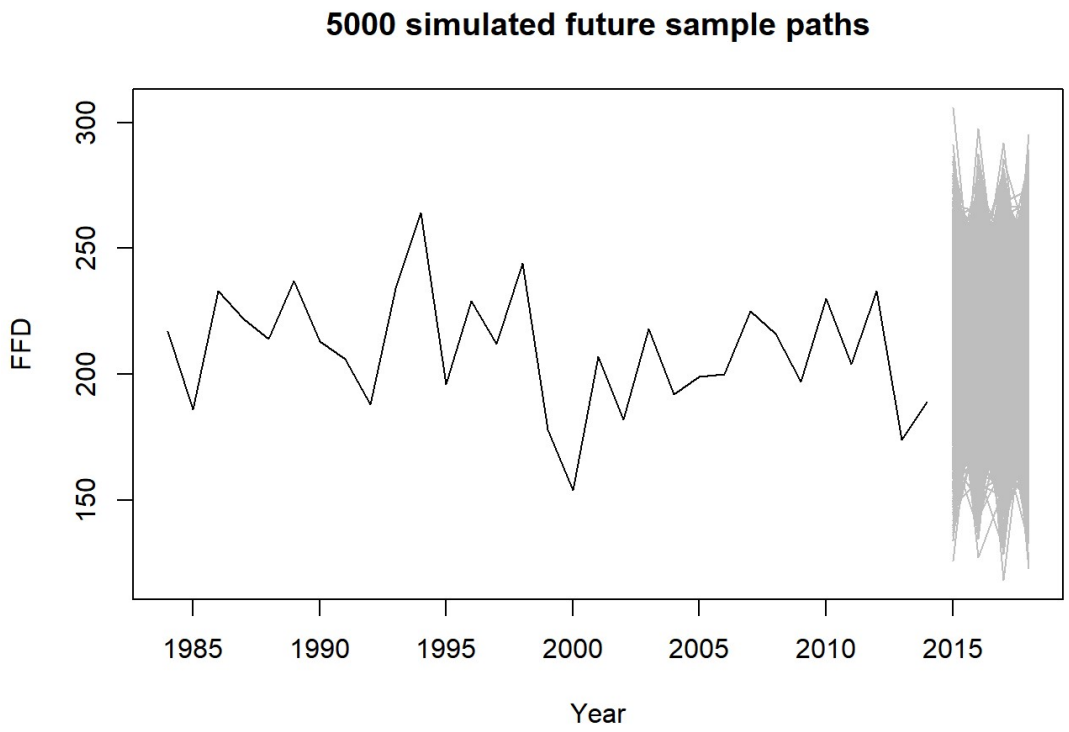
```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,A,N)
## Q* = 8.8804, df = 6, p-value = 0.1804
##
## Model df: 0.   Total lags used: 6
```

Model Summary

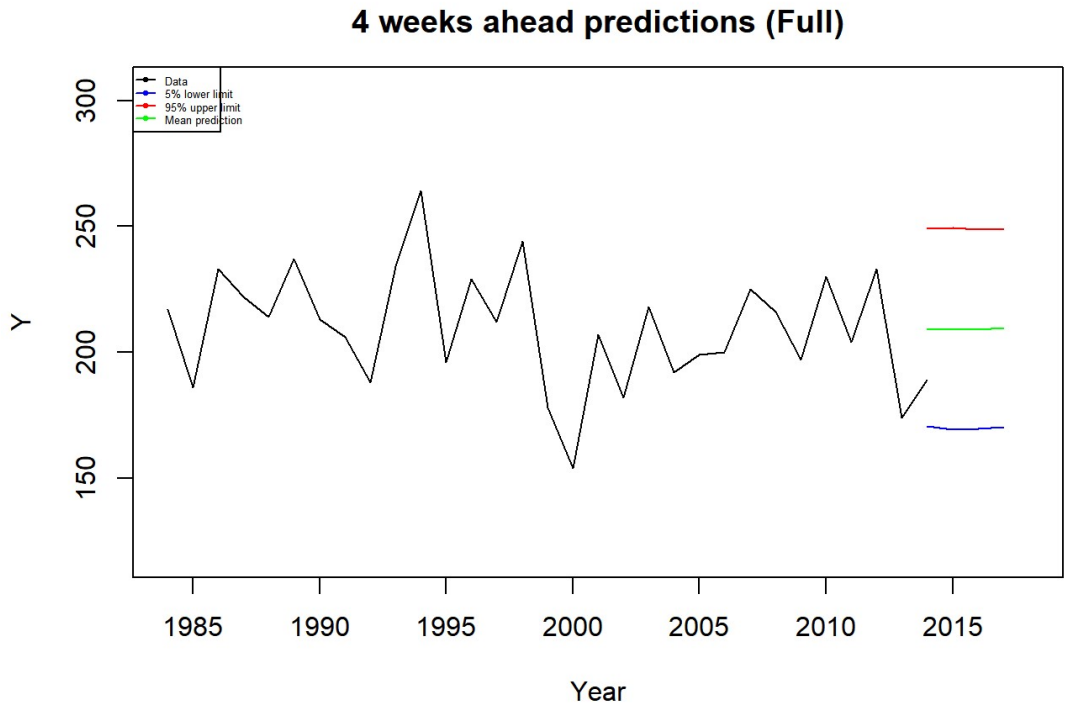
Model	AIC	BIC	MASE	Ljung-Box p-value
ANN State Space	306.9455	311.2474	0.661411	0.278
MNN State Space	306.9448	311.2468	0.6614151	0.278
AAN State Space	309.2274	316.3973	0.6199298	0.1712
MMN State Space	309.3333	316.5032	0.6185461	0.1748
MAN State Space	309.4179	316.5879	0.6229869	0.1804

The lowest MASE is seen in the MMN model, while the lowest AIC and BIC are observed in the MNN model. All the models do not seem to contain serial correlation due to high p-values from the Ljung-Box test. Based on these, the final model chosen was the MNN state space model due to its overall combination of low AIC and BIC, while having a similar MASE value compared to the other models.

```
A = ts(matrix(NA,4,5000),start=2015,frequency = 1)

M = 5000
for (i in 1:M){
  A[,i] = simulate(etsAN , initstate = etsAN$states[31,] , nsim = 4)
}

plot(ffd , ylim=range(ffd,A) , xlim=c(1984,2018),
     ylab="FFD" , xlab="Year", main="5000 simulated future sample paths")
for(i in 1:M){
  lines(A[,i],col="gray")
}
```




```
# interval estimates and bounds

N = 4
xlim=c(1984,2018)

Pi = array(NA, dim=c(N,2))
avrg = array(NA, N)

# Calculate the interval estimates and mid point
for (i in 1:N){
  Pi[i,] = quantile(A[i,], type=8, prob=c(.05,.95))
  avrg[i] = mean(A[i,]) # This would be median as well
}

# Create ts objects for plotting
Pi.lb = ts(Pi[,1],start=end(ffd),f=1)
Pi.ub = ts(Pi[,2],start=end(ffd),f=1)
avrg.pred = ts(avrg,start=end(ffd),f=1)

plot(ffd,xlim=xlim , ylim=range(ffd,A),ylab="Y",xlab="Year", main="
4 weeks ahead predictions (Full)")
lines(Pi.lb,col="blue", type="l")
lines(Pi.ub,col="red", type="l")
lines(avrg.pred,col="green", type="l")
legend("topleft", lty=1, pch=1, col=c("black","blue","red","green"), text.width =2, cex=0.4,
       c("Data","5% lower limit","95% upper limit","Mean prediction"))
```




```
##########################################################################################################################
#####################
```

5000 simulated future paths are plotted to obtain an idea of the potential range of forecasts that could be obtained. The simulated paths demonstrate a periodic rise and fall. Plotting the forecast along with the confidence intervals shows a roughly constant forecast (green) at around 210. The confidence intervals seem quite wide, ranging from roughly 170 till 250 and both bounds seem equidistant from the forecast.

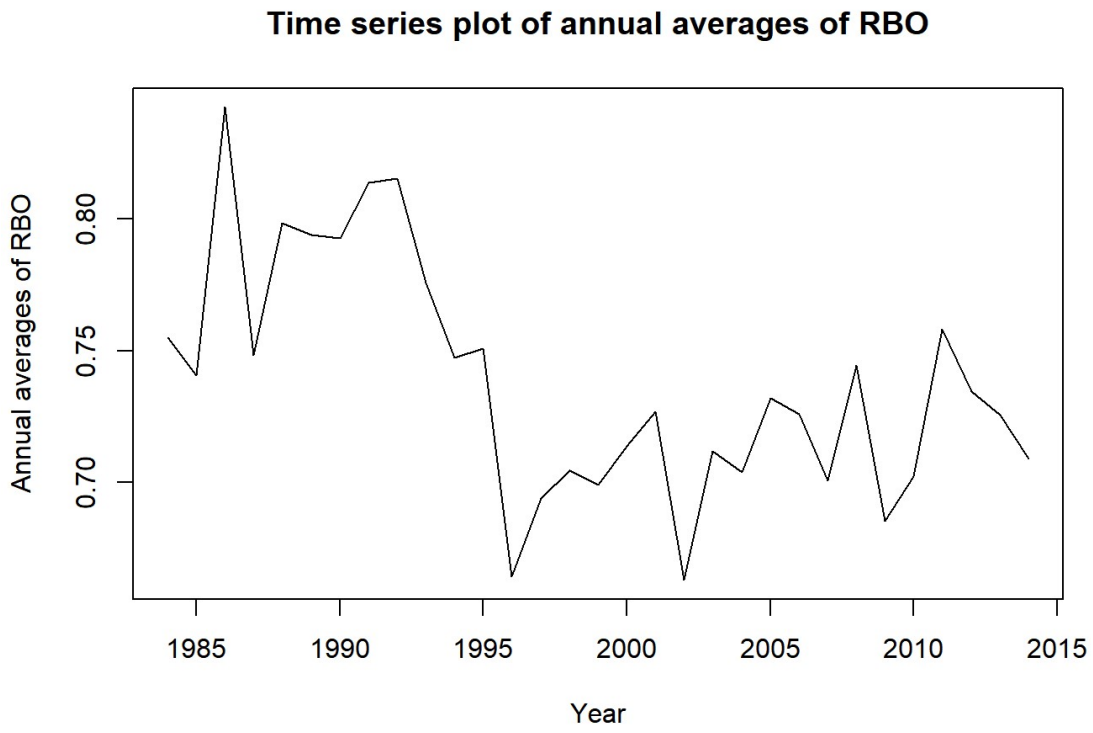
Task 3 Part A

Time series regression models, state space models, and exponential smoothing models are explored to create 3 years ahead forecasts for the Rank-based Order similarity metric (RBO) and various climate conditions which act as predictor variables. Each series is plotted to observe overall trends.

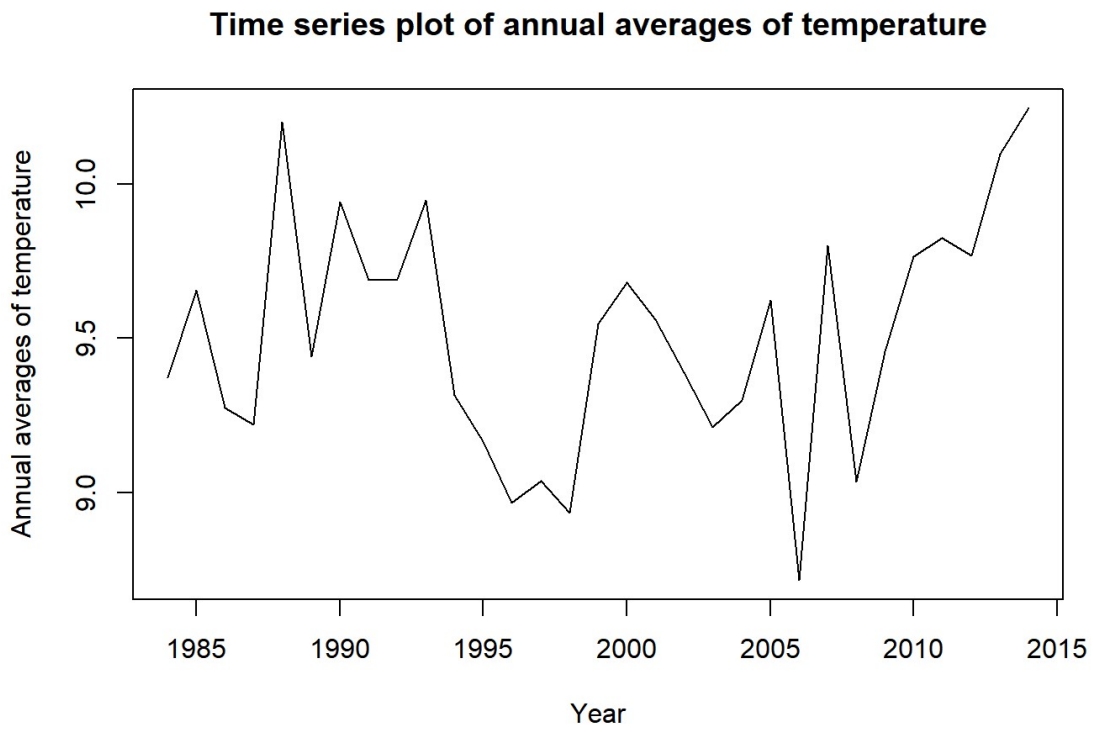
```
rbodata <- read.csv("RBO .csv")
head(rbodata)
```

```
##   Year       RBO Temperature Rainfall Radiation RelHumidity
## 1 1984 0.7550088    9.371585 2.489344  14.87158    93.92650
## 2 1985 0.7407520    9.656164 2.475890  14.68493    94.93589
## 3 1986 0.8423860    9.273973 2.421370  14.51507    94.09507
## 4 1987 0.7484425    9.219178 2.319726  14.67397    94.49699
## 5 1988 0.7984084   10.202186 2.465301  14.74863    94.08142
## 6 1989 0.7938803    9.441096 2.735890  14.78356    96.08685
```

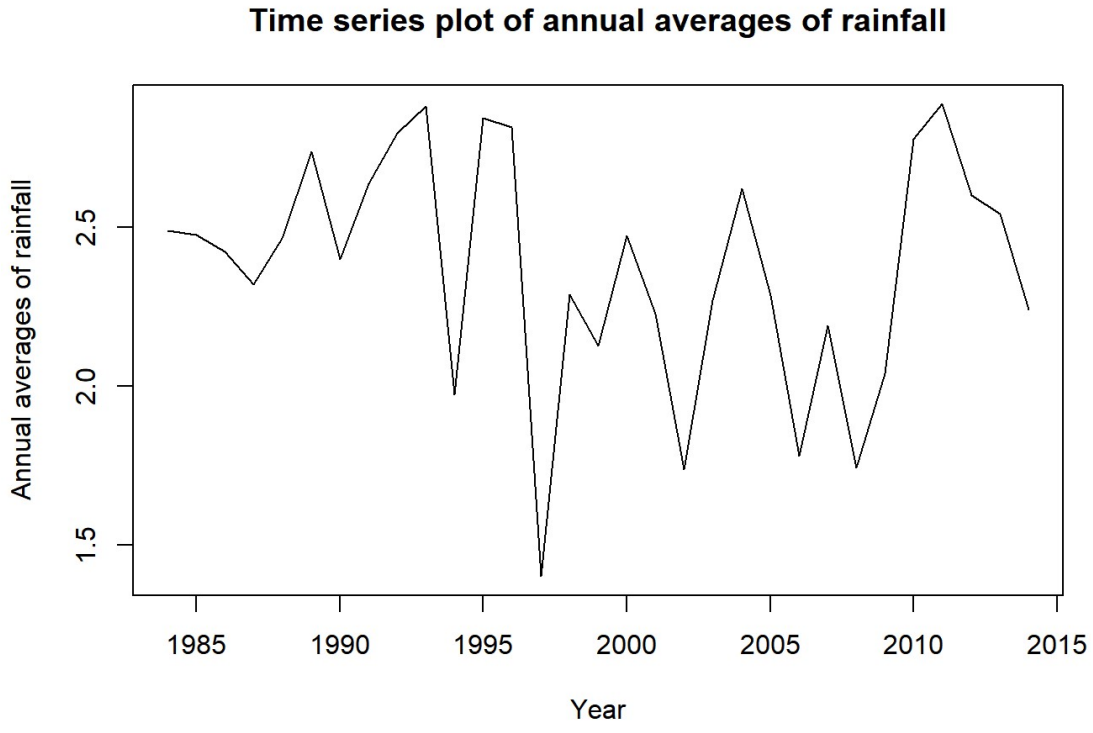
```
rbo <- ts(rbodata$RBO, start=c(1984,1), frequency=1)
temp <- ts(rbodata$Temperature, start=c(1984,1), frequency=1)
rain <- ts(rbodata$Rainfall, start=c(1984,1), frequency=1)
radiation <- ts(rbodata$Radiation, start=c(1984,1), frequency=1)
relhumid <- ts(rbodata$RelHumidity, start=c(1984,1), frequency=1)

plot(rbo, ylab='Annual averages of RBO',xlab='Year',
     main = "Time series plot of annual averages of RBO")
```



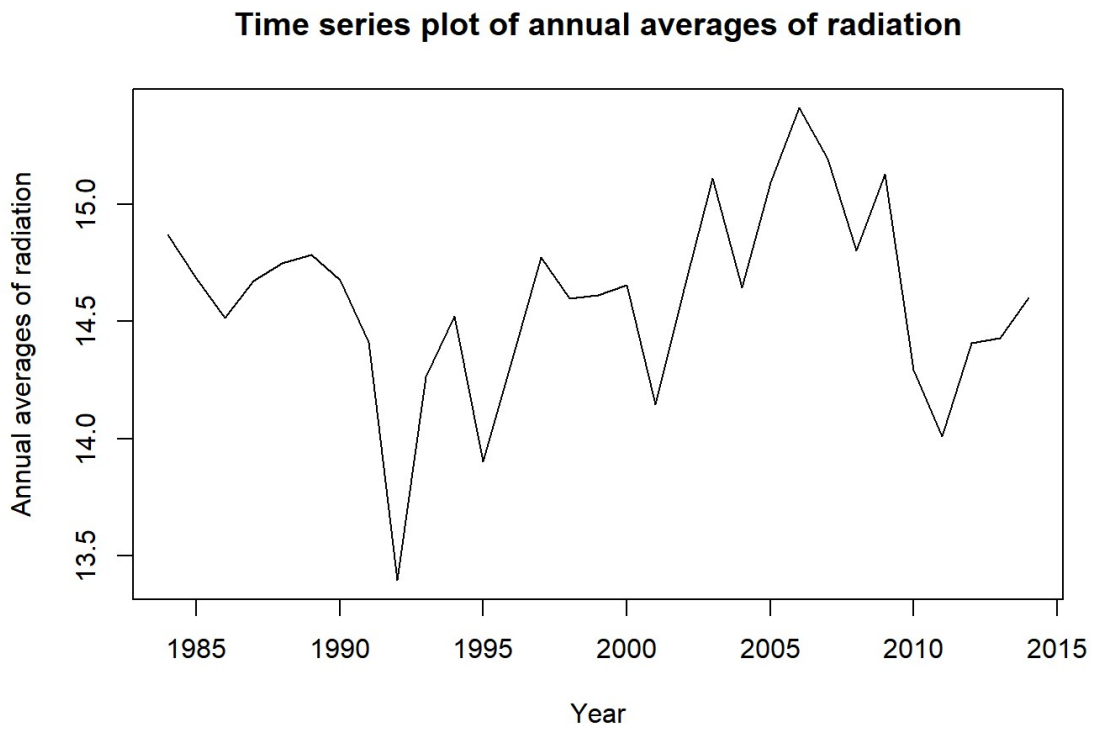

```
plot(temp, ylab='Annual averages of temperature',xlab='Year',
     main = "Time series plot of annual averages of temperature")
```



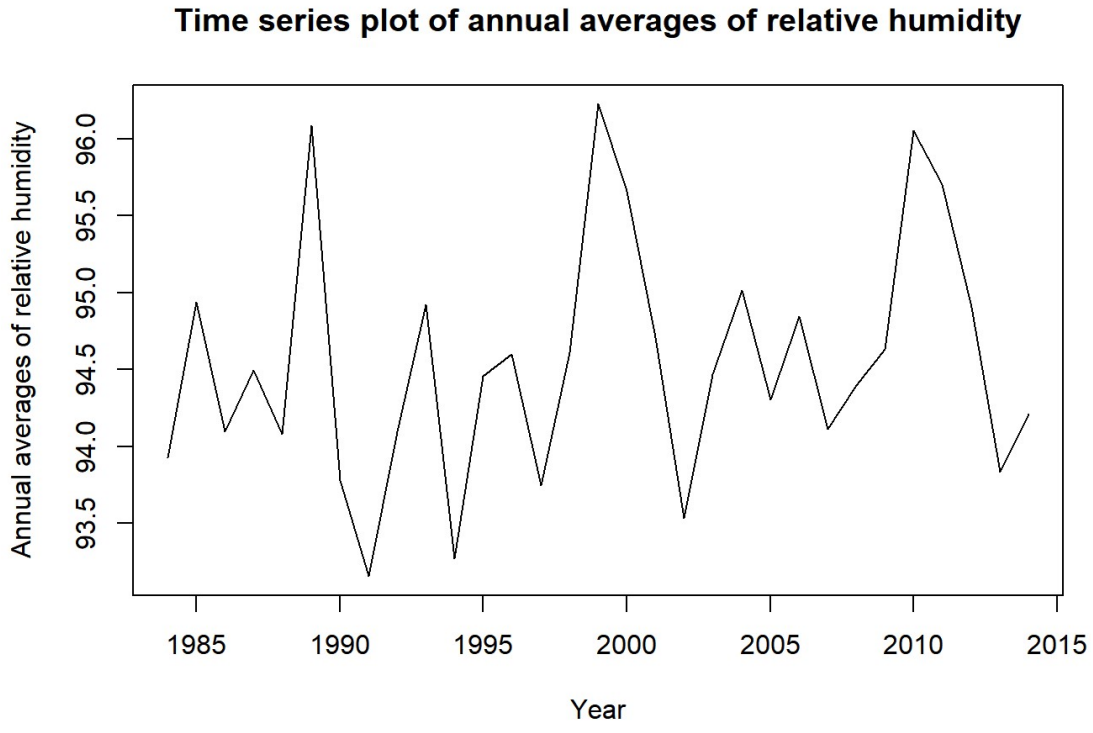

```
plot(rain, ylab='Annual averages of rainfall',xlab='Year',
     main = "Time series plot of annual averages of rainfall")
```




```
plot(radiation, ylab='Annual averages of radiation',xlab='Year',
     main = "Time series plot of annual averages of radiation")
```



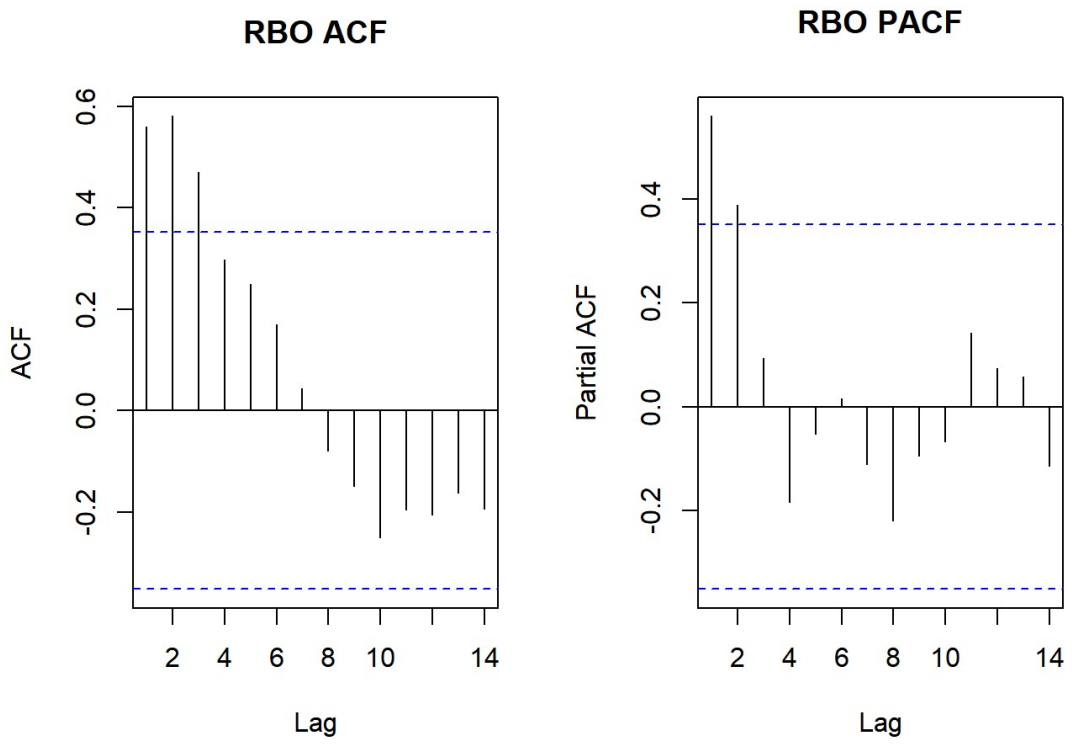

```
plot(relhumid, ylab='Annual averages of relative humidity',xlab='Year',
     main = "Time series plot of annual averages of relative humidity")
```



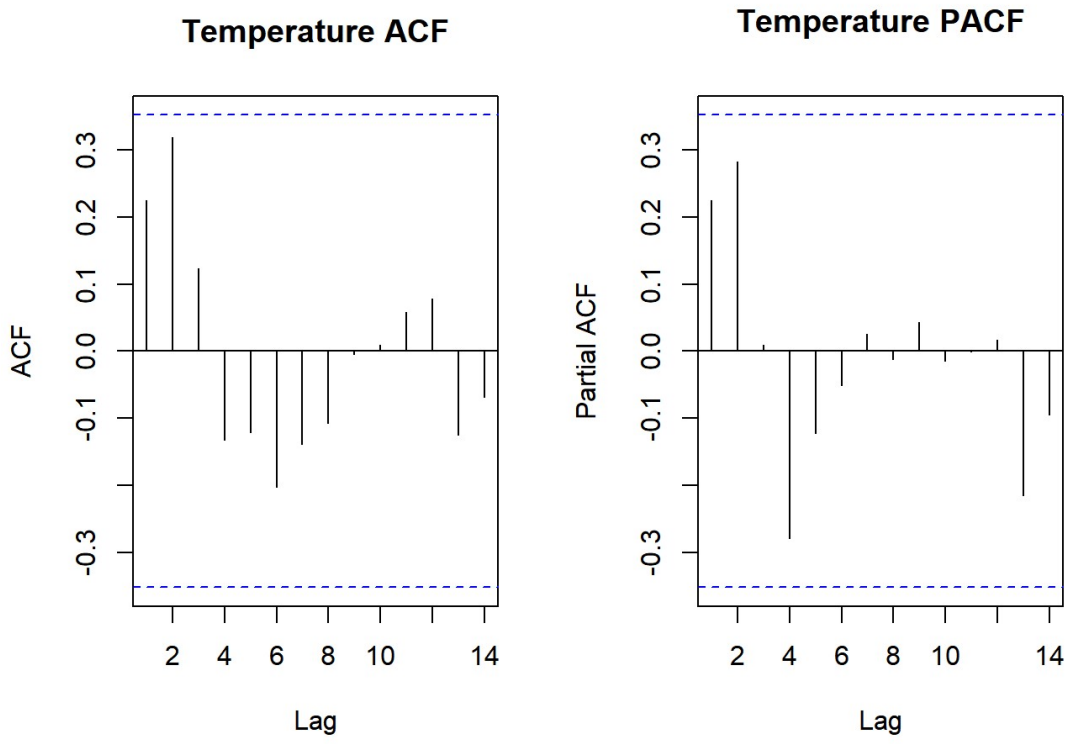
Exploring Stationarity of Variables

Plotting the ACF and PACF for the series demonstrates similar results for both variables. All the series have most peaks lying below the 95% confidence interval, with RBO and temperature displaying a more periodic rise and fall compared to the other variables. This suggests that the series are likely stationary.

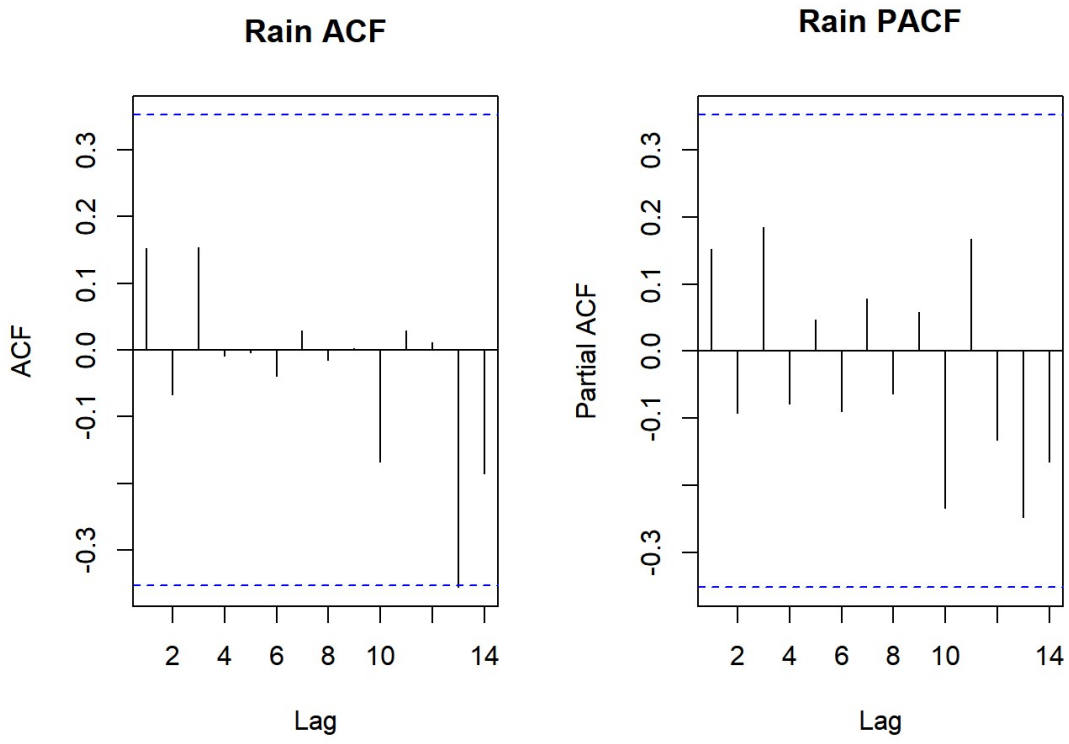
```
par(mfrow=c(1,2))
acf(rbo, max.lag = 24, main = "RBO ACF")
pacf(rbo, max.lag = 24, main = "RBO PACF")
```



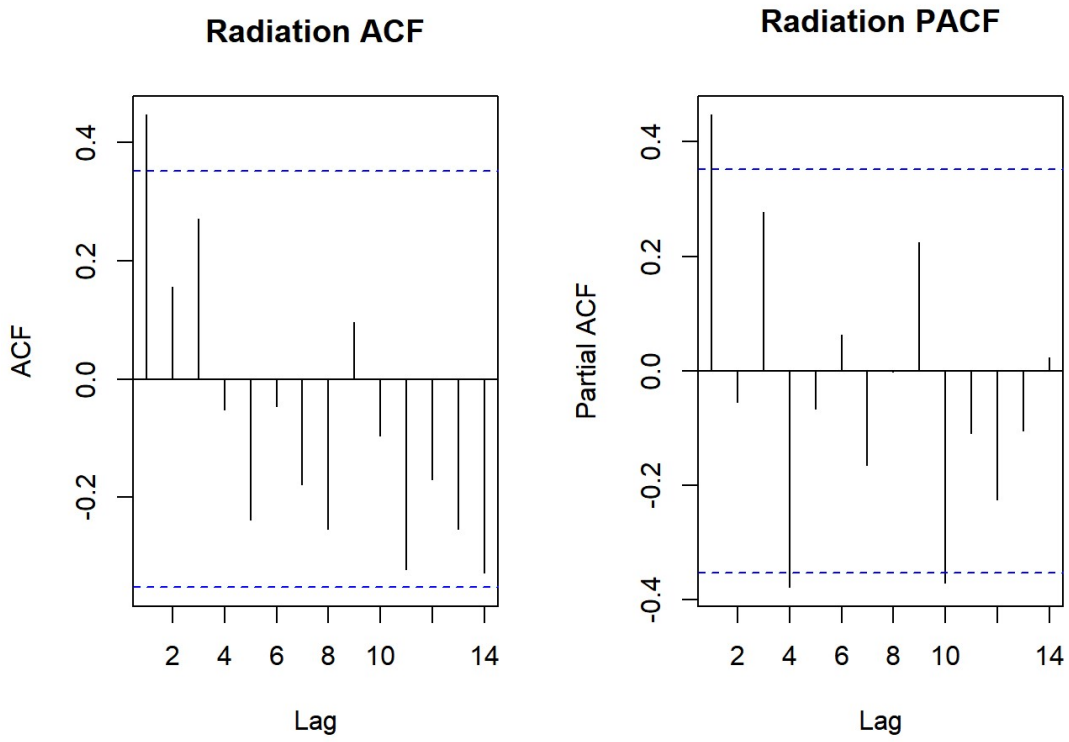

```
par(mfrow=c(1,2))
acf(temp, max.lag = 24, main="Temperature ACF")
pacf(temp, max.lag = 24, main = "Temperature PACF")
```



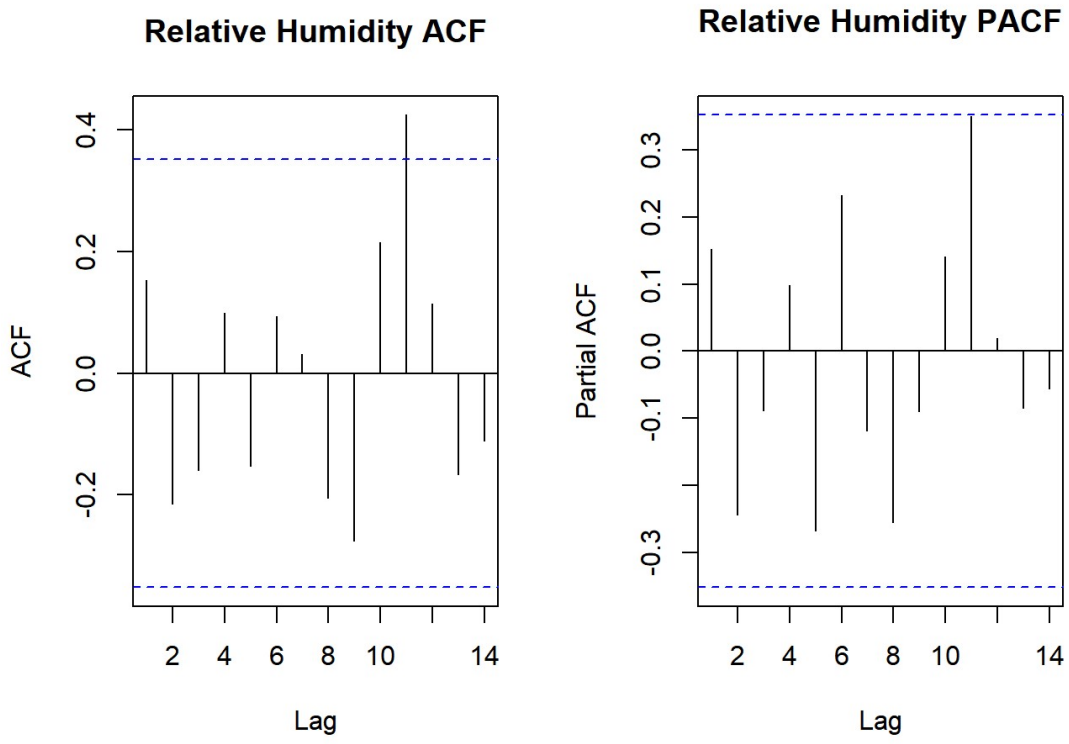

```
par(mfrow=c(1,2))
acf(rain, max.lag = 24, main = "Rain ACF")
pacf(rain, max.lag = 24, main = "Rain PACF")
```




```
par(mfrow=c(1,2))
acf(radiation, max.lag = 24, main = "Radiation ACF")
pacf(radiation, max.lag = 24, main = "Radiation PACF")
```




```
par(mfrow=c(1,2))
acf(relhumid, max.lag = 24, main = "Relative Humidity ACF")
pacf(relhumid, max.lag = 24, main = "Relative Humidity PACF")
```



For further confirmation of non-stationarity in the series, an Augmented Dicky-Fuller test is performed on the series. The absolute value of the test statistic is lower than the absolute of the critical values at 1%, 5% and 10% significance levels for all the series, suggesting that the series is stationary.

Further confirmation can be obtained using a Phillips-Perron unit root test in addition to the ADF

test. The Phillips-Perron test produces a Z-tau test statistic that is larger than the critical values at 1%, 5% and 10% significance levels for the Temperature, Rain, and Relative Humidity series, while the radiation series produces a Z-tau test statistic that is only more extreme at the 1% significance level. Since the RBO series is confirmed by both tests to be stationary, no transformations need to be carried out on the series.

```
adf.rbo = ur.df(rbo, type = "none", lags = 1, selectlags = "AIC")
summary(adf.rbo)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.082272 -0.019329  0.003007  0.022706  0.096349
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.002936   0.009292  -0.316  0.75449
## z.diff.lag -0.523248   0.163823  -3.194  0.00355 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03701 on 27 degrees of freedom
## Multiple R-squared:  0.2763, Adjusted R-squared:  0.2227
## F-statistic: 5.155 on 2 and 27 DF,  p-value: 0.0127
##
##
## Value of test-statistic is: -0.3159
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.62 -1.95 -1.61
```

```
pp.rbo = ur.pp(rbo, type = "Z-tau", lags = "short")
summary(pp.rbo)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.079686 -0.016885 -0.004797  0.021427  0.103989
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.3168     0.1158   2.735   0.0107 *
## y.l1          0.5692     0.1565   3.638   0.0011 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03795 on 28 degrees of freedom
## Multiple R-squared:  0.3209, Adjusted R-squared:  0.2967
## F-statistic: 13.23 on 1 and 28 DF,  p-value: 0.0011
##
##
## Value of test-statistic, type: Z-tau  is: -2.6249
##
##          aux. Z statistics
## Z-tau-mu            2.6059
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.665967 -2.962656 -2.620011
```

```
adf.temp = ur.df(temp, type = "none", lags = 1, selectlags = "AIC")
summary(adf.temp)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.74882 -0.24560 -0.06356  0.26178  0.91921
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## z.lag.1     0.003385   0.007510   0.451 0.655777
## z.diff.lag -0.594821   0.154274  -3.856 0.000648 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3829 on 27 degrees of freedom
## Multiple R-squared:  0.3554, Adjusted R-squared:  0.3076
## F-statistic: 7.443 on 2 and 27 DF,  p-value: 0.002663
##
##
## Value of test-statistic is: 0.4507
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.62 -1.95 -1.61
```

```
pp.temp = ur.pp(temp, type = "Z-tau", lags = "short")
summary(pp.temp)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8370 -0.2851  0.0566  0.2213  0.7546
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.0629     1.8594   3.798  0.00072 ***
## y.l1          0.2587     0.1958   1.321  0.19726
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3826 on 28 degrees of freedom
## Multiple R-squared:  0.05865,    Adjusted R-squared:  0.02503
## F-statistic: 1.745 on 1 and 28 DF,  p-value: 0.1973
##
##
## Value of test-statistic, type: Z-tau  is: -3.8554
##
##          aux. Z statistics
## Z-tau-mu            3.8677
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.665967 -2.962656 -2.620011
```

```
adf.rain = ur.df(rain, type = "none", lags = 1, selectlags = "AIC")
summary(adf.rain)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3753 -0.1936  0.1544  0.3340  0.8803
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.01662    0.03684  -0.451   0.6555
## z.diff.lag -0.36299    0.18035  -2.013   0.0542 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4736 on 27 degrees of freedom
## Multiple R-squared:  0.1428, Adjusted R-squared:  0.07929
## F-statistic: 2.249 on 2 and 27 DF,  p-value: 0.1249
##
##
## Value of test-statistic is: -0.4511
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.62 -1.95 -1.61
```

```
pp.rain = ur.pp(rain, type = "Z-tau", lags = "short")
summary(pp.rain)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0306 -0.1530  0.0554  0.2690  0.5381
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.0033     0.4491   4.461 0.000121 ***
## y.l1          0.1529     0.1868   0.818 0.420073
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3814 on 28 degrees of freedom
## Multiple R-squared:  0.02336,    Adjusted R-squared:  -0.01152
## F-statistic: 0.6697 on 1 and 28 DF,  p-value: 0.4201
##
##
## Value of test-statistic, type: Z-tau  is: -4.5072
##
##          aux. Z statistics
## Z-tau-mu            4.4335
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.665967 -2.962656 -2.620011
```

```
adf.radiation = ur.df(radiation, type = "none", lags = 1, selectlags = "AIC")
summary(adf.radiation)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.06546 -0.27928  0.06367  0.34134  0.64395
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.0007715  0.0054907  -0.141    0.889
## z.diff.lag -0.2325212  0.1870728  -1.243    0.225
##
## Residual standard error: 0.4312 on 27 degrees of freedom
## Multiple R-squared:  0.05456,    Adjusted R-squared:  -0.01548
## F-statistic: 0.779 on 2 and 27 DF,  p-value: 0.4689
##
##
## Value of test-statistic is: -0.1405
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.62 -1.95 -1.61
```

```
pp.radiation = ur.pp(radiation, type = "Z-tau", lags = "short")
summary(pp.radiation)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.10563 -0.09934  0.05929  0.19569  0.60439
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.0639     2.4425   3.302  0.00263 **
## y.l1          0.4467     0.1673   2.670  0.01249 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3705 on 28 degrees of freedom
## Multiple R-squared:  0.2029, Adjusted R-squared:  0.1745
## F-statistic: 7.128 on 1 and 28 DF,  p-value: 0.01249
##
##
## Value of test-statistic, type: Z-tau  is: -3.2491
##
##          aux. Z statistics
## Z-tau-mu             3.244
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.665967 -2.962656 -2.620011
```

```
adf.relhumid = ur.df(relhumid, type = "none", lags = 1, selectlags = "AIC")
summary(adf.relhumid)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7309 -0.7848  0.1133  0.6584  1.9218
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.0003151  0.0019849  -0.159    0.875
## z.diff.lag -0.2724937  0.1822047  -1.496    0.146
##
## Residual standard error: 1.011 on 27 degrees of freedom
## Multiple R-squared:  0.07733,    Adjusted R-squared:  0.008987
## F-statistic: 1.131 on 2 and 27 DF,  p-value: 0.3374
##
##
## Value of test-statistic is: -0.1587
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.62 -1.95 -1.61
```

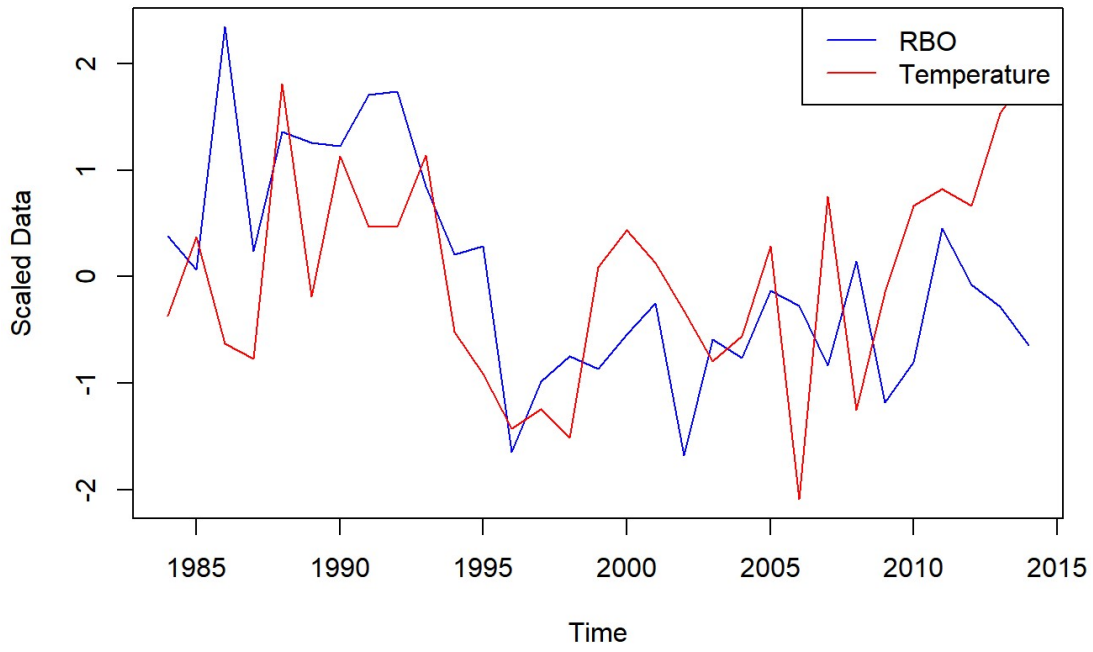
```
pp.relhumid = ur.pp(relhumid, type = "Z-tau", lags = "short")
summary(pp.relhumid)
```

```
##
## ###################################
## # Phillips-Perron Unit Root Test #
## ###################################
##
## Test regression with intercept
##
##
## Call:
## lm(formula = y ~ y.l1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.35103 -0.49219  0.02665  0.39864  1.65341
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  80.0768    17.5215   4.570 8.97e-05 ***
## y.l1          0.1532     0.1853   0.827    0.415
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7954 on 28 degrees of freedom
## Multiple R-squared:  0.02384,    Adjusted R-squared:  -0.01103
## F-statistic: 0.6837 on 1 and 28 DF,  p-value: 0.4153
##
##
## Value of test-statistic, type: Z-tau  is: -4.5348
##
##          aux. Z statistics
## Z-tau-mu            4.5352
##
## Critical values for Z statistics:
##                      1pct      5pct     10pct
## critical values -3.665967 -2.962656 -2.620011
```

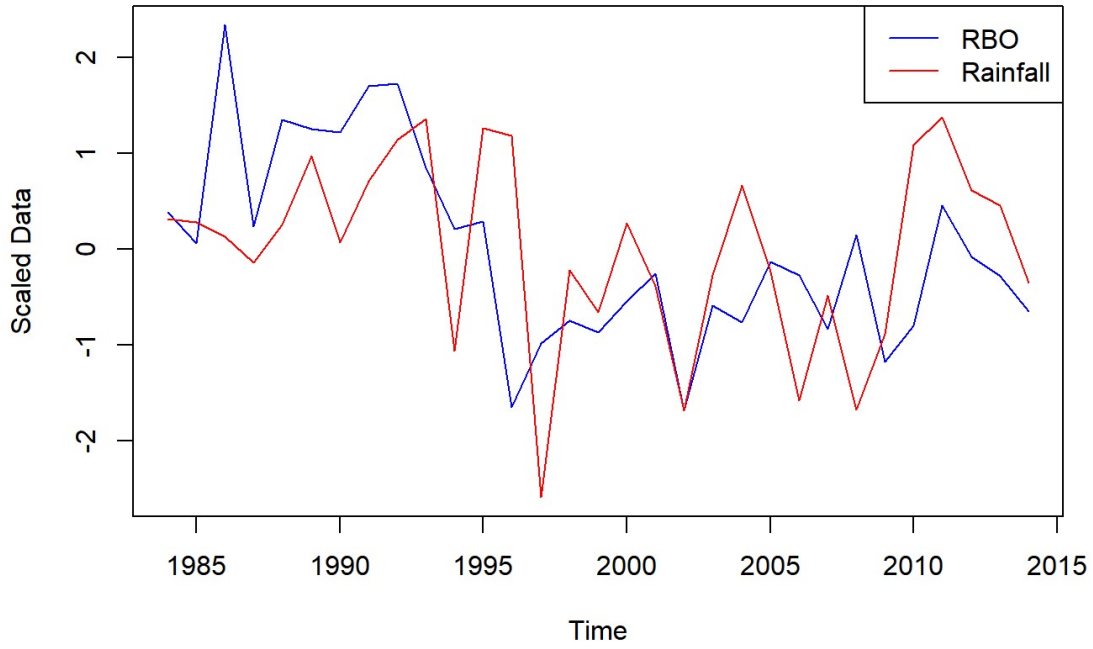
Correlating Predictor Variables with Dependent Variable

The scaled predictor series are plotted against RBO to visually identify any potential relationships between the climate conditions and RBO. Temperature and rainfall seem to follow the RBO series more closely than radiation and relative humidity.

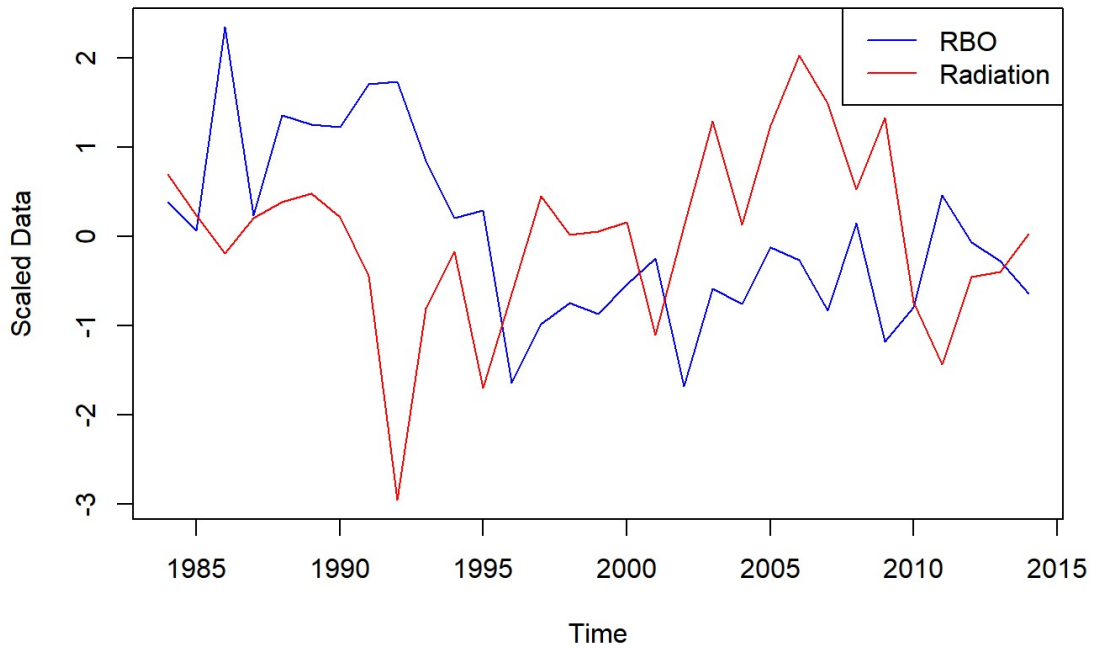
```
rbodata.ts <- ts(rbodata[,c(2,3)], start=c(1984,1), frequency=1)
rbodata_scaled = scale(rbodata.ts)
plot(rbodata_scaled, plot.type="s", col=c("blue", "red"), ylab="Scaled Data")
legend("topright", lty=1, col=c("blue", "red"), c("RBO", "Temperature"))
```



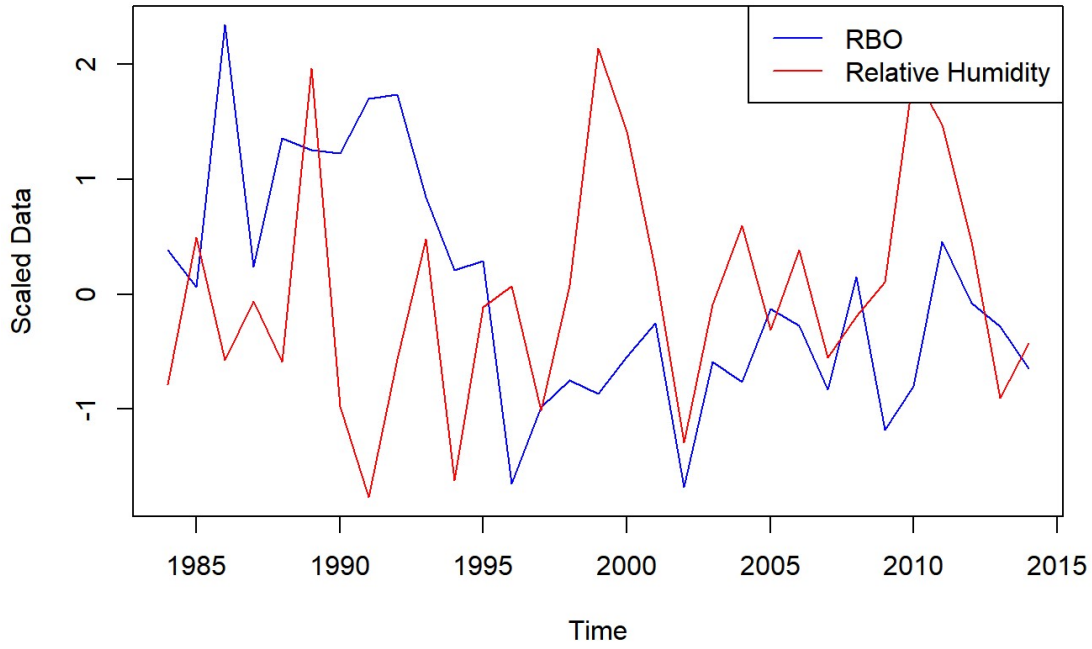

```
rbodata.ts <- ts(rbodata[,c(2,4)], start=c(1984,1), frequency=1)
rbodata_scaled = scale(rbodata.ts)
plot(rbodata_scaled, plot.type="s", col=c("blue", "red"), ylab="Scaled Data")
legend("topright", lty=1, col=c("blue", "red"), c("RBO", "Rainfall"))
```




```
rbodata.ts <- ts(rbodata[,c(2,5)], start=c(1984,1), frequency=1)
rbodata_scaled = scale(rbodata.ts)
plot(rbodata_scaled, plot.type="s", col=c("blue", "red"), ylab="Scaled Data")
legend("topright", lty=1, col=c("blue", "red"), c("RBO", "Radiation"))
```




```
rbodata.ts <- ts(rbodata[,c(2,6)], start=c(1984,1), frequency=1)
rbodata_scaled = scale(rbodata.ts)
plot(rbodata_scaled, plot.type="s", col=c("blue", "red"), ylab="Scaled Data")
legend("topright", lty=1, col=c("blue", "red"), c("RBO", "Relative Humidity"))
```



Since a visual confirmation of the correlation between predictor and dependent variables is insufficient, the variables can be correlated using the cor() function in order to obtain a numerical representation of correlation. Obtaining the correlation values shows that rainfall is the most positively correlated with RBO with a correlation of 0.39 and radiation is the most negatively correlated with a value of -0.32. However, none of the variables show particularly high correlations with RBO.

```
#correlate variables
rbodata.ts <- ts(rbodata[,2:6], start=c(1984,1), frequency=1)
cor(rbodata.ts)
```

```
##                    RBO  Temperature   Rainfall   Radiation  RelHumidity
## RBO          1.0000000  0.261000659  0.3932282 -0.31736018 -0.177634864
## Temperature  0.2610007  1.000000000  0.3933255 -0.24096625  0.009646021
## Rainfall     0.3932282  0.393325545  1.0000000 -0.58131610  0.338461007
## Radiation   -0.3173602 -0.240966245 -0.5813161  1.00000000 -0.055209652
## RelHumidity -0.1776349  0.009646021  0.3384610 -0.05520965  1.000000000
```

Fitting Distributed Lag Models

Polynomial DLM

Polynomial DLM models are first fitted using the polyDlm() function, using each of the predictor variables to predict RBO. Prior to fitting, the ideal lag value is found using the finiteDLMauto() function, which is made to generate multiple models with q ranging from 1 to 10 and uses AIC error to determine the suitability of each model. The results of finiteDLMauto() suggest that the ideal value for q is 3, and a polynomial order of 2.

```
#PolyDLM
finiteDLMauto(x = as.vector(temp), y = as.vector(rbo), q.min = 2, q.max = 10,
              model.type = "poly", error.type = "AIC", trace = TRUE)
```

```
##    q - k    MASE       AIC       BIC   GMRAE    MBRAE R.Adj.Sq   Ljung-Box
## 2  3 - 2 1.07250 -97.93803 -91.27700 1.21560  0.70726  0.18485 0.008026977
## 1  2 - 2 1.03256 -95.49894 -88.66246 0.99578  0.59614  0.15735 0.010959552
## 3  4 - 2 1.08914 -94.65533 -88.17614 1.07583  0.58883  0.22294 0.033266396
## 4  5 - 2 1.01455 -92.07915 -85.78867 1.05382  0.51307  0.21064 0.074676775
## 8  9 - 2 0.70029 -90.75952 -85.30431 0.71371  0.15795  0.10346 0.547063437
## 5  6 - 2 0.91305 -90.66426 -84.56988 0.86245  0.60059  0.23536 0.184576149
## 6  7 - 2 0.85502 -88.11262 -82.22235 0.75182 -0.33783  0.21783 0.339301381
## 9 10 - 2 0.67421 -87.92162 -82.69900 0.64483  0.33178 -0.00647 0.321859031
## 7  8 - 2 0.75509 -87.76319 -82.08572 0.60355  0.07005  0.15902 0.921975964
```

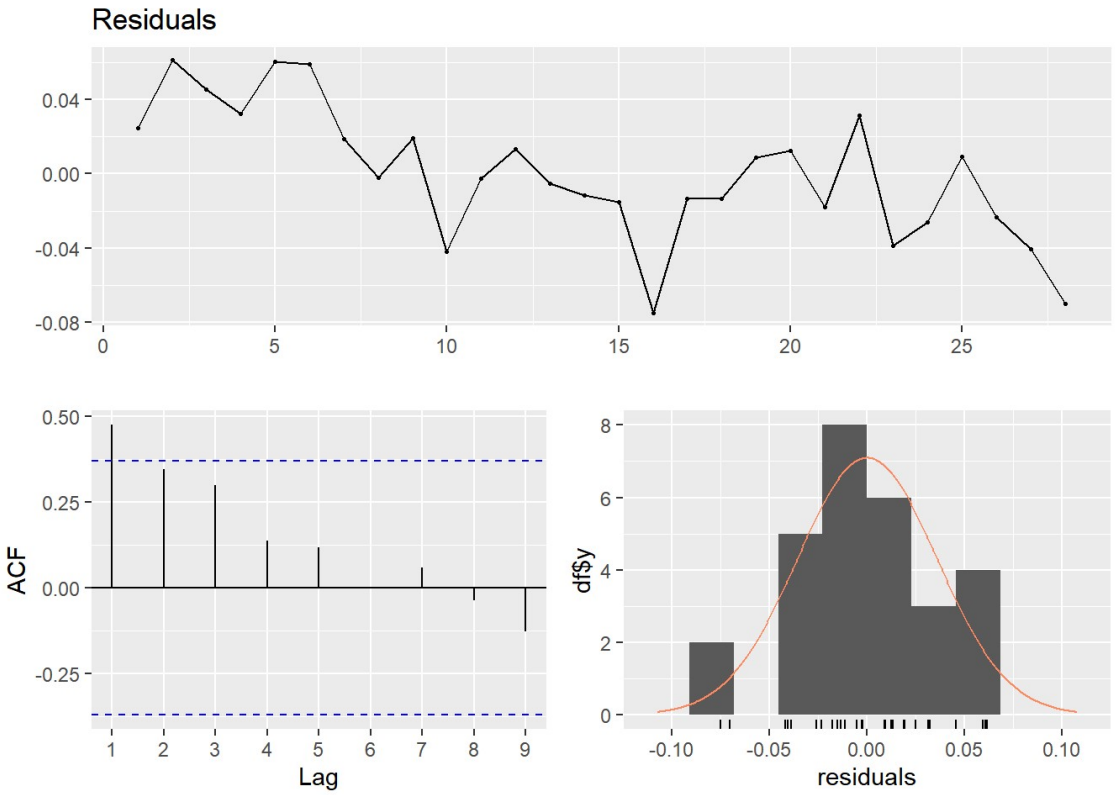
```
model.poly1 = polyDlm(x = as.vector(temp), y = as.vector(rbo), q = 3, k = 2, show.beta = TRUE)
```

```
## Estimates and t-tests for beta coefficients:
##        Estimate Std. Error t value P(>|t|)
## beta.0   0.0245     0.0201  1.2200  0.2330
## beta.1   0.0326     0.0154  2.1200  0.0444
## beta.2   0.0250     0.0162  1.5500  0.1350
## beta.3   0.0019     0.0221  0.0861  0.9320
```

```
summary(model.poly1)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.075139 -0.019311 -0.002455  0.020653  0.061476
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.063765   0.293609  -0.217    0.830
## z.t0         0.024535   0.020074   1.222    0.233
## z.t1         0.015802   0.039010   0.405    0.689
## z.t2        -0.007783   0.012572  -0.619    0.542
##
## Residual standard error: 0.03803 on 24 degrees of freedom
## Multiple R-squared:  0.2754, Adjusted R-squared:  0.1848
## F-statistic: 3.041 on 3 and 24 DF,  p-value: 0.04843
```

```
checkresiduals(model.poly1$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 7
##
## data:  Residuals
## LM test = 10.788, df = 7, p-value = 0.1481
```

```
MASE(model.poly1)
```

```
##              MASE
## model.poly1 1.0725
```

```
finiteDLMauto(x = as.vector(rain), y = as.vector(rbo), q.min = 2, q.max = 10,
              model.type = "poly", error.type = "AIC", trace = TRUE)
```

```
##    q - k    MASE       AIC       BIC   GMRAE    MBRAE R.Adj.Sq  Ljung-Box
## 2  3 - 2 0.94916 -98.84343 -92.18241 0.75174  0.19596  0.21078 0.01159322
## 1  2 - 2 0.99937 -96.70956 -89.87308 0.83164  0.44840  0.19180 0.03708415
## 3  4 - 2 1.00619 -94.04271 -87.56352 0.81974 -9.48452  0.20510 0.01870122
## 4  5 - 2 0.92642 -92.46231 -86.17182 0.87299  7.18996  0.22219 0.04046159
## 8  9 - 2 0.66465 -90.82864 -85.37342 0.54579  0.71453  0.10628 0.64416480
## 5  6 - 2 0.87732 -89.17565 -83.08127 0.78988  0.42199  0.18845 0.07236669
## 9 10 - 2 0.64518 -88.26525 -83.04264 0.54703  1.21414  0.00987 0.24089089
## 7  8 - 2 0.71953 -87.81723 -82.13976 0.55123 -3.16107  0.16100 0.73744124
## 6  7 - 2 0.83196 -86.84712 -80.95686 0.75983  0.73645  0.17548 0.21710348
```

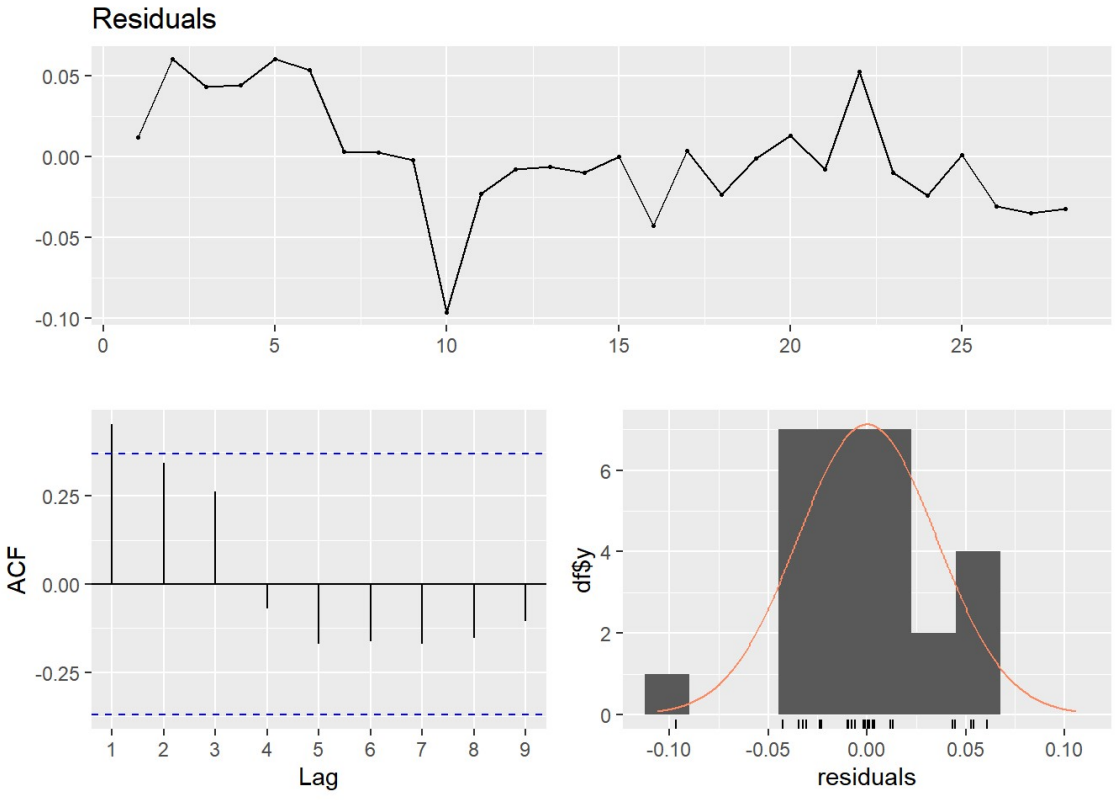
```
model.poly2 = polyDlm(x = as.vector(rain), y = as.vector(rbo), q = 3, k = 2, show.beta = TRUE)
```

```
## Estimates and t-tests for beta coefficients:
##        Estimate Std. Error t value P(>|t|)
## beta.0  0.03920     0.0174   2.260  0.0331
## beta.1  0.03110     0.0129   2.420  0.0232
## beta.2  0.01980     0.0128   1.540  0.1350
## beta.3  0.00534     0.0175   0.304  0.7640
```

```
summary(model.poly2)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.096663 -0.023215 -0.001439  0.012290  0.060753
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.507827   0.078501   6.469 1.09e-06 ***
## z.t0         0.039221   0.017386   2.256   0.0335 *
## z.t1        -0.006539   0.027911  -0.234   0.8168
## z.t2        -0.001585   0.008902  -0.178   0.8602
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03742 on 24 degrees of freedom
## Multiple R-squared:  0.2985, Adjusted R-squared:  0.2108
## F-statistic: 3.404 on 3 and 24 DF,  p-value: 0.03391
```

```
checkresiduals(model.poly2$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 7
##
## data:  Residuals
## LM test = 11.234, df = 7, p-value = 0.1287
```

```
MASE(model.poly2)
```

```
##                  MASE
## model.poly2 0.9491588
```

```
finiteDLMauto(x = as.vector(radiation), y = as.vector(rbo), q.min = 2, q.max = 10,
              model.type = "poly", error.type = "AIC", trace = TRUE)
```

```
##    q - k    MASE       AIC       BIC   GMRAE    MBRAE R.Adj.Sq    Ljung-Box
## 2  3 - 2 1.14015 -93.41298 -86.75196 1.29140 -1.52895  0.04187 0.0006244607
## 1  2 - 2 1.17471 -91.50708 -84.67061 1.10279  2.19244  0.03299 0.0024745598
## 8  9 - 2 0.69743 -89.93849 -84.48328 0.62884  0.21382  0.06937 0.9853434812
## 3  4 - 2 1.20364 -89.71544 -83.23625 1.37757 -1.52136  0.06693 0.0007961637
## 7  8 - 2 0.72595 -89.08630 -83.40883 0.49184  0.19854  0.20604 0.3642897864
## 4  5 - 2 1.08508 -88.46836 -82.17788 1.17879  0.08589  0.09304 0.0033744550
## 9 10 - 2 0.67120 -88.15084 -82.92823 0.61521  0.53618  0.00446 0.4225785008
## 5  6 - 2 0.95730 -87.48527 -81.39089 0.90285 -0.76340  0.13168 0.0140139215
## 6  7 - 2 0.84943 -86.62398 -80.73372 0.65591  0.26051  0.16778 0.0990790188
```

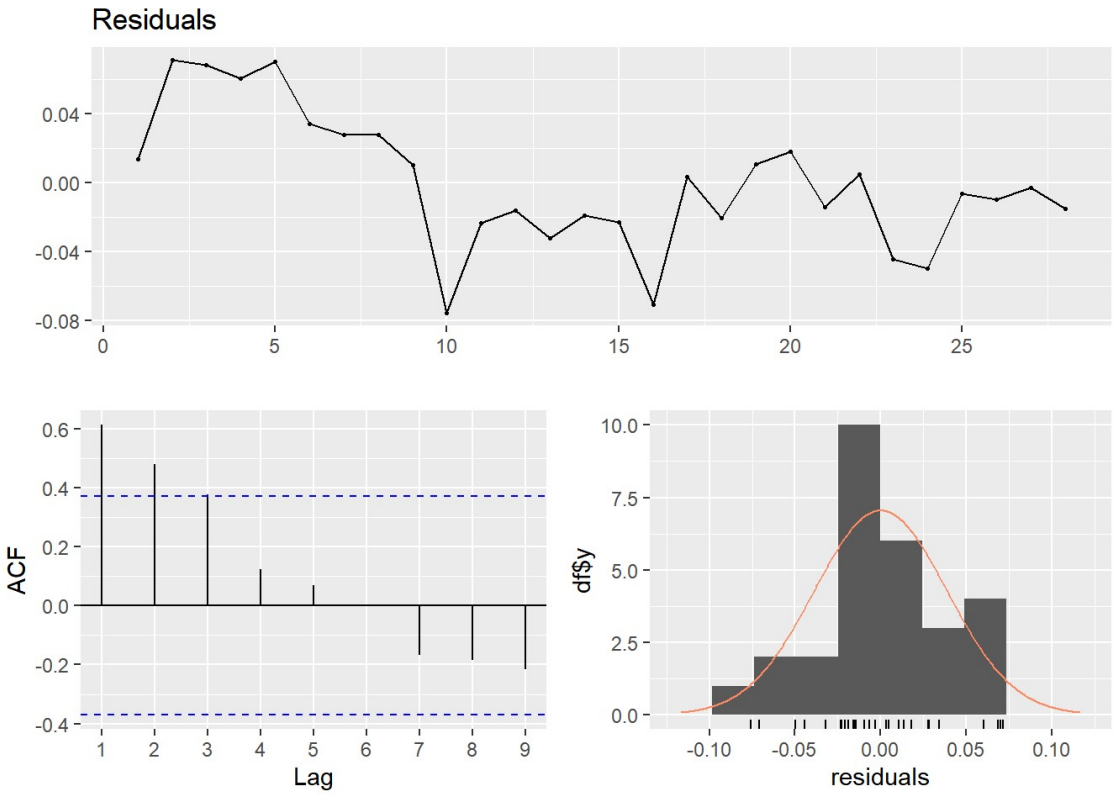
```
model.poly3 = polyDlm(x = as.vector(radiation), y = as.vector(rbo), q = 3, k = 2, show.beta = TRUE)
```

```
## Estimates and t-tests for beta coefficients:
##        Estimate Std. Error t value P(>|t|)
## beta.0 -0.03690     0.0189  -1.960  0.0614
## beta.1 -0.00336     0.0131  -0.256  0.8000
## beta.2  0.01280     0.0132   0.976  0.3390
## beta.3  0.01170     0.0188   0.624  0.5390
```

```
summary(model.poly3)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.075731 -0.021146 -0.004749  0.020568  0.071458
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.962500   0.391271   2.460   0.0215 *
## z.t0        -0.036924   0.018850  -1.959   0.0619 .
## z.t1         0.042244   0.033854   1.248   0.2241
## z.t2        -0.008679   0.010868  -0.799   0.4324
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04123 on 24 degrees of freedom
## Multiple R-squared:  0.1483, Adjusted R-squared:  0.04187
## F-statistic: 1.393 on 3 and 24 DF,  p-value: 0.269
```

```
checkresiduals(model.poly3$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 7
##
## data:  Residuals
## LM test = 13.608, df = 7, p-value = 0.05862
```

```
MASE(model.poly3)
```

```
##                MASE
## model.poly3 1.140154
```

```
finiteDLMauto(x = as.vector(relhumid), y = as.vector(rbo), q.min = 2, q.max = 10,
              model.type = "poly", error.type = "AIC", trace = TRUE)
```

```
##    q - k    MASE       AIC       BIC   GMRAE    MBRAE R.Adj.Sq    Ljung-Box
## 2  3 - 2 1.20544 -90.71527 -84.05425 1.35246  0.26853 -0.05504 0.0005634652
## 1  2 - 2 1.23588 -88.37920 -81.54272 1.29446  0.39157 -0.07714 0.0043469395
## 3  4 - 2 1.26502 -86.21246 -79.73328 1.16416  0.95992 -0.06233 0.0006363657
## 9 10 - 2 0.68219 -85.61489 -80.39227 0.53165  0.28527 -0.12333 0.7192581192
## 8  9 - 2 0.77265 -85.27436 -79.81915 0.65716 -0.55774 -0.15040 0.6878894730
## 4  5 - 2 1.21844 -83.61100 -77.32052 1.08023 -0.16008 -0.09326 0.0017585153
## 5  6 - 2 1.12479 -82.49202 -76.39764 1.15595  2.67273 -0.06029 0.0083593631
## 7  8 - 2 0.88272 -81.50599 -75.82852 0.82113  0.08551 -0.10391 0.1778166984
## 6  7 - 2 1.03932 -80.66896 -74.77869 0.97954  1.94371 -0.06659 0.0423819721
```

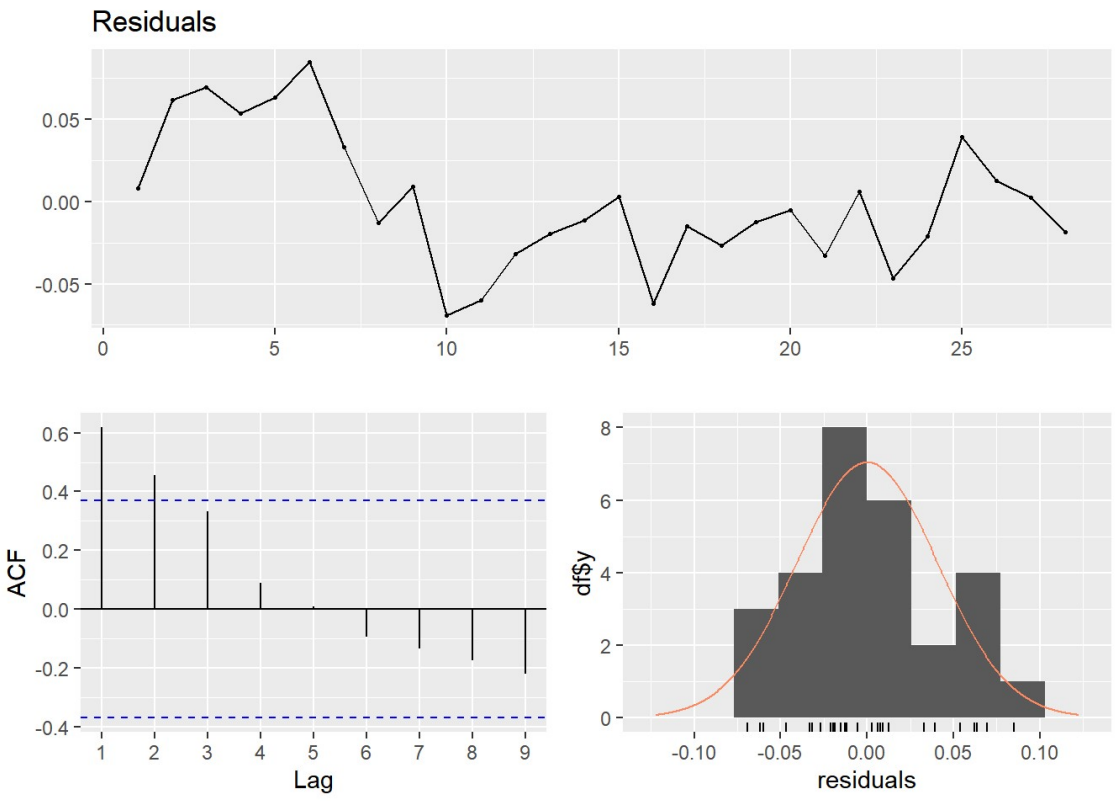
```
model.poly4 = polyDlm(x = as.vector(relhumid), y = as.vector(rbo), q = 3, k = 2, show.beta = TRUE)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value P(>|t|)
## beta.0 -0.00998    0.00986  -1.010   0.321
## beta.1 -0.00296    0.00705  -0.420   0.678
## beta.2 -0.00274    0.00706  -0.388   0.702
## beta.3 -0.00932    0.00993  -0.938   0.357
```

```
summary(model.poly4)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.069356 -0.022466 -0.008493  0.017672  0.084900
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.097339   2.037675   1.520    0.142
## z.t0        -0.009983   0.009861  -1.012    0.321
## z.t1         0.010427   0.015685   0.665    0.513
## z.t2        -0.003402   0.005068  -0.671    0.508
##
## Residual standard error: 0.04327 on 24 degrees of freedom
## Multiple R-squared:  0.06219,    Adjusted R-squared:  -0.05504
## F-statistic: 0.5305 on 3 and 24 DF,  p-value: 0.6657
```

```
checkresiduals(model.poly4$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 7
##
## data:  Residuals
## LM test = 14.437, df = 7, p-value = 0.04394
```

```
MASE(model.poly4)
```

```
##                 MASE
## model.poly4 1.205438
```

Polynomial Model Summary

Model	MASE	Model p-value	Model R-squared value	Breusch-Godfrey p-value
Temperature	1.0725	0.04843	0.1848	0.1481
Rain	0.949	0.03391	0.2108	0.03878
Radiation	1.14	0.269	0.04187	0.05862
Relative Humidity	1.205	0.6657	-0.05504	0.04394

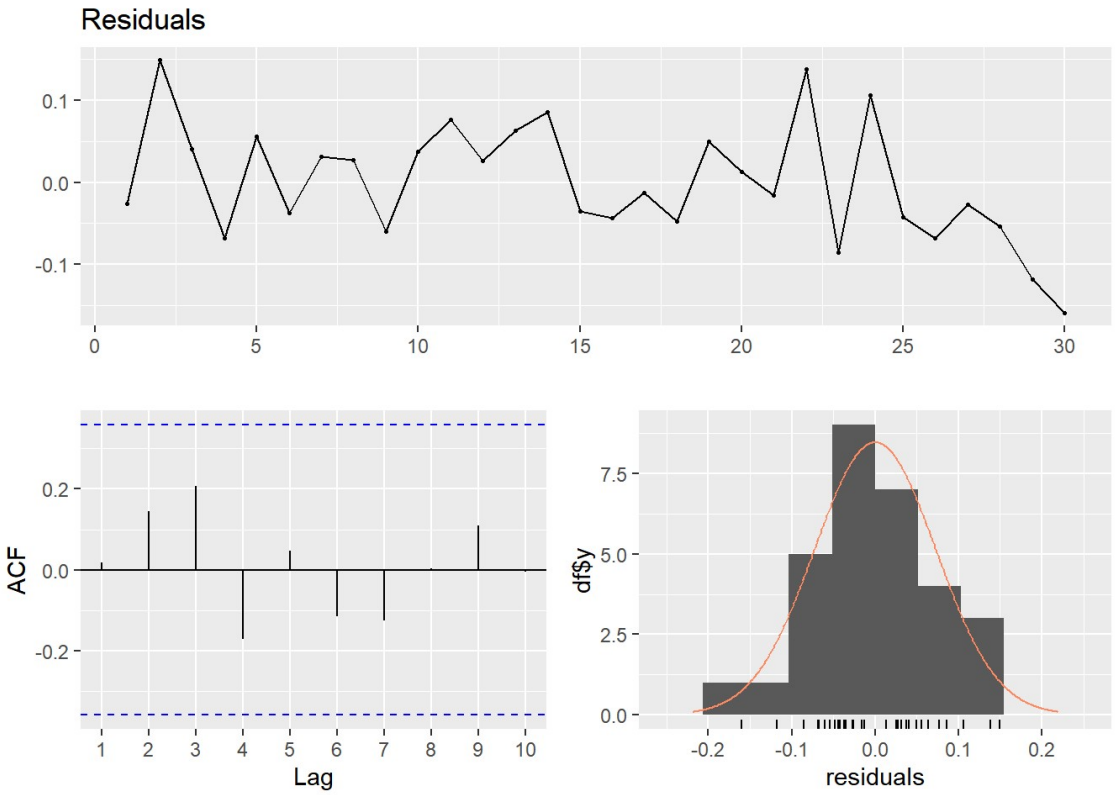
Only the temperature and rainfall predictors produce polynomial DLM models that are significant, with p-values below 0.05. Of these, rainfall has the lowest MASE and the highest R-squared value. However, none of the models have high R-squared values, implying that none of the models describe the data very well. The Breusch-Godfrey tests reveal that rainfall and relative humidity produce models with serial correlation due to p-values lower than 0.05.

Koyck (Geometric) DLM

```
# Kocyk
model.Koyck1 = koyckDlm(x = as.vector(temp), y = as.vector(rbo), intercept = TRUE)
summary(model.Koyck1)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.15981 -0.04678 -0.01440  0.04750  0.14952
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.2032     1.6184  -0.743    0.464
## Y.1           0.2469     0.4609   0.536    0.597
## X.t           0.1847     0.1947   0.949    0.351
##
## Residual standard error: 0.07557 on 27 degrees of freedom
## Multiple R-Squared: -1.597,  Adjusted R-squared: -1.789
## Wald test: 2.119 on 2 and 27 DF,  p-value: 0.1397
##
## Diagnostic tests:
## NULL
##
##                              alpha      beta      phi
## Geometric coefficients:  -1.597614 0.1847273 0.246894
```

```
checkresiduals(model.Koyck1$model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 3.9579, df = 6, p-value = 0.6824
##
## Model df: 0.   Total lags used: 6
```

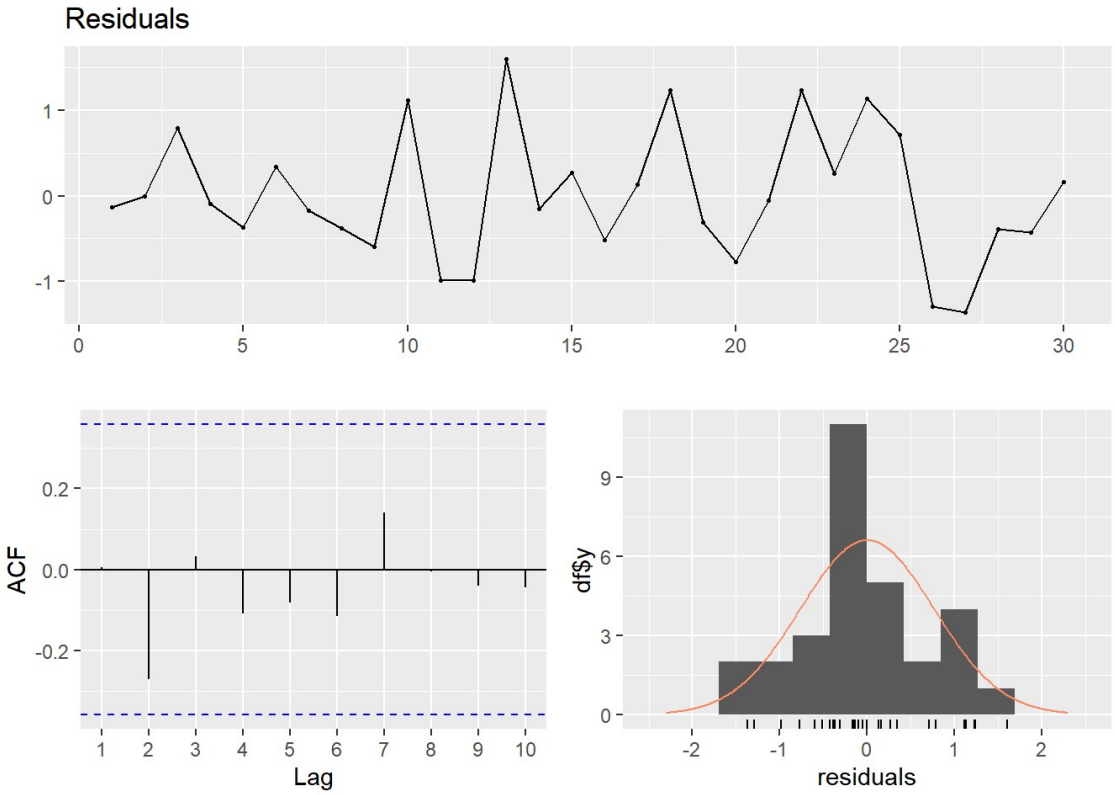
```
MASE(model.Koyck1)
```

```
##                  MASE
## model.Koyck1 1.915015
```

```
model.Koyck2 = koyckDlm(x = as.vector(rain), y = as.vector(rbo), intercept = TRUE)
summary(model.Koyck2)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3665 -0.4155 -0.1142  0.3241  1.6012
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.3207     2.4302   0.132    0.896
## Y.1          -6.5147   243.8216  -0.027    0.979
## X.t           2.2101    76.0635   0.029    0.977
##
## Residual standard error: 0.7951 on 27 degrees of freedom
## Multiple R-Squared: -286.5,  Adjusted R-squared: -307.8
## Wald test: 0.01549 on 2 and 27 DF,  p-value: 0.9846
##
## Diagnostic tests:
## NULL
##
##                               alpha    beta       phi
## Geometric coefficients:  0.04267914 2.21011 -6.514689
```

```
checkresiduals(model.Koyck2$model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 3.7654, df = 6, p-value = 0.7084
##
## Model df: 0.   Total lags used: 6
```

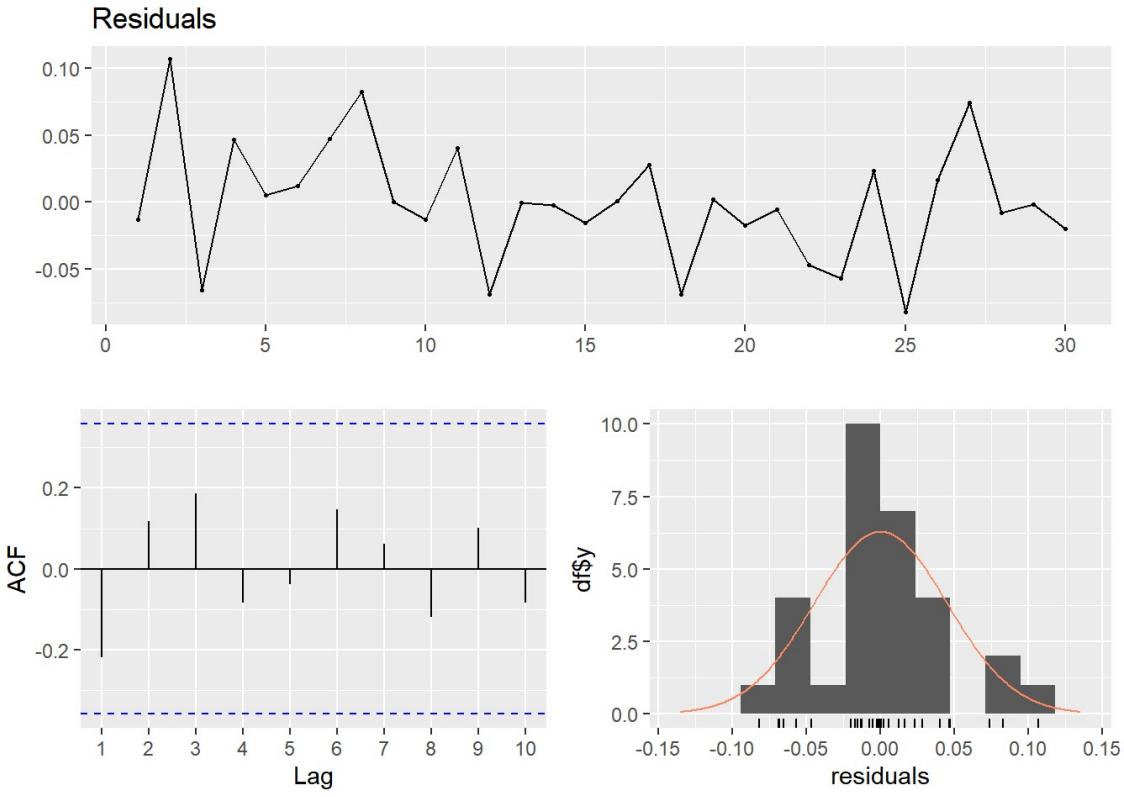
```
MASE(model.Koyck2)
```

```
##                   MASE
## model.Koyck2 19.10576
```

```
model.Koyck3 = koyckDlm(x = as.vector(radiation), y = as.vector(rbo), intercept = TRUE)
summary(model.Koyck3)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.082255 -0.017008 -0.001036  0.021424  0.106984
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.48011    0.94819  -0.506   0.6167
## Y.1          0.69801    0.24502   2.849   0.0083 **
## X.t          0.04812    0.05661   0.850   0.4028
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0467 on 27 degrees of freedom
## Multiple R-Squared: 0.008467,    Adjusted R-squared: -0.06498
## Wald test: 4.731 on 2 and 27 DF,  p-value: 0.01732
##
## Diagnostic tests:
## NULL
##
##                              alpha       beta       phi
## Geometric coefficients:  -1.589802 0.04811971 0.6980071
```

```
checkresiduals(model.Koyck3$model)
```



```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 4.4493, df = 6, p-value = 0.6161
##
## Model df: 0.   Total lags used: 6
```

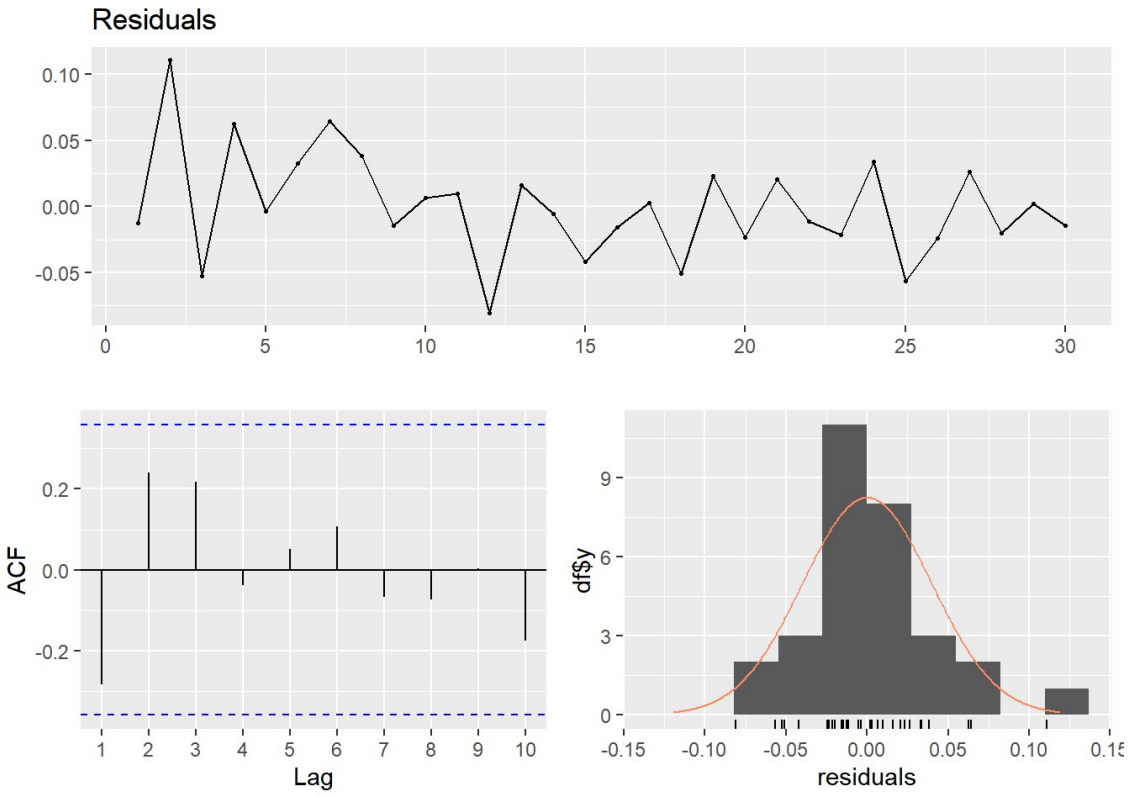
```
MASE(model.Koyck3)
```

```
##                  MASE
## model.Koyck3 1.031423
```

```
model.Koyck4 = koyckDlm(x = as.vector(relhumid), y = as.vector(rbo), intercept = TRUE)
summary(model.Koyck4)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.080897 -0.021103 -0.004676  0.022673  0.111041
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.16679    8.04941  -0.145   0.8858
## Y.1          0.62503    0.34753   1.798   0.0833 .
## X.t          0.01525    0.08274   0.184   0.8551
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04127 on 27 degrees of freedom
## Multiple R-Squared: 0.2256,  Adjusted R-squared: 0.1682
## Wald test: 5.612 on 2 and 27 DF,  p-value: 0.009161
##
## Diagnostic tests:
## NULL
##
##                              alpha       beta       phi
## Geometric coefficients:  -3.111733 0.01525207 0.6250339
```

```
checkresiduals(model.Koyck4$model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 6.9394, df = 6, p-value = 0.3265
##
## Model df: 0.   Total lags used: 6
```

```
MASE(model.Koyck4)
```

```
##                   MASE
## model.Koyck4 0.9559618
```

Koyck Model Summary

Model	MASE	Model p-value	Model R-squared value	Ljung-Box p-value
Temperature	1.915	0.1397	-1.789	0.6824
Rain	19.106	0.9846	-307.8	0.05953
Radiation	1.031	0.01732	-0.06498	0.6161
Relative Humidity	0.956	0.009161	0.1682	0.3265

Koyck DLM models are fitted using the koyckDlm() function, and their summaries, residuals, and MASE are reported in the table above. Only radiation and relative humidity produce Koyck models that are significant due to their lower p-values. Of these, relative humidity has the lowest MASE and the highest R-squared value, making it the best model of the four. However, an R-squared value of 0.1682 does not suggest that the model explains the data well. All the Koyck models have no serial correlation due to high p-values obtained from the Ljung-Box tests.

Finite DLM

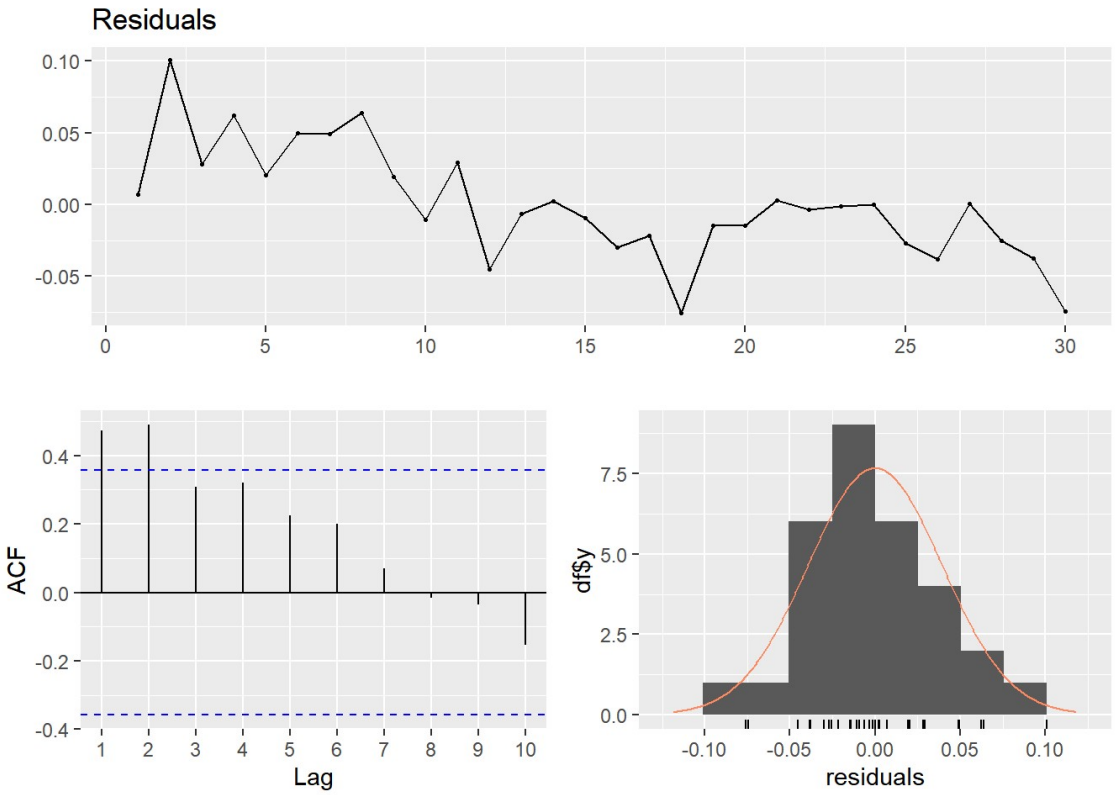
```
# Finite DLM
finiteDLMauto(x = as.vector(temp), y = as.vector(rbo), q.min = 1, q.max = 8,
              model.type = "dlm", error.type = "AIC", trace = TRUE)
```

```
##   q - k    MASE        AIC       BIC   GMRAE    MBRAE R.Adj.Sq   Ljung-Box
## 1     1 0.92390 -101.86168 -96.25690 0.76768  0.53945  0.18416 0.006390220
## 3     3 1.03366  -96.76727 -88.77404 1.08845  1.18155  0.17423 0.004429978
## 2     2 1.03256  -95.49894 -88.66246 0.99578  0.59614  0.15735 0.010959552
## 4     4 1.00537  -92.75653 -83.68567 0.88387 -1.10388  0.21265 0.019611693
## 5     5 0.85942  -91.46337 -81.39860 0.87678  0.21656  0.25697 0.039965557
## 6     6 0.81034  -85.74127 -74.77139 0.60702  0.16991  0.16483 0.074004639
## 8     8 0.64976  -83.28717 -70.79674 0.57103  0.60701  0.11382 0.991762977
## 7     7 0.75190  -82.04015 -70.25962 0.64261 -0.13363  0.11454 0.217224071
```

```
model.dlm1 = dlm(x = as.vector(temp), y = as.vector(rbo), q = 1)
summary(model.dlm1)
```

```
##
## Call:
## lm(formula = model.formula, data = design)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.075797 -0.024417 -0.002201  0.020077  0.100856
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.05297    0.24450   0.217    0.830
## x.t          0.01922    0.02019   0.952    0.350
## x.1          0.05285    0.02156   2.451    0.021 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04087 on 27 degrees of freedom
## Multiple R-squared:  0.2404, Adjusted R-squared:  0.1842
## F-statistic: 4.273 on 2 and 27 DF,  p-value: 0.02442
##
## AIC and BIC values for the model:
##         AIC      BIC
## 1 -101.8617 -96.2569
```

```
checkresiduals(model.dlm1$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 6
##
## data:  Residuals
## LM test = 11.738, df = 6, p-value = 0.06806
```

```
MASE(model.dlm1)
```

```
##                MASE
## model.dlm1 0.9239038
```

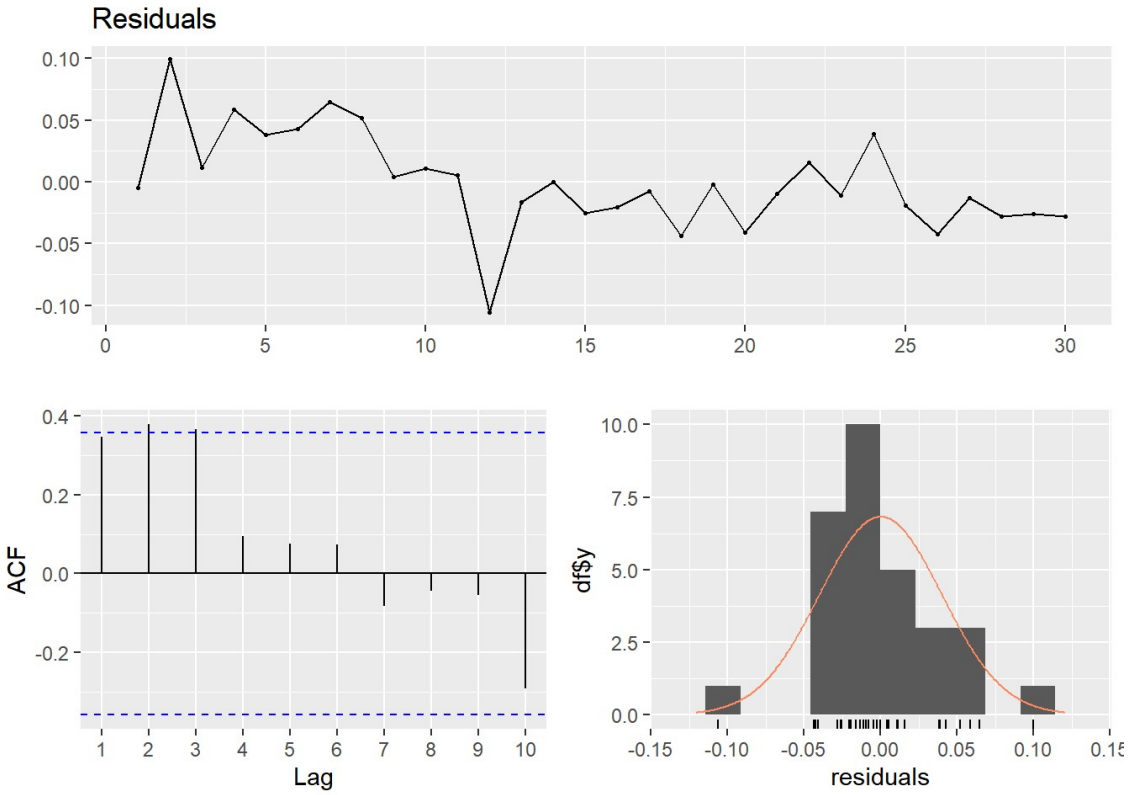
```
finiteDLMauto(x = as.vector(rain), y = as.vector(rbo), q.min = 1, q.max = 8,
              model.type = "dlm", error.type = "AIC", trace = TRUE)
```

```
##   q - k    MASE        AIC       BIC   GMRAE    MBRAE R.Adj.Sq  Ljung-Box
## 1     1 0.94180 -100.89798 -95.29319 0.94415  0.46484  0.15753 0.04687413
## 3     3 0.97969  -97.19966 -89.20643 0.86453  0.97573  0.18688 0.01249933
## 2     2 0.99937  -96.70956 -89.87308 0.83164  0.44840  0.19180 0.03708415
## 4     4 1.03883  -90.46187 -81.39101 0.86444  0.78209  0.14281 0.02073547
## 5     5 0.92568  -87.24242 -77.17765 0.76943  0.47689  0.12600 0.02871866
## 6     6 0.85440  -82.31788 -71.34800 0.51455 27.53976  0.04227 0.04784411
## 7     7 0.82934  -77.98405 -66.20351 0.77764  0.94056 -0.04849 0.17026081
## 8     8 0.72338  -76.81922 -64.32879 0.62316  0.02713 -0.17396 0.59232039
```

```
model.dlm2 = dlm(x = as.vector(rain), y = as.vector(rbo), q = 1)
summary(model.dlm2)
```

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

```
checkresiduals(model.dlm2$model)
```




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

```
MASE(model.dlm2)
```

```
##                MASE
## model.dlm2 0.9417954
```

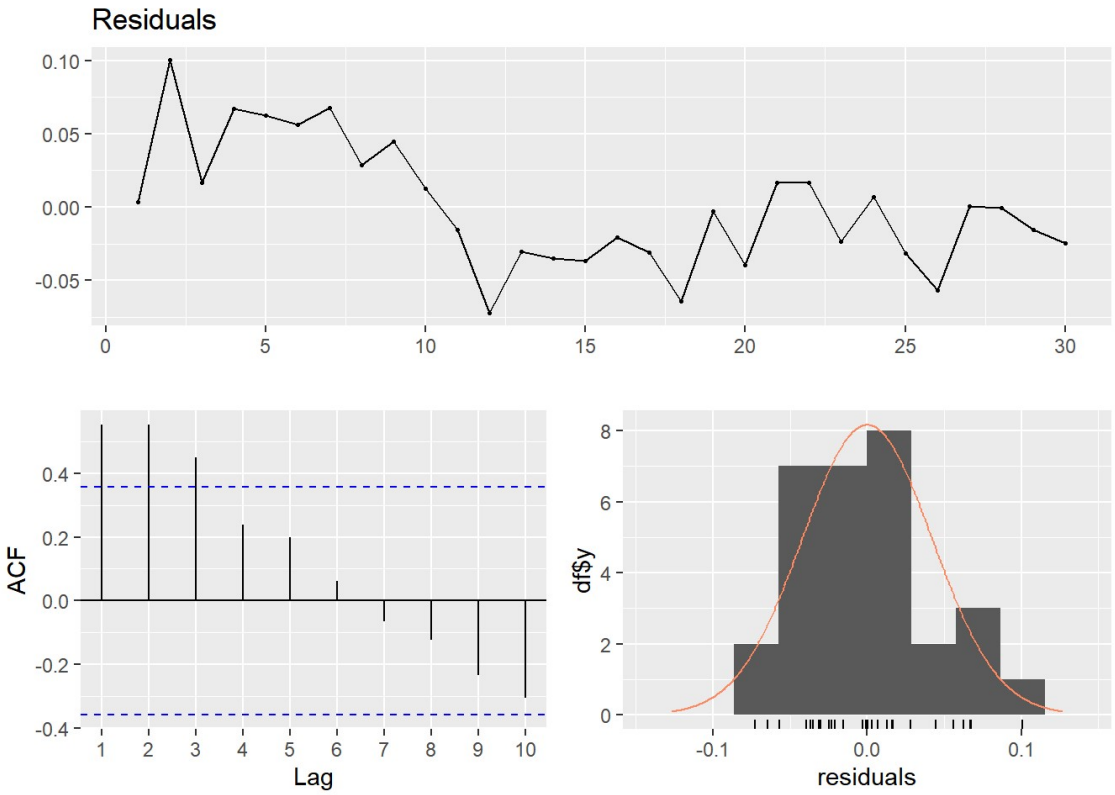
```
finiteDLMauto(x = as.vector(radiation), y = as.vector(rbo), q.min = 1, q.max = 8,
              model.type = "dlm", error.type = "AIC", trace = TRUE)
```

```
##   q - k    MASE       AIC       BIC   GMRAE    MBRAE R.Adj.Sq    Ljung-Box
## 1     1 1.06303 -97.71113 -92.10634 1.18033  2.85932  0.06311 0.0015201563
## 3     3 1.12846 -92.38658 -84.39335 1.15033  0.64125  0.03438 0.0005032338
## 2     2 1.17471 -91.50708 -84.67061 1.10279  2.19244  0.03299 0.0024745598
## 4     4 1.19981 -86.43017 -77.35931 1.27178 -0.09837  0.00476 0.0007267546
## 5     5 1.07186 -83.32223 -73.25746 1.15984  2.21790 -0.01624 0.0036551584
## 6     6 0.94832 -81.85403 -70.88414 0.86545 -0.85749  0.02433 0.0134482398
## 8     8 0.70733 -80.65686 -68.16642 0.59509  0.78620  0.00645 0.2032244911
## 7     7 0.85847 -79.04997 -67.26943 0.67378  1.85251 -0.00294 0.0831448135
```

```
model.dlm3 = dlm(x = as.vector(radiation), y = as.vector(rbo), q = 1)
summary(model.dlm3)
```

```
##
## Call:
## lm(formula = model.formula, data = design)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.072417 -0.030826 -0.001718  0.016681  0.100514
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.13698    0.34038   3.340  0.00246 **
## x.t         -0.04431    0.02234  -1.983  0.05764 .
## x.1          0.01689    0.02216   0.762  0.45261
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0438 on 27 degrees of freedom
## Multiple R-squared:  0.1277, Adjusted R-squared:  0.06311
## F-statistic: 1.977 on 2 and 27 DF,  p-value: 0.1581
##
## AIC and BIC values for the model:
##         AIC       BIC
## 1 -97.71113 -92.10634
```

```
checkresiduals(model.dlm3$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 6
##
## data:  Residuals
## LM test = 13.502, df = 6, p-value = 0.03573
```

```
MASE(model.dlm3)
```

```
##               MASE
## model.dlm3 1.063029
```

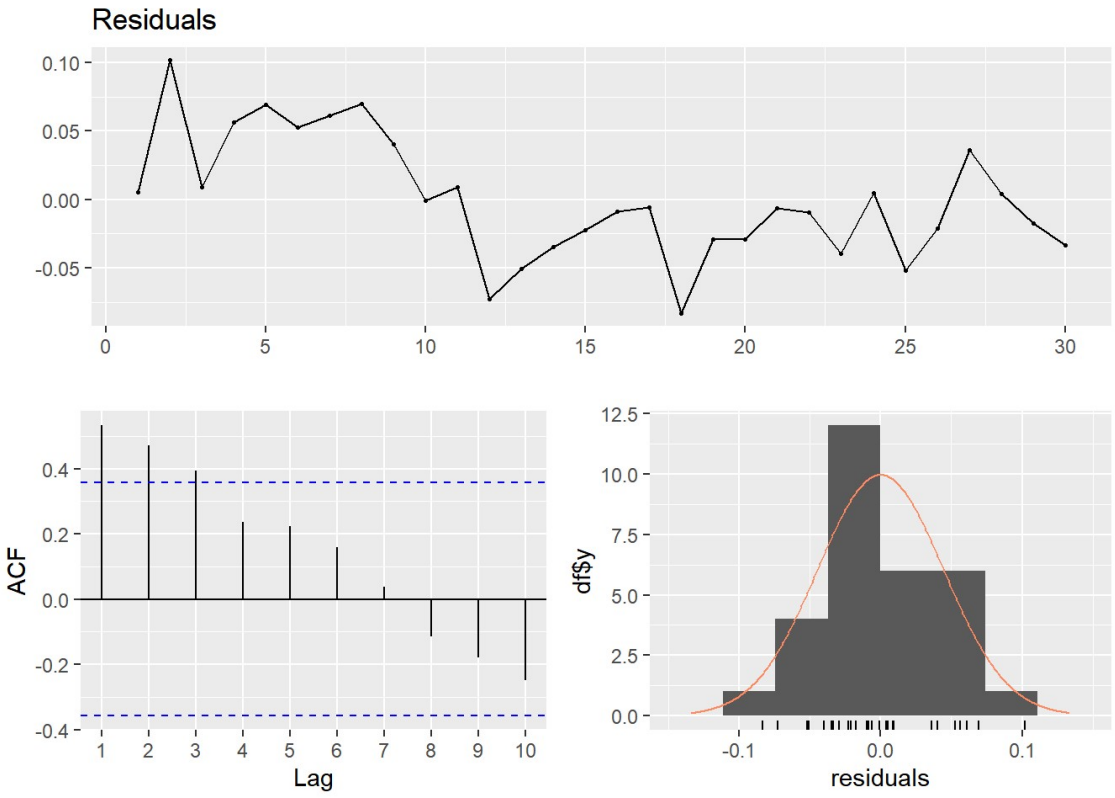
```
finiteDLMauto(x = as.vector(relhumid), y = as.vector(rbo), q.min = 1, q.max = 8,
              model.type = "dlm", error.type = "AIC", trace = TRUE)
```

```
##   q - k    MASE       AIC       BIC   GMRAE    MBRAE R.Adj.Sq    Ljung-Box
## 1     1 1.10110 -94.56619 -88.96140 1.22180  2.01650 -0.04044 0.0021897031
## 3     3 1.17112 -89.19540 -81.20218 1.12421  0.09980 -0.08219 0.0002775153
## 2     2 1.23588 -88.37920 -81.54272 1.29446  0.39157 -0.07714 0.0043469395
## 4     4 1.26243 -82.80086 -73.73000 1.31804  1.18794 -0.13842 0.0002642674
## 5     5 1.14964 -79.40410 -69.33932 1.00804  0.38188 -0.18152 0.0004056958
## 6     6 1.02674 -76.22147 -65.25158 0.82281  0.36858 -0.22222 0.0061384348
## 7     7 0.92355 -74.80399 -63.02345 0.77382  0.64955 -0.19704 0.0043529149
## 8     8 0.84661 -72.65628 -60.16585 0.79021 -4.31457 -0.40689 0.0443006703
```

```
model.dlm4 = dlm(x = as.vector(relhumid), y = as.vector(rbo), q = 1)
summary(model.dlm4)
```

```
##
## Call:
## lm(formula = model.formula, data = design)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.083453 -0.029388 -0.006143  0.029204  0.101854
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.887511   1.343538   1.405    0.171
## x.t         -0.009241   0.010967  -0.843    0.407
## x.1         -0.002923   0.010884  -0.269    0.790
##
## Residual standard error: 0.04616 on 27 degrees of freedom
## Multiple R-squared:  0.03131,    Adjusted R-squared:  -0.04044
## F-statistic: 0.4364 on 2 and 27 DF,  p-value: 0.6509
##
## AIC and BIC values for the model:
##         AIC      BIC
## 1 -94.56619 -88.9614
```

```
checkresiduals(model.dlm4$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 6
##
## data:  Residuals
## LM test = 13.056, df = 6, p-value = 0.04215
```

```
MASE(model.dlm4)
```

```
##                MASE
## model.dlm4 1.101099
```

Finite DLM Model Summary

Model	MASE	Model p-value	Model R-squared value	Breusch-Godfrey p-value
Temperature	0.924	0.02442	0.1842	0.06806
Rain	0.942	0.03767	0.1575	0.1391
Radiation	1.063	0.3042	0.06311	0.03573
Relative Humidity	1.101	0.6509	-0.04044	0.04215

Multiple finite DLM models are fitted using the dlm() function. Prior to fitting, the ideal lag value is found using the finiteDLMauto() function, which is made to generate multiple models with q ranging from 1 to 8 and uses AIC error to determine the suitability of each model. The results of finiteDLMauto() suggest that the ideal value for q is 1.

Temperature and rainfall produce finite DLM models that are significant due to their relatively low p-values and relatively high R-squared values. Of these, the model produced by temperature has the lowest MASE and highest R-squared value, making it the most suitable from the finite DLM models.

ARDL Model

ARDL models are created for each predictor and their summaries, residuals, and MASE are analysed below:

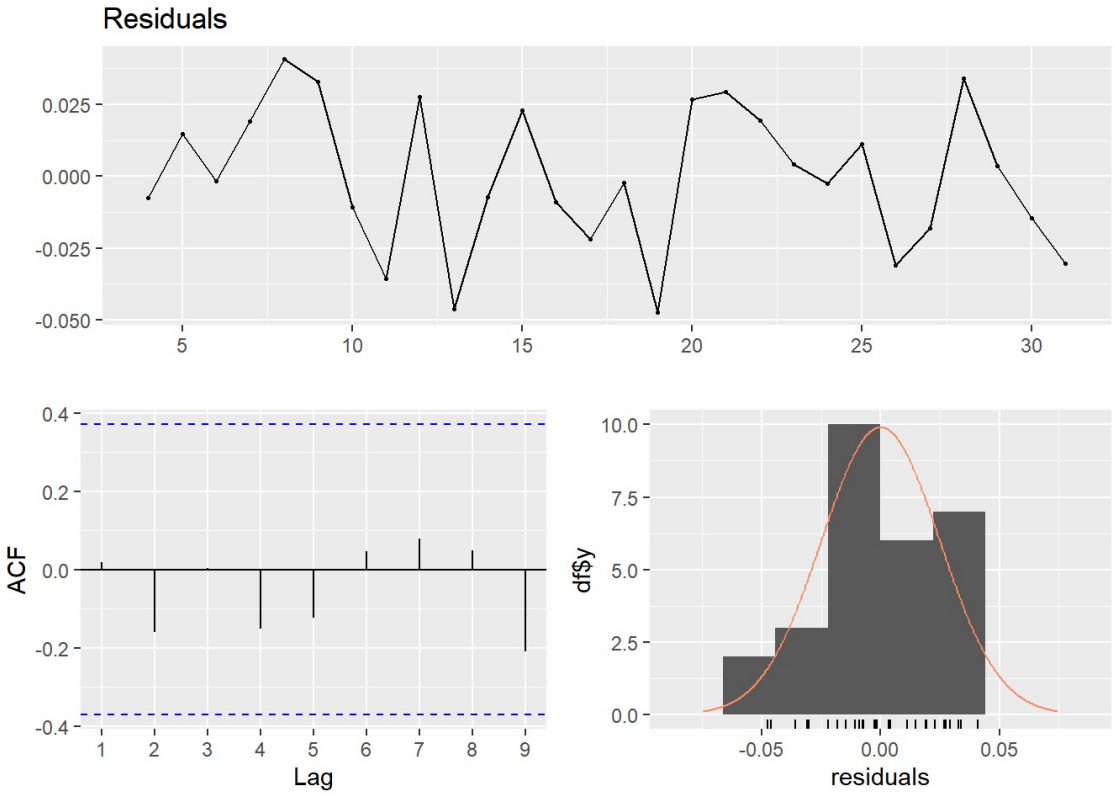
```
#ARDL

#temp
columns = c("p","q","AIC","BIC")
df = data.frame(matrix(nrow = 0, ncol = length(columns)))
colnames(df) = columns

for(i in 1:5){
  for(j in 1:5){
    model.ARDL = ardlDlm(x = as.vector(temp), y = as.vector(rbo), p = i, q = j)
    new_row = data.frame(p = i, q=j, AIC=AIC(model.ARDL$model), BIC=BIC(model.ARDL$model))
    df = bind_rows(df, new_row)
  }
}

df[order(df$AIC),]
```

```
##    p q        AIC        BIC
## 12 3 2 -112.34584 -101.68821
## 13 3 3 -111.27343  -99.28358
## 3  1 3 -109.66585 -100.34042
## 8  2 3 -109.63896  -98.98132
## 1  1 1 -107.84188 -100.83590
## 11 3 1 -106.23515  -96.90972
## 17 4 2 -105.90379  -94.24126
## 2  1 2 -105.43977  -97.23600
## 7  2 2 -104.87521  -95.30414
## 18 4 3 -104.08513  -91.12676
## 14 3 4 -103.73107  -90.77270
## 4  1 4 -102.33843  -91.97173
## 9  2 4 -102.29557  -90.63304
## 6  2 1 -102.14956  -93.94579
## 19 4 4 -102.09820  -87.84400
## 16 4 1 -102.09407  -91.72737
## 22 5 2  -99.02330  -86.44233
## 21 5 1  -97.85381  -86.53094
## 5  1 5  -97.42825  -86.10538
## 23 5 3  -97.26093  -83.42187
## 15 3 5  -97.11952  -83.28046
## 10 2 5  -96.20616  -83.62519
## 24 5 4  -95.45616  -80.35900
## 20 4 5  -95.40963  -80.31247
## 25 5 5  -93.50136  -77.14610
```

```
model.ARDL1 <- ardlDlm(x = as.vector(temp), y = as.vector(rbo), p = 3, q = 2)
summary(model.ARDL1)
```

```
##
## Time series regression with "ts" data:
## Start = 4, End = 31
##
## Call:
## dynlm(formula = as.formula(model.text), data = data, start = 1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.04749 -0.01552 -0.00196  0.02033  0.04084
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.21430    0.22697   0.944   0.3558
## X.t          0.01329    0.01581   0.841   0.4100
## X.1          0.03610    0.01666   2.168   0.0418 *
## X.2         -0.02530    0.01905  -1.328   0.1984
## X.3         -0.03256    0.01970  -1.653   0.1131
## Y.1          0.33766    0.16642   2.029   0.0553 .
## Y.2          0.47160    0.17754   2.656   0.0148 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02824 on 21 degrees of freedom
## Multiple R-squared:  0.6504, Adjusted R-squared:  0.5505
## F-statistic: 6.512 on 6 and 21 DF,  p-value: 0.0005378
```

```
checkresiduals(model.ARDL1$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 9.1439, df = 10, p-value = 0.5185
```

```
MASE(model.ARDL1)
```

```
##                  MASE
## model.ARDL1 0.7726235
```

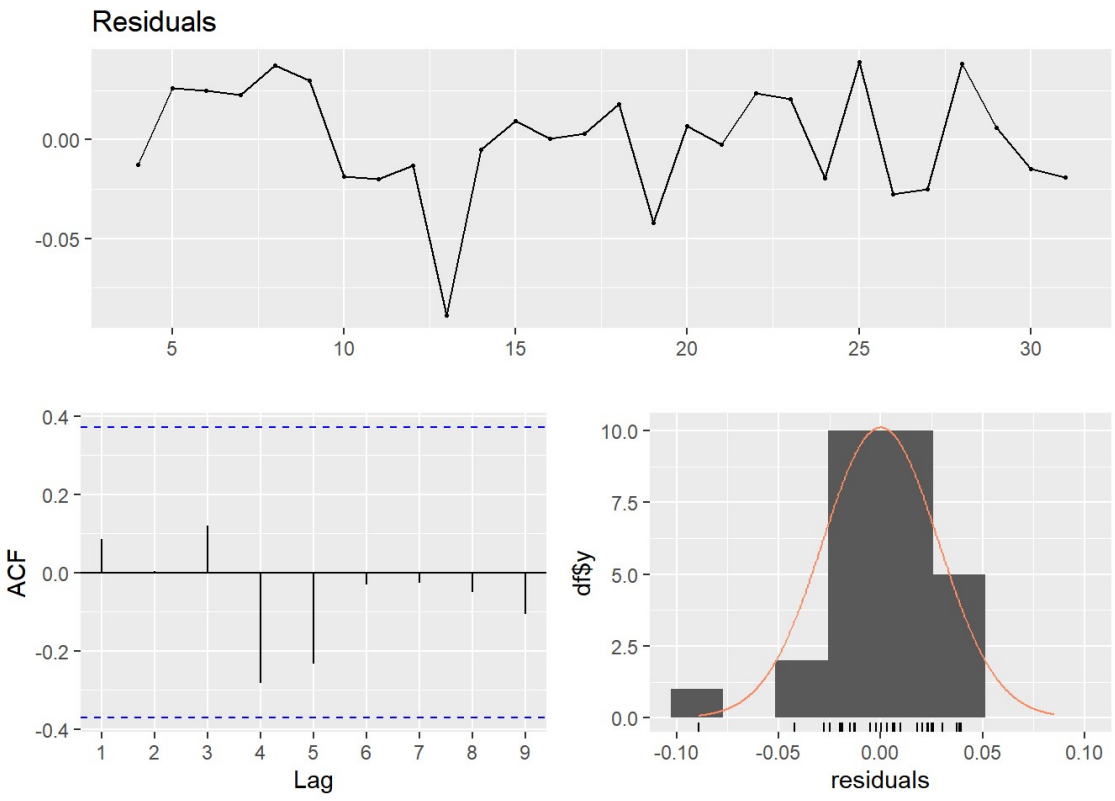
```
df = df[0,]

#rain
for(i in 1:5){
  for(j in 1:5){
    model.ARDL = ardlDlm(x = as.vector(rain), y = as.vector(rbo), p = i, q = j)
    new_row = data.frame(p = i, q=j, AIC=AIC(model.ARDL$model), BIC=BIC(model.ARDL$model))
    df = bind_rows(df, new_row)
  }
}

df[order(df$AIC),]
```

```
##    p q        AIC       BIC
## 3  1 3 -106.92478 -97.59935
## 12 3 2 -106.47543 -95.81780
## 1  1 1 -105.26187 -98.25588
## 13 3 3 -105.19964 -93.20980
## 8  2 3 -104.92840 -94.27076
## 2  1 2 -103.36807 -95.16429
## 4  1 4 -102.06783 -91.70114
## 11 3 1 -102.02868 -92.70325
## 7  2 2 -101.45523 -91.88416
## 17 4 2 -100.48808 -88.82555
## 9  2 4 -100.08923 -88.42670
## 18 4 3 -100.00494 -87.04657
## 14 3 4  -99.66585 -86.70748
## 6  2 1  -99.21505 -91.01127
## 19 4 4  -98.96532 -84.71111
## 16 4 1  -96.40802 -86.04133
## 5  1 5  -95.80256 -84.47969
## 22 5 2  -94.04730 -81.46633
## 23 5 3  -93.91526 -80.07620
## 10 2 5  -93.80342 -81.22246
## 21 5 1  -93.55318 -82.23031
## 15 3 5  -93.30294 -79.46387
## 24 5 4  -92.68035 -77.58319
## 20 4 5  -92.62017 -77.52301
## 25 5 5  -90.68282 -74.32757
```

```
model.ARDL2 <- ardlDlm(x = as.vector(rain), y = as.vector(rbo), p = 1, q = 3)
summary(model.ARDL2)
```

```
##
## Time series regression with "ts" data:
## Start = 4, End = 31
##
## Call:
## dynlm(formula = as.formula(model.text), data = data, start = 1)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.089117 -0.018820  0.001881  0.023063  0.039494
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.184586   0.114255   1.616   0.1204
## X.t         0.022068   0.016922   1.304   0.2057
## X.1         0.002395   0.017264   0.139   0.8909
## Y.1         0.254216   0.182738   1.391   0.1781
## Y.2         0.328851   0.173602   1.894   0.0714 .
## Y.3         0.081803   0.175168   0.467   0.6451
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0315 on 22 degrees of freedom
## Multiple R-squared:  0.5443, Adjusted R-squared:  0.4408
## F-statistic: 5.256 on 5 and 22 DF,  p-value: 0.002523
```

```
checkresiduals(model.ARDL2$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 9
##
## data:  Residuals
## LM test = 10.906, df = 9, p-value = 0.2822
```

```
MASE(model.ARDL2)
```

```
##                  MASE
## model.ARDL2 0.8322089
```

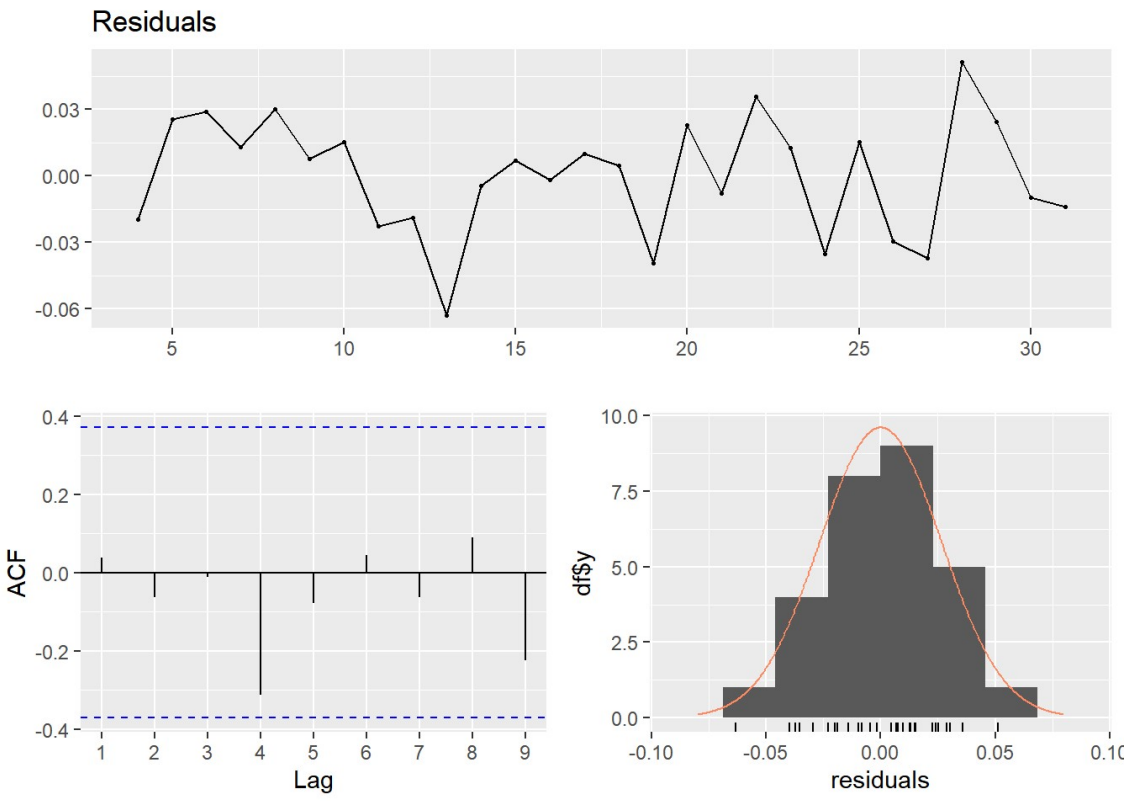
```
df = df[0,]

#radiation
for(i in 1:5){
  for(j in 1:5){
    model.ARDL = ardlDlm(x = as.vector(radiation), y = as.vector(rbo), p = i, q = j)
    new_row = data.frame(p = i, q=j, AIC=AIC(model.ARDL$model), BIC=BIC(model.ARDL$model))
    df = bind_rows(df, new_row)
  }
}

df[order(df$AIC),]
```

```
##    p q        AIC        BIC
## 3  1 3 -110.63384 -101.30840
## 8  2 3 -109.73015  -99.07252
## 12 3 2 -109.52593  -98.86830
## 13 3 3 -108.37333  -96.38349
## 1  1 1 -107.56217 -100.55618
## 2  1 2 -106.90395  -98.70018
## 7  2 2 -106.36636  -96.79529
## 4  1 4 -105.79231  -95.42561
## 9  2 4 -104.70473  -93.04219
## 11 3 1 -104.08228  -94.75685
## 17 4 2 -103.34313  -91.68060
## 14 3 4 -102.94049  -89.98212
## 18 4 3 -101.94383  -88.98546
## 19 4 4 -100.97902  -86.72481
## 6  2 1 -100.80851  -92.60473
## 5  1 5 -100.20325  -88.88038
## 16 4 1  -99.86384  -89.49714
## 10 2 5  -98.98255  -86.40158
## 15 3 5  -97.36077  -83.52170
## 22 5 2  -96.91278  -84.33182
## 21 5 1  -96.72745  -85.40458
## 23 5 3  -95.91334  -82.07428
## 20 4 5  -95.51106  -80.41390
## 24 5 4  -94.72831  -79.63116
## 25 5 5  -93.64810  -77.29285
```

```
model.ARDL3 <- ardlDlm(x = as.vector(radiation), y = as.vector(rbo), p = 1, q = 3)
summary(model.ARDL3)
```

```
##
## Time series regression with "ts" data:
## Start = 4, End = 31
##
## Call:
## dynlm(formula = as.formula(model.text), data = data, start = 1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.06308 -0.01905  0.00578  0.01710  0.05139
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.05413    0.32389   0.167   0.8688
## X.t         -0.02521    0.01588  -1.588   0.1266
## X.1          0.03255    0.01559   2.087   0.0487 *
## Y.1          0.35973    0.16351   2.200   0.0386 *
## Y.2          0.35620    0.15889   2.242   0.0354 *
## Y.3          0.05924    0.16656   0.356   0.7255
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02948 on 22 degrees of freedom
## Multiple R-squared:  0.6009, Adjusted R-squared:  0.5101
## F-statistic: 6.624 on 5 and 22 DF,  p-value: 0.0006674
```

```
checkresiduals(model.ARDL3$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 9
##
## data:  Residuals
## LM test = 12.329, df = 9, p-value = 0.1954
```

```
MASE(model.ARDL3)
```

```
##                  MASE
## model.ARDL3 0.8202653
```

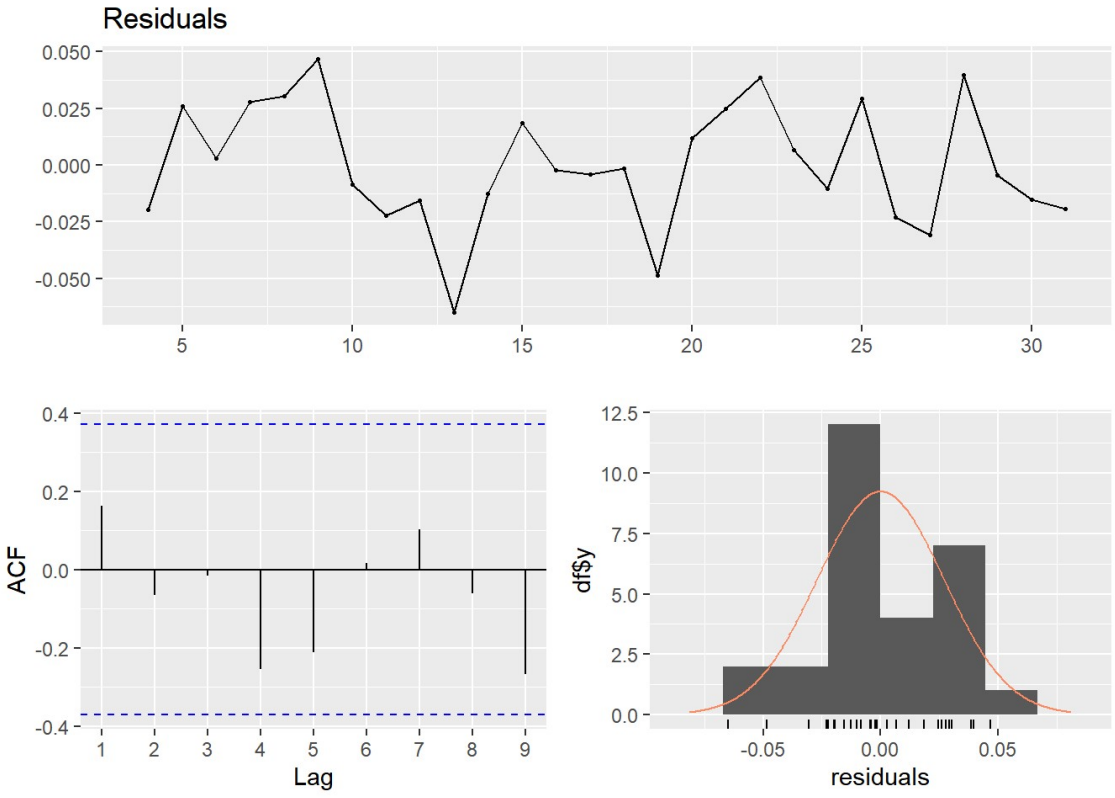
```
df = df[0,]

#relative humidity
for(i in 1:5){
  for(j in 1:5){
    model.ARDL = ardlDlm(x = as.vector(relhumid), y = as.vector(rbo), p = i, q = j)
    new_row = data.frame(p = i, q=j, AIC=AIC(model.ARDL$model), BIC=BIC(model.ARDL$model))
    df = bind_rows(df, new_row)
  }
}

df[order(df$AIC, decreasing = TRUE),]
```

```
##    p q        AIC       BIC
## 20 4 5  -93.41579 -78.31863
## 5  1 5  -95.07433 -83.75146
## 15 3 5  -95.36449 -81.52543
## 10 2 5  -95.44080 -82.85984
## 25 5 5  -96.63220 -80.27694
## 6  2 1  -96.65849 -88.45472
## 16 4 1  -97.72156 -87.35487
## 19 4 4  -98.43992 -84.18572
## 24 5 4  -98.61453 -83.51737
## 21 5 1  -99.29280 -87.96993
## 18 4 3  -99.83121 -86.87284
## 14 3 4 -100.37940 -87.42103
## 23 5 3 -100.48468 -86.64562
## 11 3 1 -100.54651 -91.22108
## 7  2 2 -100.67980 -91.10872
## 9  2 4 -101.39344 -89.73091
## 4  1 4 -101.43104 -91.06435
## 17 4 2 -101.68893 -90.02640
## 2  1 2 -102.12876 -93.92498
## 22 5 2 -102.28468 -89.70372
## 1  1 1 -103.40878 -96.40280
## 13 3 3 -105.86566 -93.87582
## 3  1 3 -106.34977 -97.02434
## 12 3 2 -107.17784 -96.52020
## 8  2 3 -107.64860 -96.99096
```

```
model.ARDL4 <- ardlDlm(x = as.vector(relhumid), y = as.vector(rbo), p = 2, q = 3)
summary(model.ARDL4)
```

```
##
## Time series regression with "ts" data:
## Start = 4, End = 31
##
## Call:
## dynlm(formula = as.formula(model.text), data = data, start = 1)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.065061 -0.016411 -0.003237  0.025084  0.046790
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.776421   1.537664  -1.806   0.0853 .
## X.t          0.013964   0.009090   1.536   0.1394
## X.1          0.002465   0.008465   0.291   0.7738
## X.2          0.013427   0.008286   1.620   0.1201
## Y.1          0.277394   0.169636   1.635   0.1169
## Y.2          0.479571   0.189200   2.535   0.0193 *
## Y.3          0.171679   0.188515   0.911   0.3728
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03071 on 21 degrees of freedom
## Multiple R-squared:  0.5866, Adjusted R-squared:  0.4684
## F-statistic: 4.966 on 6 and 21 DF,  p-value: 0.00262
```

```
checkresiduals(model.ARDL4$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 13.095, df = 10, p-value = 0.2184
```

```
MASE(model.ARDL4)
```

```
##                  MASE
## model.ARDL4 0.8184751
```

ARDL Model Summary

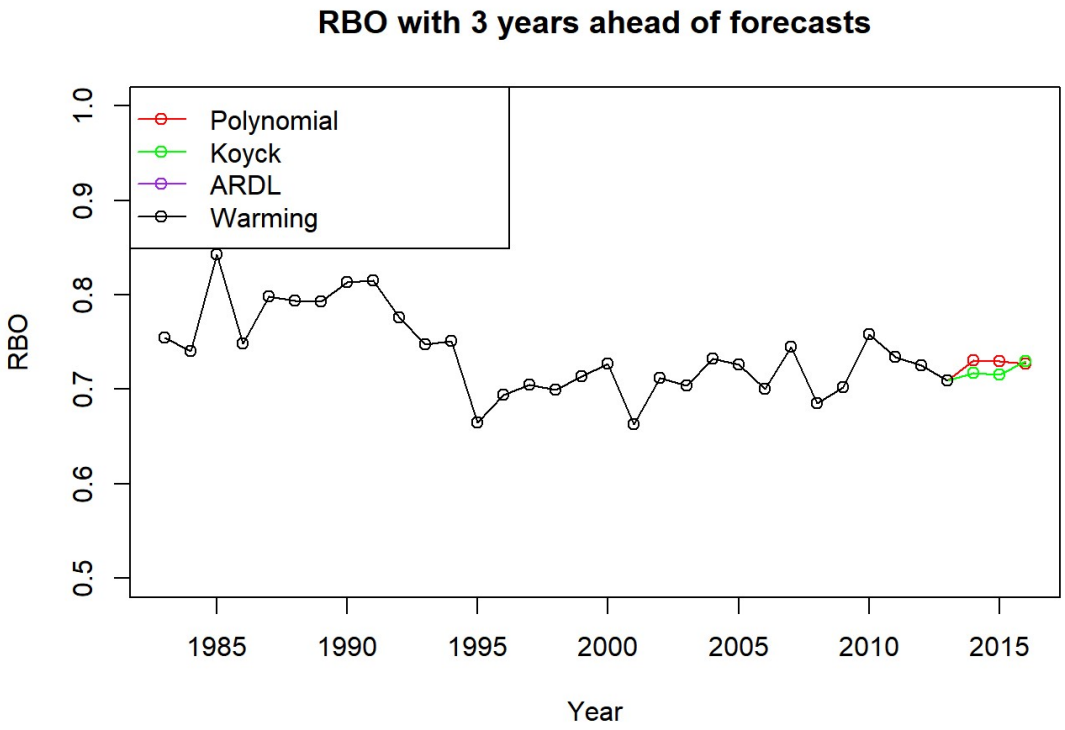
Model	MASE	Model p-value	Model R-squared value	Breusch-Godfrey p-value
Temperature	0.773	0.0005378	0.5505	0.5185
Rain	0.832	0.002523	0.4408	0.2822
Radiation	0.820	0.0006674	0.5101	0.1954
Relative Humidity	0.818	0.00262	0.4684	0.2184

All the ARDL models are significant and show the best R-squared values thus far. Of these, temperature produced the best ARDL model, having the lowest MASE, lowest p-value, highest R-squared value, and highest p-value for the Breusch-Godfrey test. All models have no serial correlation since all of them have high p-values for the Breusch-Godfrey test.

Forecasting with Time Series Regression Models

Three years ahead forecasts are created for the polynomial and Koyck models. The resulting plot shows that Koyck and polynomial models produces somewhat similar forecasts where the Koyck predictions remain slightly lower than the polynomial forecast until 2018, where they align.

```
predictor <- read.csv("Covariate x-values for Task 3 .csv")
model.poly.Frc = dLagM::forecast(model.poly2, x = t(predictor$Rainfall), h=3)
model.koyck.Frc = dLagM::forecast(model.Koyck4, x = t(predictor$RelHumidity), h=3)

plot(ts(c(as.vector(rbo), model.poly.Frc$forecasts), start = 1983, frequency = 1), type="o", col="red", ylim=
c(0.5, 1),
     ylab="RBO", xlab="Year", main="RBO with 3 years ahead of forecasts")
lines(ts(c(as.vector(rbo), model.koyck.Frc$forecasts), start = 1983, frequency = 1), col="green",type="o")
lines(ts(as.vector(rbo), start = 1983, frequency = 1),col="black",type="o")
legend("topleft",lty=1, pch = 1, text.width = 11, col=c( "red", "green","purple", "black"),
       c("Polynomial", "Koyck", "ARDL", "Warming"))
```


RBO with 3 years ahead of forecasts
Polynomial
Koyck
ARDL
Warming
RBO
0.5
0.6
0.7
0.8
0.9
1.0
1985
1990
1995
2000
2005
2010
2015
Year

Exponential Smoothing Models

A simple exponential smoothing model, a Holt's linear trend model, and a Holt's linear trend model with additive damped trend are implemented using the RBO series. Models that take into account seasonal components are not considered due to the annual frequency of the time series data. The three above models are created for the FFD series using the ses() and holt() functions.

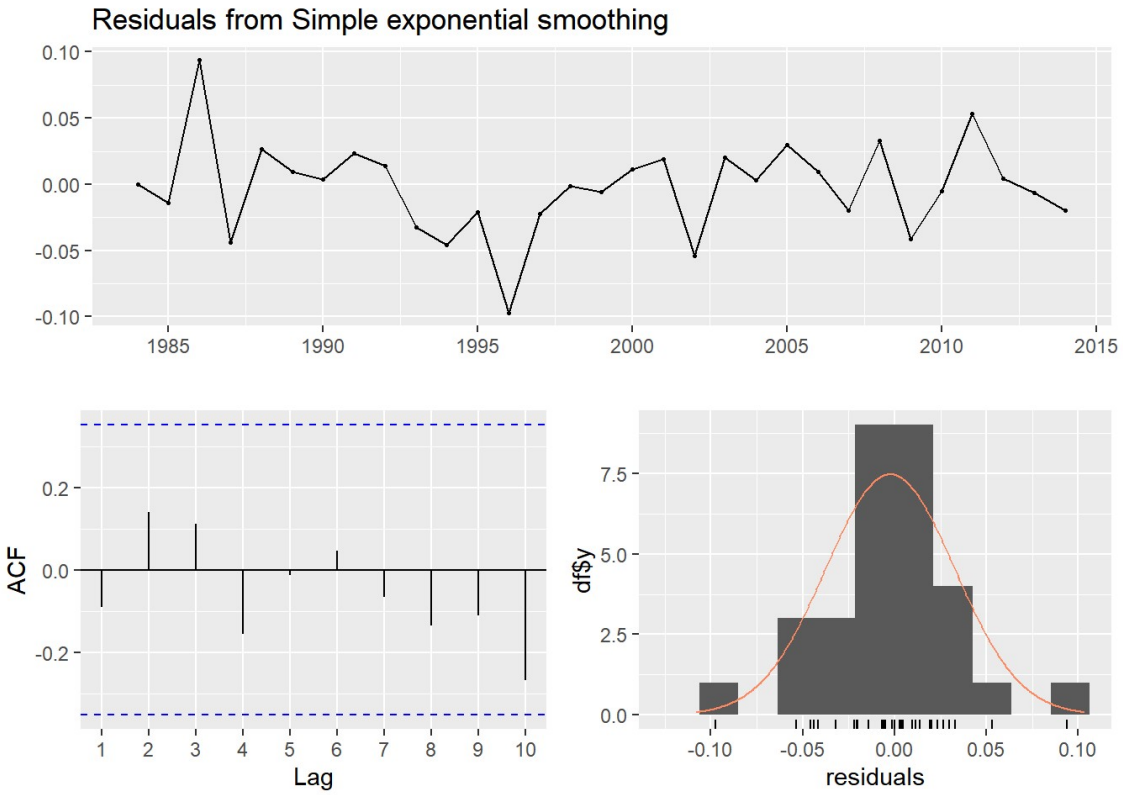
The results of the point forecasts show that the SES and the additive damped trend models produce a constant forecast at around the 0.72 mark. In contrast, the Holt's linear trend forecast starts slightly lower at around 0.716 and decreases sightly for the duration of the forecast.

The three above models are created for the RBO series using the ses() and holt() functions.

```
fit1.ses <- ses(rbo, initial="simple", h=3)
summary(fit1.ses)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = rbo, h = 3, initial = "simple")
##
##   Smoothing parameters:
##     alpha = 0.468
##
##   Initial states:
##     l = 0.755
##
##   sigma:  0.0347
## Error measures:
##                       ME       RMSE        MAE        MPE     MAPE      MASE
## Training set -0.00243338 0.03473574 0.02534398 -0.4992045 3.447876 0.8224266
##                     ACF1
## Training set -0.09061714
##
## Forecasts:
##      Point Forecast     Lo 80     Hi 80     Lo 95     Hi 95
## 2015      0.7197027 0.6751870 0.7642183 0.6516219 0.7877835
## 2016      0.7197027 0.6705526 0.7688528 0.6445341 0.7948713
## 2017      0.7197027 0.6663189 0.7730864 0.6380593 0.8013461
```

```
checkresiduals(fit1.ses)
```

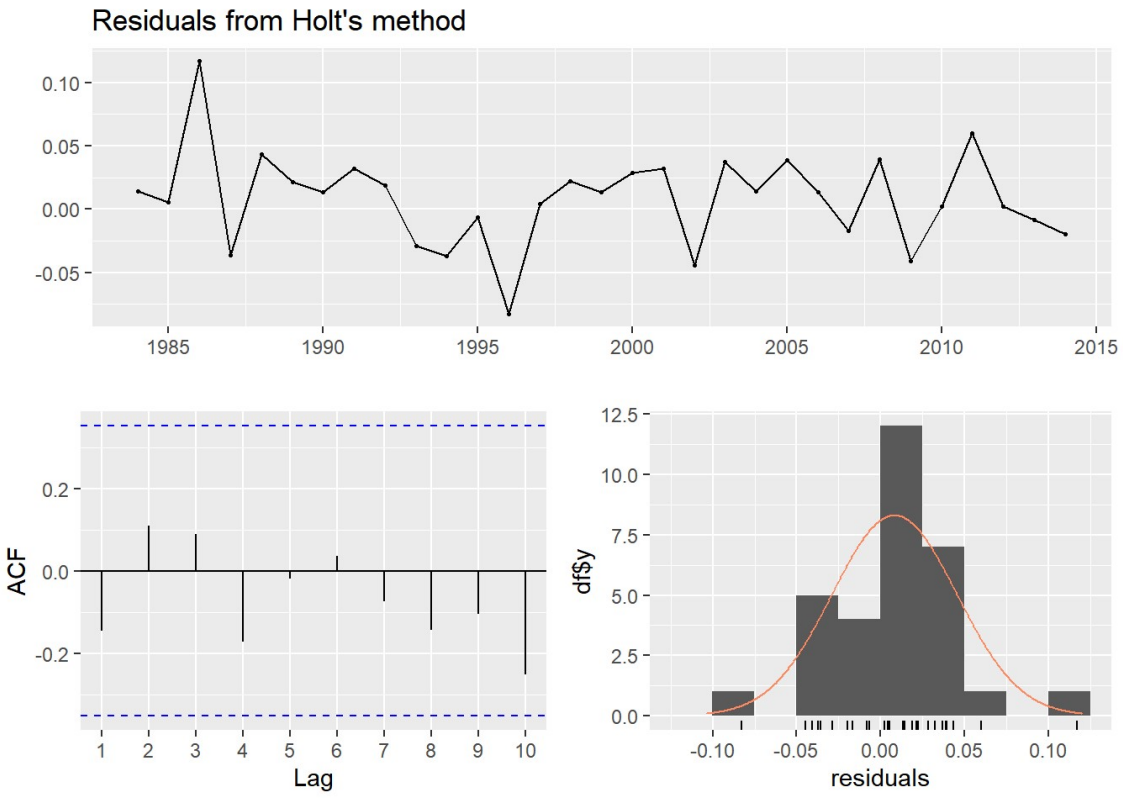



```
##
##  Ljung-Box test
##
## data:  Residuals from Simple exponential smoothing
## Q* = 2.467, df = 6, p-value = 0.8721
##
## Model df: 0.   Total lags used: 6
```

```
fit2.holt <- holt(rbo, initial="simple", h=3)
summary(fit2.holt)
```

```
##
## Forecast method: Holt's method
##
## Model Information:
## Holt's method
##
## Call:
##  holt(y = rbo, h = 3, initial = "simple")
##
##   Smoothing parameters:
##     alpha = 0.5678
##     beta  = 0.0888
##
##   Initial states:
##     l = 0.755
##     b = -0.0143
##
##   sigma:  0.0376
## Error measures:
##                       ME       RMSE        MAE       MPE     MAPE      MASE
## Training set 0.008262872 0.03758621 0.02896041 0.9541908 3.905009 0.9397818
##                    ACF1
## Training set -0.1449597
##
## Forecasts:
##      Point Forecast     Lo 80     Hi 80     Lo 95     Hi 95
## 2015      0.7162102 0.6680415 0.7643789 0.6425426 0.7898778
## 2016      0.7148677 0.6572445 0.7724909 0.6267407 0.8029948
## 2017      0.7135252 0.6456320 0.7814184 0.6096915 0.8173589
```

```
checkresiduals(fit2.holt)
```

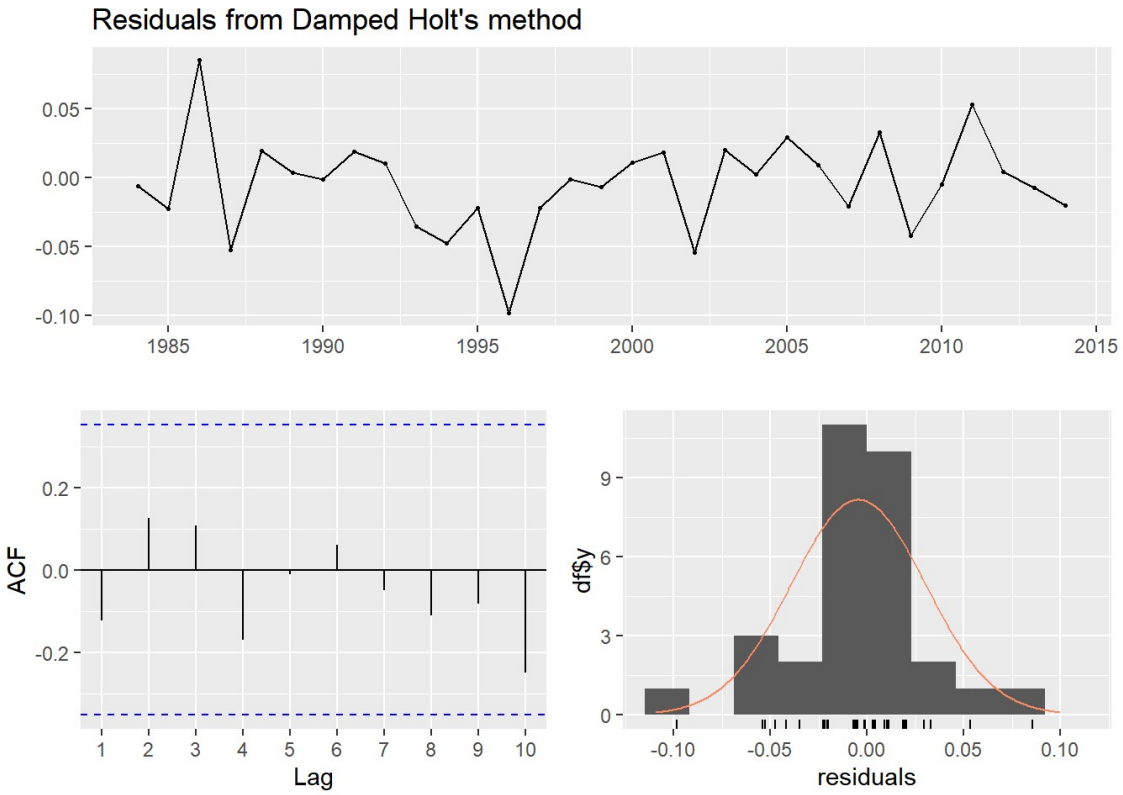


```
##
##  Ljung-Box test
##
## data:  Residuals from Holt's method
## Q* = 2.6105, df = 6, p-value = 0.8559
##
## Model df: 0.   Total lags used: 6
```

```
fit3.holt <- holt(rbo, damped=TRUE, initial="simple", h=3) # Fit with additive damped trend
summary(fit3.holt)
```

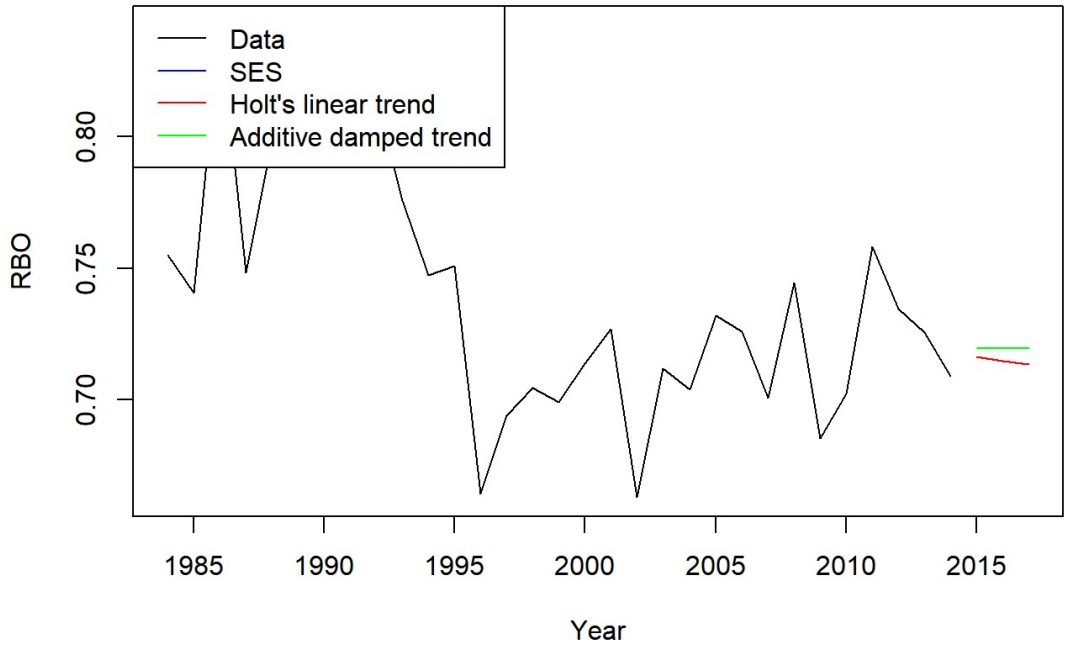
```
##
## Forecast method: Damped Holt's method
##
## Model Information:
## Damped Holt's method
##
## Call:
##  holt(y = rbo, h = 3, damped = TRUE, initial = "simple")
##
##   Smoothing parameters:
##     alpha = 0.4773
##     beta  = 1e-04
##     phi   = 0.8
##
##   Initial states:
##     l = 0.7542
##     b = 0.0082
##
##   sigma:  0.0377
##
##       AIC      AICc       BIC
## -90.15879 -86.65879 -81.55487
##
## Error measures:
##                       ME       RMSE        MAE        MPE     MAPE      MASE
## Training set -0.004553134 0.03457183 0.02527824 -0.7722916 3.450154 0.8202932
##                    ACF1
## Training set -0.1237046
##
## Forecasts:
##      Point Forecast     Lo 80     Hi 80     Lo 95     Hi 95
## 2015      0.7195752 0.6711967 0.7679537 0.6455866 0.7935638
## 2016      0.7195801 0.6659717 0.7731885 0.6375931 0.8015670
## 2017      0.7195840 0.6612112 0.7779567 0.6303106 0.8088574
```

```
checkresiduals(fit3.holt)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt's method
## Q* = 2.7586, df = 6, p-value = 0.8385
##
## Model df: 0.   Total lags used: 6
```

```
plot(rbo, type="l", ylab="RBO", xlab="Year", fcol="white", plot.conf=FALSE, xlim=c(1984,2017))
lines(fit1.ses$mean, col="blue", type="l")
lines(fit2.holt$mean, col="red", type="l")
lines(fit3.holt$mean, col="green", type="l")
legend("topleft", lty=1, col=c("black","blue","red","green"),
       c("Data","SES", "Holt's linear trend", "Additive damped trend"))
```



Model Summary

Model	AIC	BIC	MASE	Ljung-Box p-value
Simple SES	NA	NA	0.8224266	0.8721
Simple Holt	NA	NA	0.9397818	0.8559
Additive Damped Trend Holt SES	-90.15879	-81.55487	0.8202932	0.8385

Checking the summaries and residuals for each model reveals that the additive damped trend Holt model produces the lowest overall MASE and can be considered to be the best overall model.

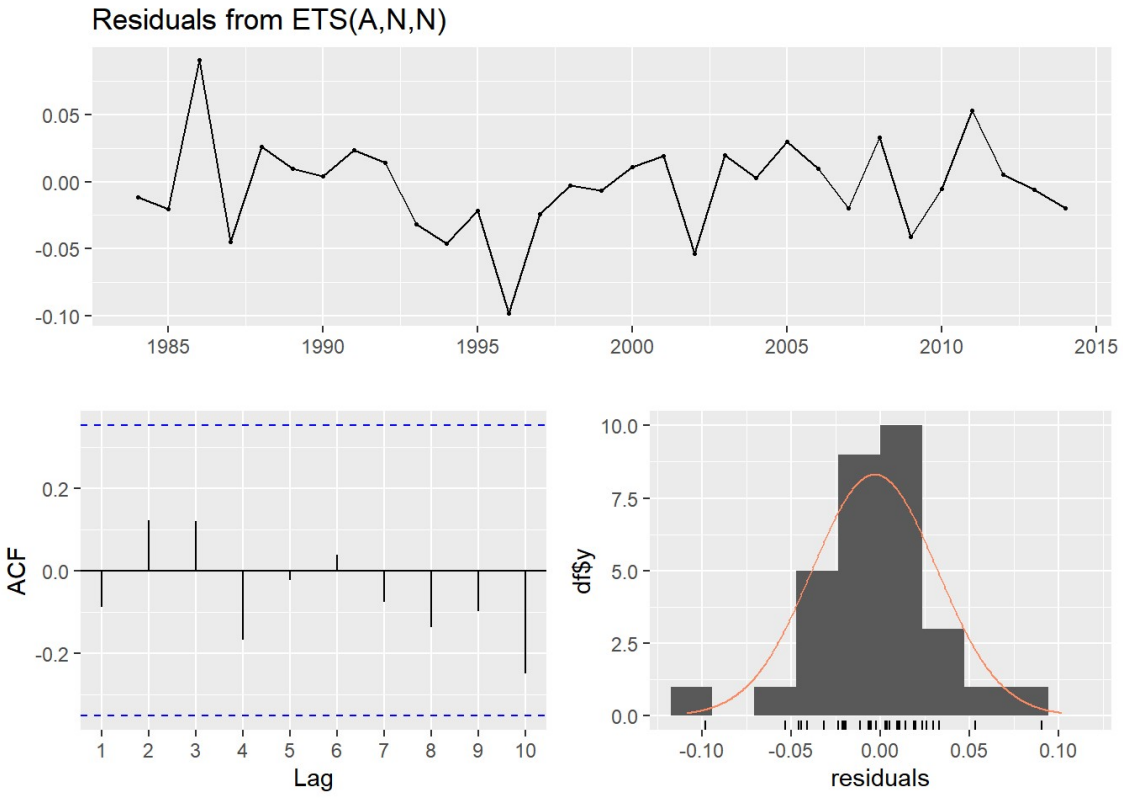
State Space Models

ANN, MNN, AAN, and MMN state space models are created for the RBO series and their residuals and summaries are discussed below:

```
etsAN = ets(rbo, model="ANN")
summary(etsAN)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = rbo, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 0.4546
##
##   Initial states:
##     l = 0.7664
##
##   sigma:  0.0358
##
##       AIC      AICc       BIC
## -96.01071 -95.12182 -91.70874
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE     MAPE     MASE
## Training set -0.003300842 0.0346545 0.0259001 -0.6143979 3.524917 0.840473
##                     ACF1
## Training set -0.08757701
```

```
checkresiduals(etsAN)
```

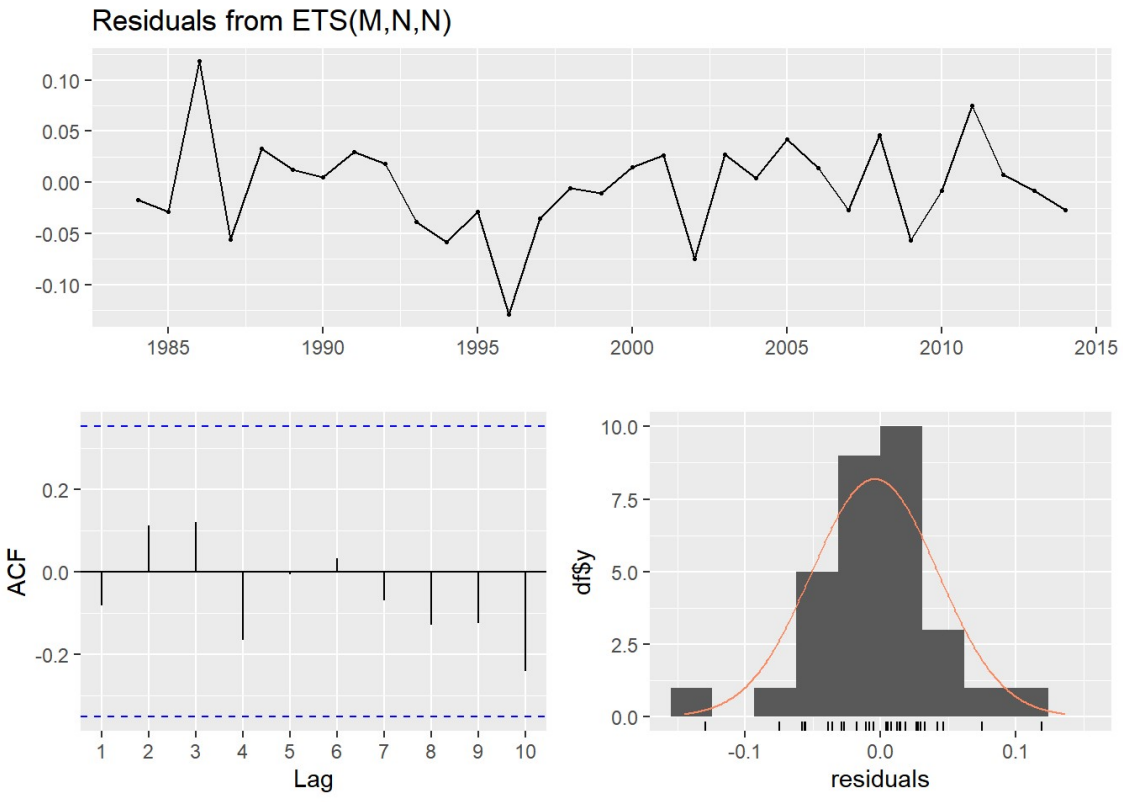


```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,N)
## Q* = 2.4644, df = 6, p-value = 0.8724
##
## Model df: 0.   Total lags used: 6
```

```
etsMN = ets(rbo, model="MNN")
summary(etsMN)
```

```
## ETS(M,N,N)
##
## Call:
##  ets(y = rbo, model = "MNN")
##
##   Smoothing parameters:
##     alpha = 0.4421
##
##   Initial states:
##     l = 0.7685
##
##   sigma:  0.0479
##
##       AIC      AICc       BIC
## -96.69180 -95.80291 -92.38984
##
## Training set error measures:
##                        ME       RMSE        MAE        MPE     MAPE      MASE
## Training set -0.003529451 0.03466016 0.02607253 -0.6459794 3.549133 0.8460683
##                    ACF1
## Training set -0.0753464
```

```
checkresiduals(etsMN)
```

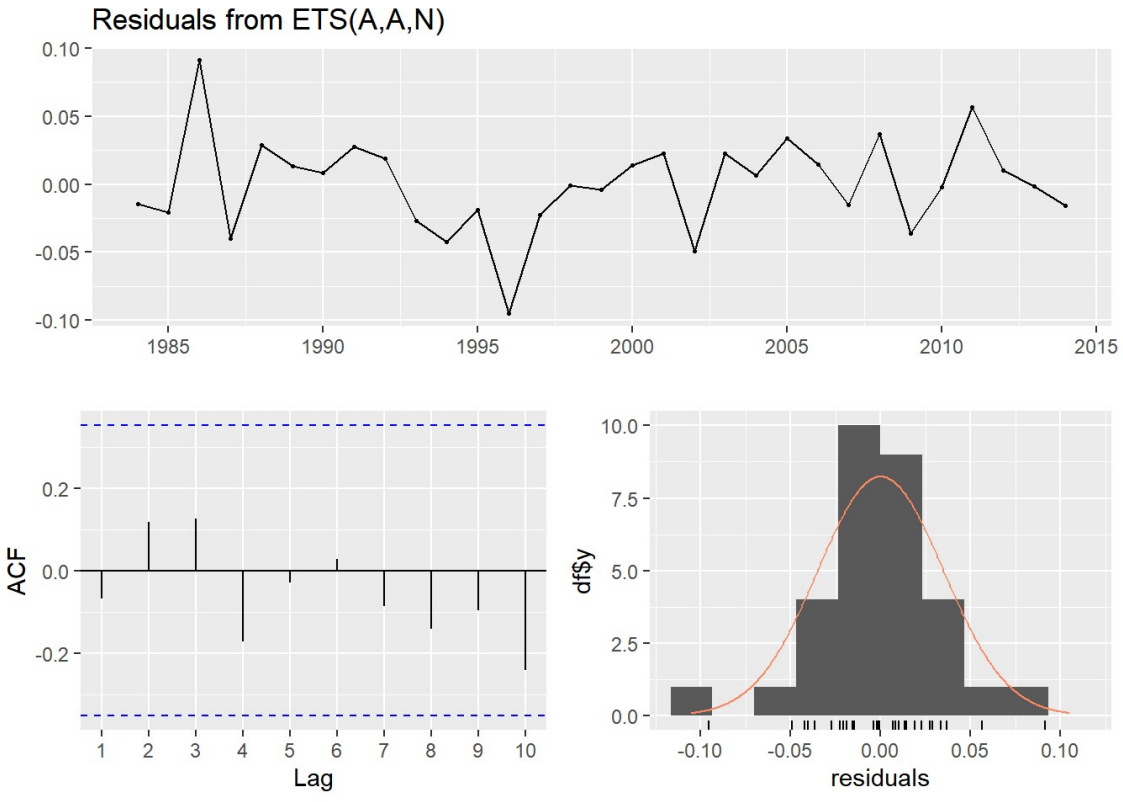


```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,N)
## Q* = 2.2852, df = 6, p-value = 0.8917
##
## Model df: 0.   Total lags used: 6
```

```
etsAA = ets(rbo, model="AAN")
summary(etsAA)
```

```
## ETS(A,A,N)
##
## Call:
##  ets(y = rbo, model = "AAN")
##
##   Smoothing parameters:
##     alpha = 0.4315
##     beta  = 1e-04
##
##   Initial states:
##     l = 0.7714
##     b = -0.0017
##
##   sigma:  0.0369
##
##       AIC      AICc       BIC
## -92.34177 -89.94177 -85.17183
##
## Training set error measures:
##                        ME       RMSE        MAE       MPE     MAPE      MASE
## Training set 5.416433e-06 0.03446995 0.02622961 -0.164231 3.555782 0.8511657
##                      ACF1
## Training set -0.06701124
```

```
checkresiduals(etsAA)
```

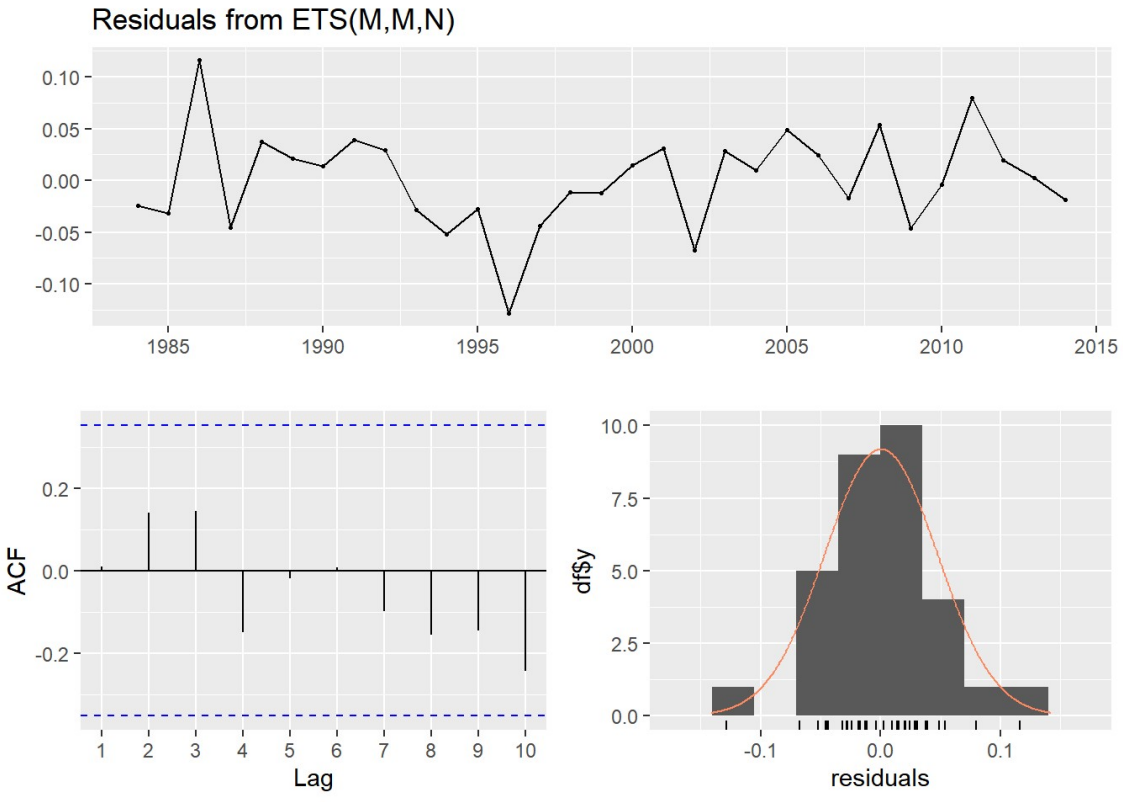


```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,A,N)
## Q* = 2.4085, df = 6, p-value = 0.8786
##
## Model df: 0.   Total lags used: 6
```

```
etsMM = ets(rbo, model="MMN")
summary(etsMM)
```

```
## ETS(M,M,N)
##
## Call:
##  ets(y = rbo, model = "MMN")
##
##   Smoothing parameters:
##     alpha = 0.3551
##     beta  = 1e-04
##
##   Initial states:
##     l = 0.7756
##     b = 0.9975
##
##   sigma:  0.0497
##
##       AIC      AICc       BIC
## -92.87345 -90.47345 -85.70351
##
## Training set error measures:
##                         ME       RMSE        MAE        MPE     MAPE      MASE
## Training set -3.033522e-05 0.03455053 0.02699819 -0.1767263 3.660989 0.8761064
##                     ACF1
## Training set 0.01629496
```

```
checkresiduals(etsMM)
```

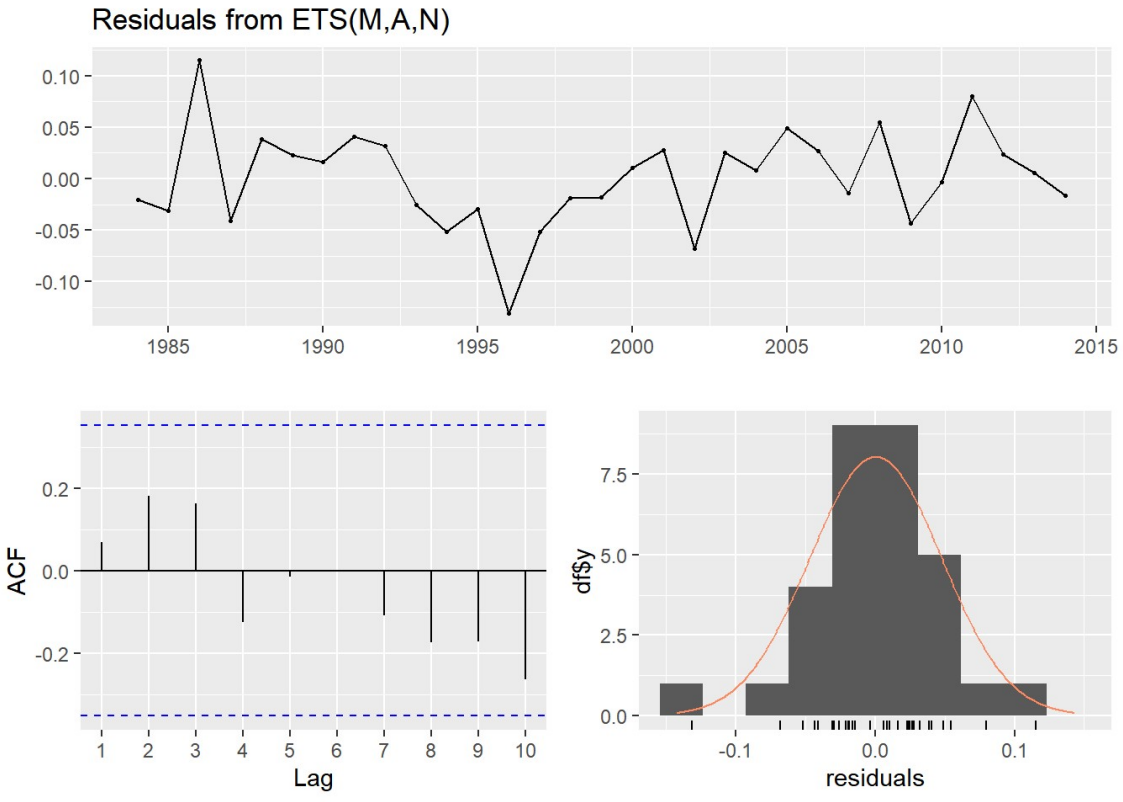


```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,M,N)
## Q* = 2.3252, df = 6, p-value = 0.8875
##
## Model df: 0.   Total lags used: 6
```

```
etsMA = ets(rbo, model="MAN")
summary(etsMA)
```

```
## ETS(M,A,N)
##
## Call:
##  ets(y = rbo, model = "MAN")
##
##   Smoothing parameters:
##     alpha = 0.3099
##     beta  = 1e-04
##
##   Initial states:
##     l = 0.7728
##     b = -0.0018
##
##   sigma:  0.0501
##
##       AIC      AICc       BIC
## -92.38368 -89.98368 -85.21374
##
## Training set error measures:
##                        ME       RMSE        MAE        MPE    MAPE      MASE
## Training set 3.067015e-05 0.03485212 0.02734062 -0.1759735 3.7086 0.8872188
##                    ACF1
## Training set 0.07723508
```

```
checkresiduals(etsMA)
```



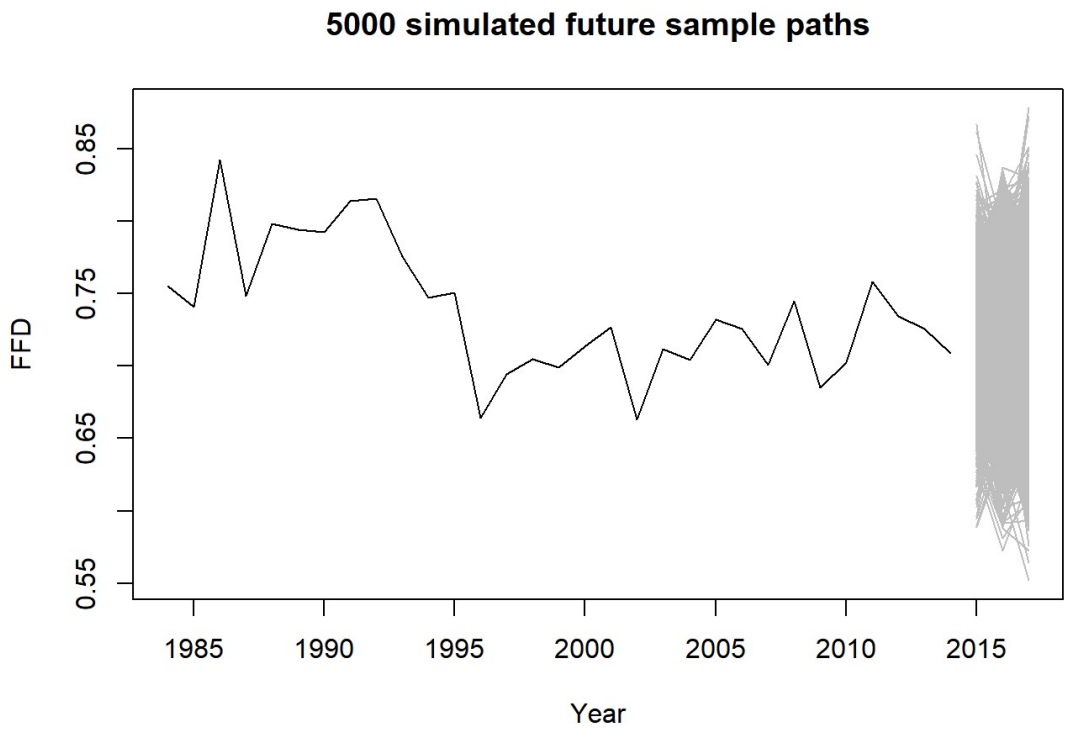
```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,A,N)
## Q* = 2.8699, df = 6, p-value = 0.825
##
## Model df: 0.   Total lags used: 6
```

Model Summary

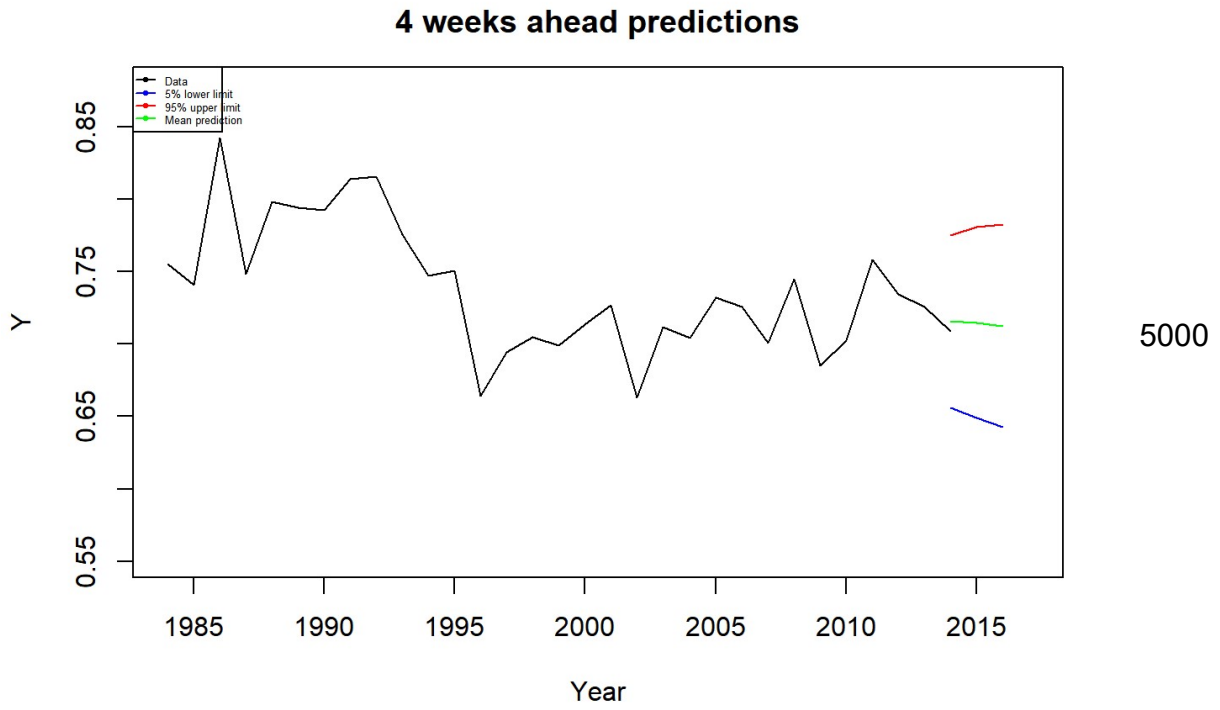
Model	AIC	BIC	MASE	Ljung-Box p-value
ANN State Space	-96.01071	-91.70874	0.840473	0.8724
MNN State Space	-96.69180	-92.38984	0.8460683	0.8917
AAN State Space	-92.34177	-85.17183	0.8511657	0.8786
MMN State Space	-92.87345	-85.70351	0.8761064	0.8875
MAN State Space	-92.38368	-85.21374	0.8872188	0.825

The lowest MASE value is seen in the ANN state space model, but the lowest AIC and BIC are observed in the AAN model. All the models do not seem to have serial correlation due to the failure to reject the null hypothesis of serial correlation using the Ljung-Box test. The AAN state space model is chosen to be the best due to its lower AIC and BIC values and since its MASE is not too much higher than that of th ANN model.

```
A = ts(matrix(NA,3,5000),start=2015,frequency = 1)

M = 5000
for (i in 1:M){
  A[,i] = simulate(etsAA , initstate = etsAA$states[30,] , nsim = 3)
}

plot(rbo , ylim=range(rbo,A) , xlim=c(1984,2017),
     ylab="FFD" , xlab="Year", main="5000 simulated future sample paths")
for(i in 1:M){
  lines(A[,i],col="gray")
}
```




```
# interval estimates and bounds

N = 3
xlim=c(1984,2017)

Pi = array(NA, dim=c(N,2))
avrg = array(NA, N)

# Calculate the interval estimates and mid point
for (i in 1:N){
  Pi[i,] = quantile(A[i,], type=8, prob=c(.05,.95))
  avrg[i] = mean(A[i,]) # This would be median as well
}

# Create ts objects for plotting
Pi.lb = ts(Pi[,1],start=end(rbo),f=1)
Pi.ub = ts(Pi[,2],start=end(rbo),f=1)
avrg.pred = ts(avrg,start=end(rbo),f=1)

plot(rbo,xlim=xlim , ylim=range(rbo,A),ylab="Y",xlab="Year", main="
4 weeks ahead predictions")
lines(Pi.lb,col="blue", type="l")
lines(Pi.ub,col="red", type="l")
lines(avrg.pred,col="green", type="l")
legend("topleft", lty=1, pch=1, col=c("black","blue","red","green"), text.width =2, cex=0.4,
       c("Data","5% lower limit","95% upper limit","Mean prediction"))
```



5000

simulated future paths are plotted to obtain an idea of the potential range of forecasts that could be obtained. The simulated paths demonstrate a periodic rise and fall. Plotting the forecast along with the confidence intervals shows a roughly constant forecast (green) at around 0.71. The confidence intervals seem quite wide, ranging from roughly 0.65 till 0.78 and both bounds seem equidistant from the forecast.

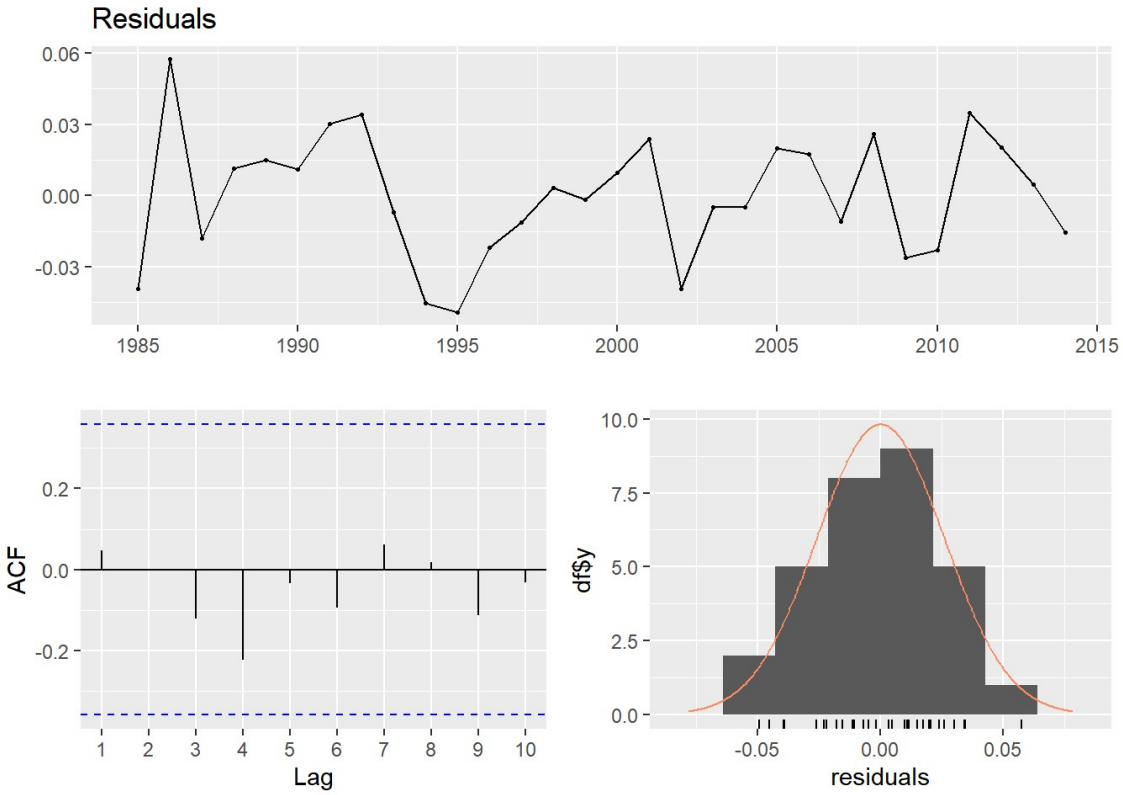
Task 3 Part B

A drought period occurring from 1996 onwards implies that an intervention point begins at that year. Therefore, a dynamic linear model can be implemented with an intervention point set to in the year 1996. Multiple dynamic linear models are implemented based on various combinations of Y(t), step functions, trend, and their lags. The summaries and residuals of these models are compared and the ideal model is chosen based on factors such as the model p-value, R-squared, AIC, BIC, and results of the Breusch-Godfrey test.

```
Y.t = rbo
T = 13
S.t = 1*(seq(Y.t) >= T)
S.t.1 = Lag(S.t,+1)

model1 = dynlm(Y.t ~ L(Y.t , k = 1 ) + S.t + trend(Y.t))
summary(model1)
```

```
##
## Time series regression with "ts" data:
## Start = 1985, End = 2014
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, k = 1) + S.t + trend(Y.t))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.049309 -0.017320  0.000662  0.019222  0.057536
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.927274   0.144362   6.423 8.34e-07 ***
## L(Y.t, k = 1) -0.199722   0.186129  -1.073 0.293115
## S.t           -0.114898   0.025816  -4.451 0.000143 ***
## trend(Y.t)     0.001840   0.001093   1.683 0.104318
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02754 on 26 degrees of freedom
## Multiple R-squared:  0.668,  Adjusted R-squared:  0.6297
## F-statistic: 17.44 on 3 and 26 DF,  p-value: 2.073e-06
```

```
checkresiduals(model1)
```

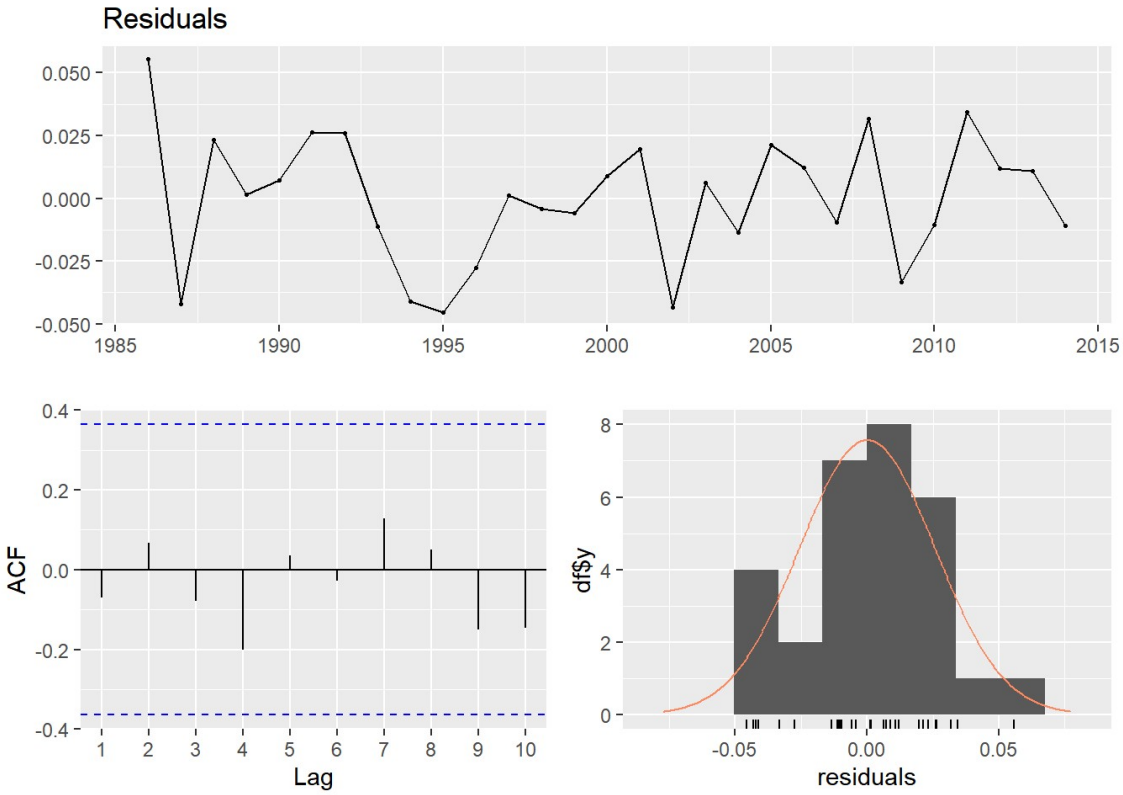



```
##
##  Breusch-Godfrey test for serial correlation of order up to 7
##
## data:  Residuals
## LM test = 6.1617, df = 7, p-value = 0.521
```

```
model2 = dynlm(Y.t ~ L(Y.t , k = 2 ) + S.t + trend(Y.t))
summary(model2)
```

```
##
## Time series regression with "ts" data:
## Start = 1986, End = 2014
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, k = 2) + S.t + trend(Y.t))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.045517 -0.011302  0.001648  0.019562  0.055581
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.908321   0.143497   6.330 1.26e-06 ***
## L(Y.t, k = 2) -0.166685   0.183717  -0.907 0.372915
## S.t           -0.110586   0.025209  -4.387 0.000183 ***
## trend(Y.t)     0.001444   0.001090   1.325 0.197263
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02725 on 25 degrees of freedom
## Multiple R-squared:  0.6874, Adjusted R-squared:  0.6498
## F-statistic: 18.32 on 3 and 25 DF,  p-value: 1.688e-06
```

```
checkresiduals(model2)
```

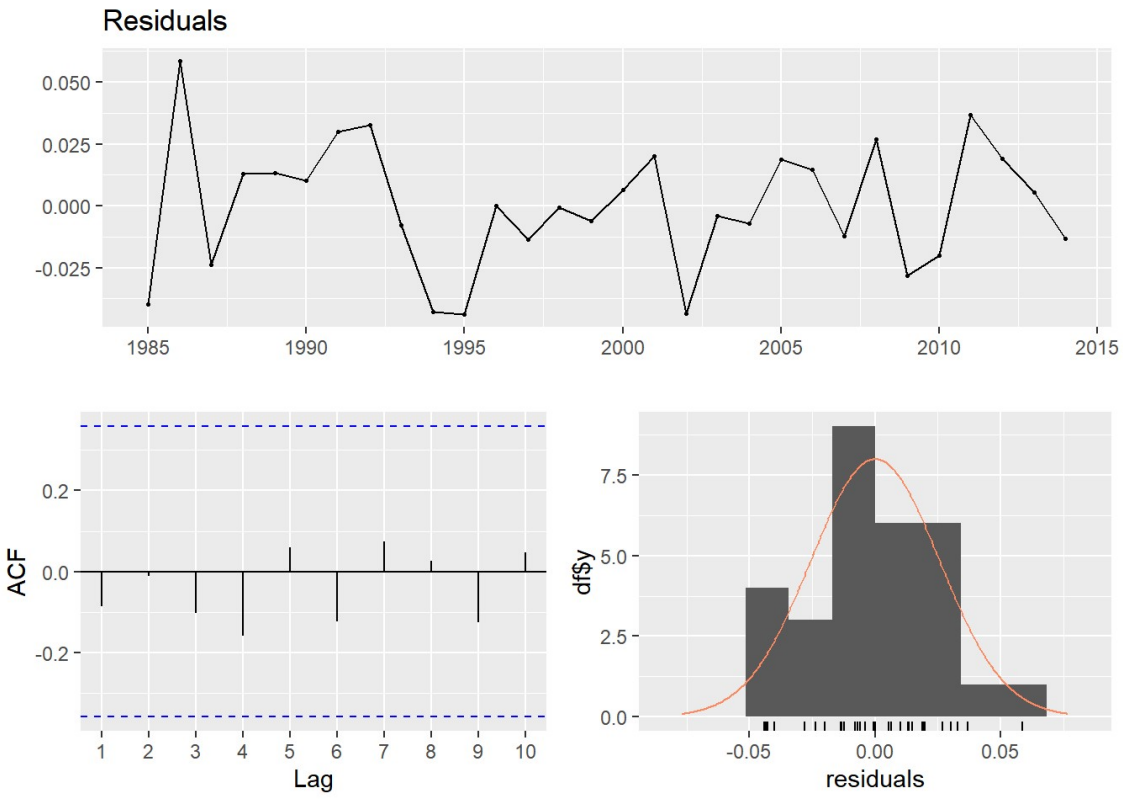



```
##
##  Breusch-Godfrey test for serial correlation of order up to 7
##
## data:  Residuals
## LM test = 3.1577, df = 7, p-value = 0.87
```

```
model3 = dynlm(Y.t ~ L(Y.t , k = 1 ) + S.t + S.t.1 + trend(Y.t))
summary(model3)
```

```
##
## Time series regression with "ts" data:
## Start = 1985, End = 2014
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, k = 1) + S.t + S.t.1 + trend(Y.t))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.043993 -0.013514 -0.000361  0.017843  0.058620
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.873331   0.155517   5.616 7.65e-06 ***
## L(Y.t, k = 1) -0.126246   0.202073  -0.625 0.537792
## S.t           -0.131229   0.031112  -4.218 0.000283 ***
## S.t.1          0.031041   0.032850   0.945 0.353738
## trend(Y.t)     0.001318   0.001227   1.074 0.293298
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02759 on 25 degrees of freedom
## Multiple R-squared:  0.6794, Adjusted R-squared:  0.6281
## F-statistic: 13.25 on 4 and 25 DF,  p-value: 6.329e-06
```

```
checkresiduals(model3)
```

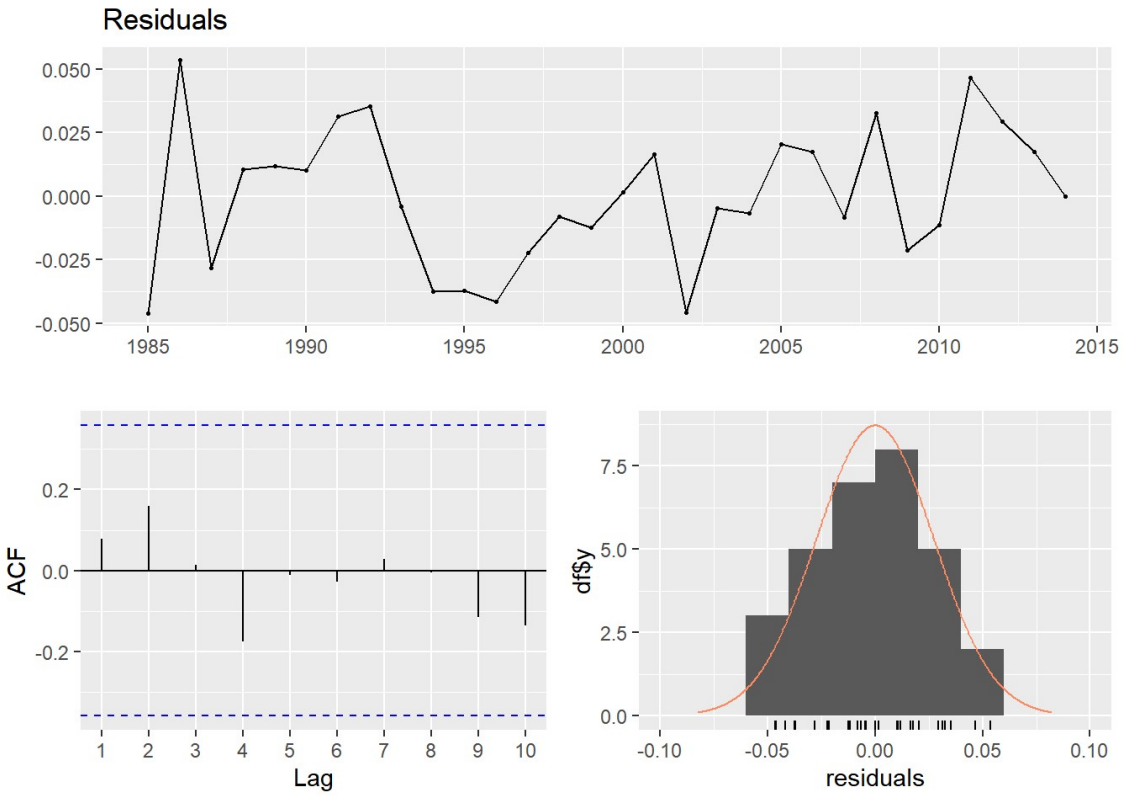



```
##
##  Breusch-Godfrey test for serial correlation of order up to 8
##
## data:  Residuals
## LM test = 9.3145, df = 8, p-value = 0.3165
```

```
model4 = dynlm(Y.t ~ L(Y.t , k = 1 ) + S.t)
summary(model4)
```

```
##
## Time series regression with "ts" data:
## Start = 1985, End = 2014
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, k = 1) + S.t)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.046261 -0.019149 -0.001989  0.017593  0.053684
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.87648    0.14589   6.008 2.07e-06 ***
## L(Y.t, k = 1) -0.11850    0.18577  -0.638    0.529
## S.t           -0.08151    0.01707  -4.775 5.58e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02846 on 27 degrees of freedom
## Multiple R-squared:  0.6318, Adjusted R-squared:  0.6045
## F-statistic: 23.17 on 2 and 27 DF,  p-value: 1.387e-06
```

```
checkresiduals(model4)
```

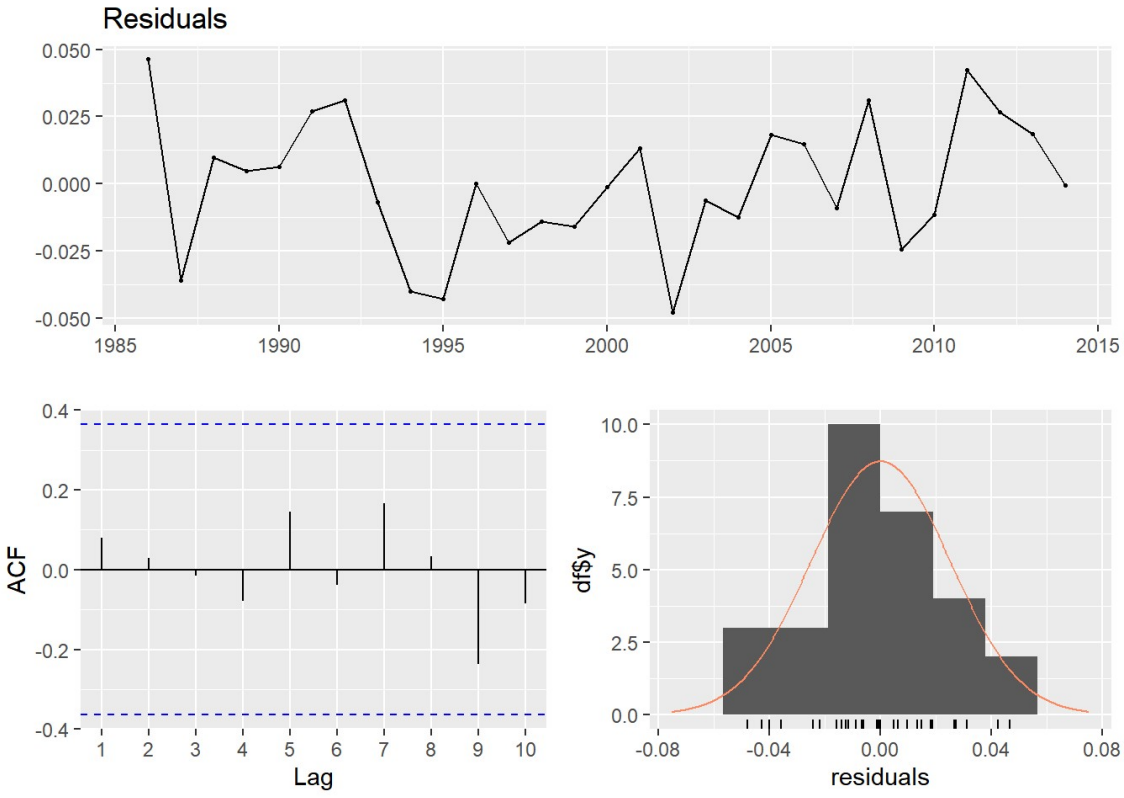



```
##
##  Breusch-Godfrey test for serial correlation of order up to 6
##
## data:  Residuals
## LM test = 2.3825, df = 6, p-value = 0.8814
```

```
model5 = dynlm(Y.t ~ L(Y.t , k = 1 ) + L(Y.t , k = 2 ) + S.t+ S.t.1)
summary(model5)
```

```
##
## Time series regression with "ts" data:
## Start = 1986, End = 2014
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, k = 1) + L(Y.t, k = 2) + S.t + S.t.1)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.048023 -0.013945 -0.000536  0.018214  0.046523
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.94084    0.22914   4.106 0.000403 ***
## L(Y.t, k = 1) -0.12295    0.19020  -0.646 0.524141
## L(Y.t, k = 2) -0.07139    0.18671  -0.382 0.705555
## S.t           -0.13074    0.03049  -4.288 0.000254 ***
## S.t.1          0.04118    0.02990   1.377 0.181144
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02706 on 24 degrees of freedom
## Multiple R-squared:  0.7039, Adjusted R-squared:  0.6546
## F-statistic: 14.27 on 4 and 24 DF,  p-value: 4.286e-06
```

```
checkresiduals(model5)
```

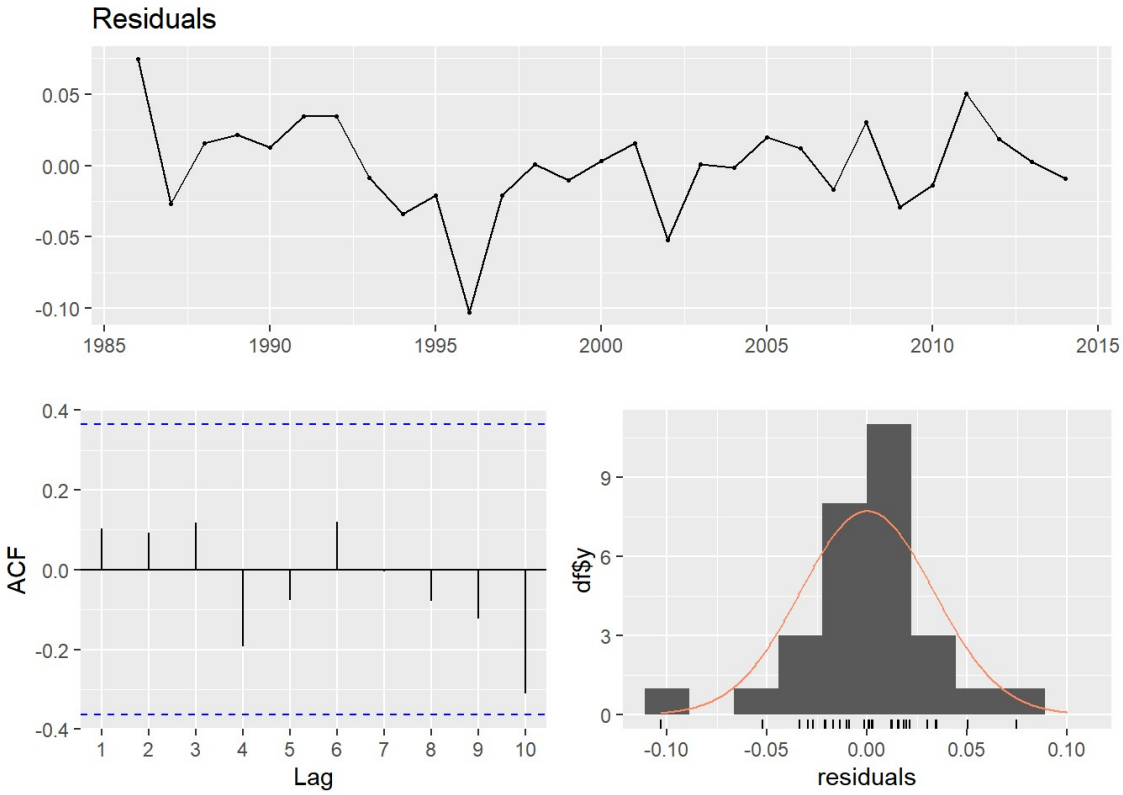



```
##
##  Breusch-Godfrey test for serial correlation of order up to 8
##
## data:  Residuals
## LM test = 3.2286, df = 8, p-value = 0.9192
```

```
model6 = dynlm(Y.t ~ L(Y.t , k = 1 ) + L(Y.t , k = 2 ) + S.t.1)
summary(model6)
```

```
##
## Time series regression with "ts" data:
## Start = 1986, End = 2014
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, k = 1) + L(Y.t, k = 2) + S.t.1)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.103024 -0.017057  0.001085  0.018410  0.074655
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.57255    0.27662   2.070    0.049 *
## L(Y.t, k = 1)  0.09903    0.23833   0.416    0.681
## L(Y.t, k = 2)  0.16135    0.23263   0.694    0.494
## S.t.1         -0.04473    0.02890  -1.548    0.134
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03524 on 25 degrees of freedom
## Multiple R-squared:  0.4771, Adjusted R-squared:  0.4143
## F-statistic: 7.602 on 3 and 25 DF,  p-value: 0.0008917
```

```
checkresiduals(model6)
```

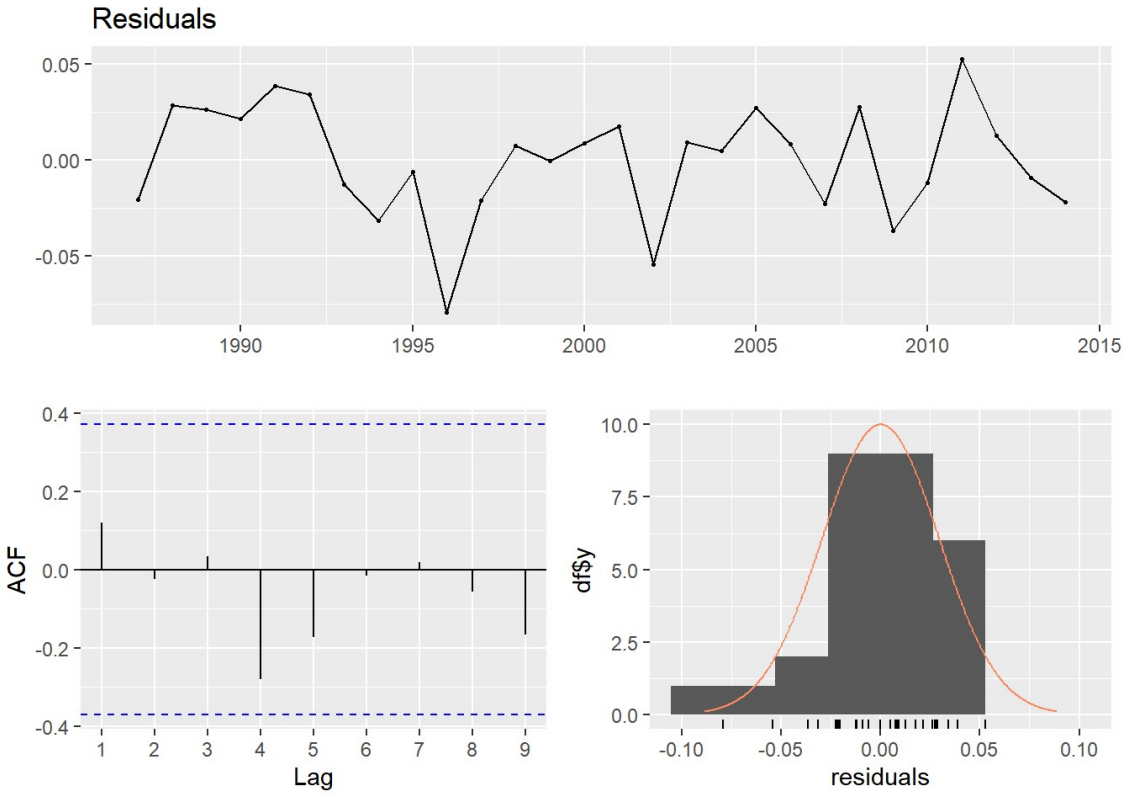



```
##
##  Breusch-Godfrey test for serial correlation of order up to 7
##
## data:  Residuals
## LM test = 7.5048, df = 7, p-value = 0.3783
```

```
model7 = dynlm(Y.t ~ L(Y.t , k = 1 ) + L(Y.t , k = 2 ) + L(Y.t , k = 3 )
               + S.t.1)
summary(model7)
```

```
##
## Time series regression with "ts" data:
## Start = 1987, End = 2014
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, k = 1) + L(Y.t, k = 2) + L(Y.t,
##     k = 3) + S.t.1)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.079352 -0.020435  0.006226  0.022716  0.052732
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.165271   0.355572   0.465    0.646
## L(Y.t, k = 1) 0.322634   0.232432   1.388    0.178
## L(Y.t, k = 2) 0.325156   0.221039   1.471    0.155
## L(Y.t, k = 3) 0.120302   0.217785   0.552    0.586
## S.t.1         0.001292   0.036445   0.035    0.972
##
## Residual standard error: 0.032 on 23 degrees of freedom
## Multiple R-squared:  0.5085, Adjusted R-squared:  0.4231
## F-statistic:  5.95 on 4 and 23 DF,  p-value: 0.00194
```

```
checkresiduals(model7)
```

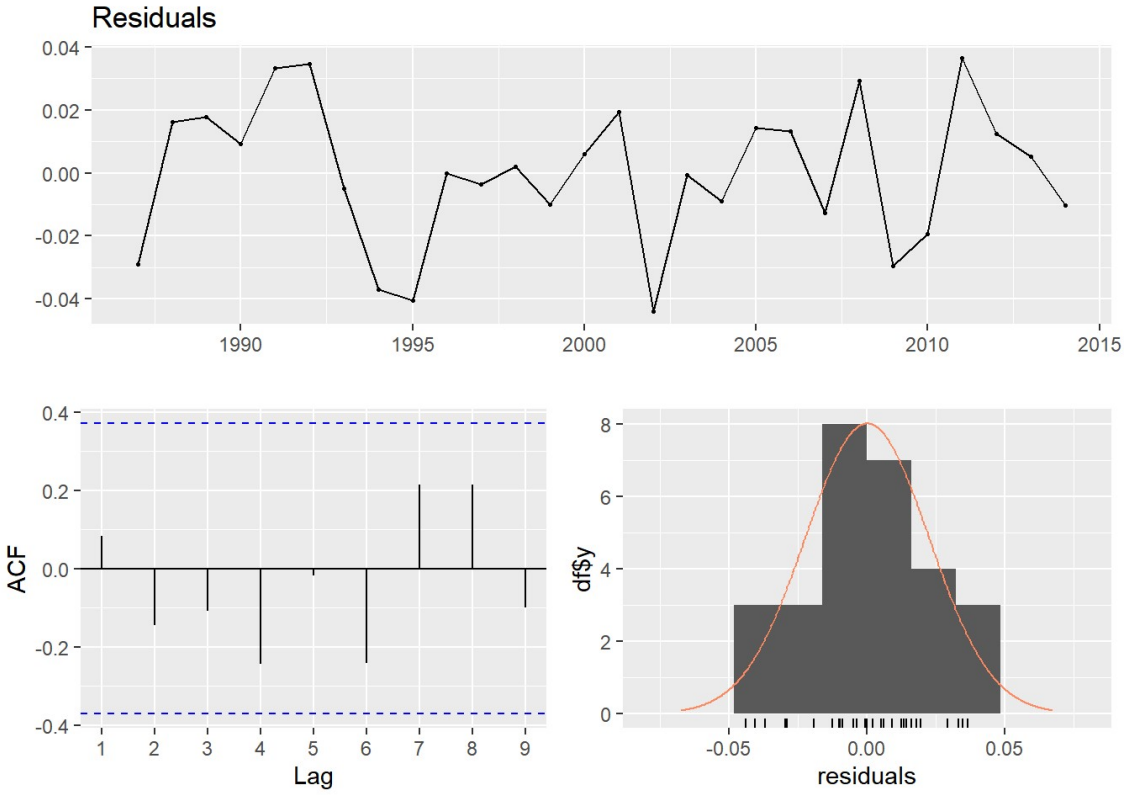



```
##
##  Breusch-Godfrey test for serial correlation of order up to 8
##
## data:  Residuals
## LM test = 6.9961, df = 8, p-value = 0.5371
```

```
model8 = dynlm(Y.t ~ L(Y.t , k = 1 ) + L(Y.t , k = 2 ) +
                 L(Y.t , k = 3 ) + S.t + S.t.1 + trend(Y.t))
summary(model8)
```

```
##
## Time series regression with "ts" data:
## Start = 1987, End = 2014
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, k = 1) + L(Y.t, k = 2) + L(Y.t,
##     k = 3) + S.t + S.t.1 + trend(Y.t))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.044049 -0.010746  0.001046  0.014841  0.036573
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.982617   0.394351   2.492  0.02115 *
## L(Y.t, k = 1) -0.099472   0.233822  -0.425  0.67486
## L(Y.t, k = 2) -0.074252   0.220238  -0.337  0.73936
## L(Y.t, k = 3) -0.096569   0.192328  -0.502  0.62082
## S.t           -0.134072   0.035254  -3.803  0.00104 **
## S.t.1          0.020597   0.041647   0.495  0.62605
## trend(Y.t)     0.001614   0.001377   1.172  0.25414
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02551 on 21 degrees of freedom
## Multiple R-squared:  0.7149, Adjusted R-squared:  0.6334
## F-statistic: 8.774 on 6 and 21 DF,  p-value: 7.469e-05
```

```
checkresiduals(model8)
```

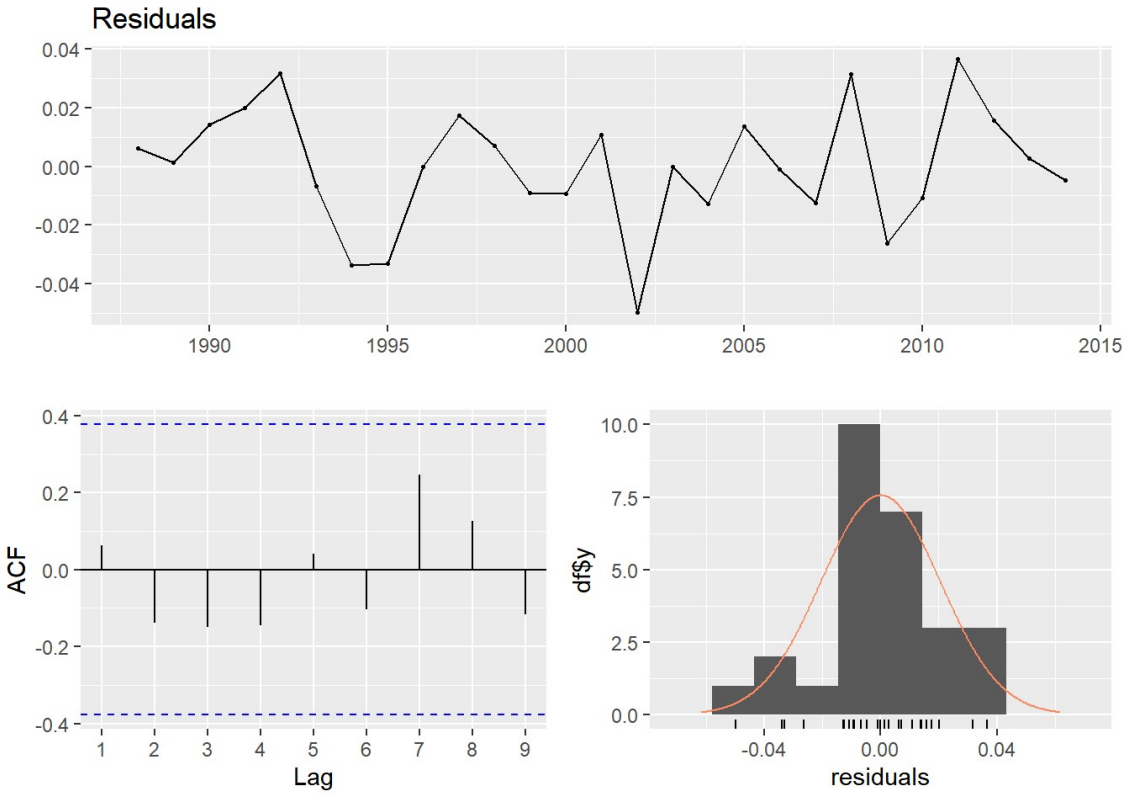



```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 18.295, df = 10, p-value = 0.05019
```

```
model9 = dynlm(Y.t ~ L(Y.t , k = 1 ) + L(Y.t , k = 2 ) +
                 L(Y.t , k = 3 ) + L(Y.t , k = 4 ) + S.t + S.t.1 + trend(Y.t))
summary(model9)
```

```
##
## Time series regression with "ts" data:
## Start = 1988, End = 2014
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, k = 1) + L(Y.t, k = 2) + L(Y.t,
##     k = 3) + L(Y.t, k = 4) + S.t + S.t.1 + trend(Y.t))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.04991 -0.01002  0.00000  0.01395  0.03674
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.251544   0.406140   3.082 0.006142 **
## L(Y.t, k = 1) -0.021892   0.233576  -0.094 0.926307
## L(Y.t, k = 2) -0.161943   0.213999  -0.757 0.458483
## L(Y.t, k = 3) -0.178848   0.186043  -0.961 0.348462
## L(Y.t, k = 4) -0.238617   0.159391  -1.497 0.150807
## S.t           -0.131867   0.033506  -3.936 0.000887 ***
## S.t.1         -0.004797   0.043191  -0.111 0.912731
## trend(Y.t)     0.001192   0.001319   0.904 0.377537
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02406 on 19 degrees of freedom
## Multiple R-squared:  0.7693, Adjusted R-squared:  0.6842
## F-statistic: 9.049 on 7 and 19 DF,  p-value: 6.317e-05
```

```
checkresiduals(model9)
```



```
##
##  Breusch-Godfrey test for serial correlation of order up to 11
##
## data:  Residuals
## LM test = 9.5081, df = 11, p-value = 0.5751
```

```
aic = AIC(model1, model2, model3, model4, model5, model6, model7, model8, model9)
bic = BIC(model1, model2, model3, model4, model5, model6, model7, model8, model9)

aic[order(aic$AIC),]
```

```
##        df       AIC
## model1  5 -124.6894
## model3  6 -123.7422
## model4  4 -123.5867
## model2  5 -120.9662
## model5  6 -120.5453
## model8  8 -118.0505
## model9  9 -116.1286
## model7  6 -106.8079
## model6  5 -106.0480
```

```
bic[order(bic$BIC),]
```

```
##        df        BIC
## model4  4 -117.98191
## model1  5 -117.68342
## model3  6 -115.33501
## model2  5 -114.12975
## model5  6 -112.34153
## model8  8 -107.39291
## model9  9 -104.46604
## model6  5  -99.21156
## model7  6  -98.81468
```

```
modelno <- c("Model 1", "Model 2", "Model 3", "Model 4", "Model 5", "Model 6", "Model 7", "Model 8", "Model
9")
models <- c("Y(1) + S(t) + trend", "Y(2) + S(t), trend", "Y(1) + S(t) + S(1), trend", "Y(t) + S(t)", "Y(1) +
Y(2) + S(t) + S(1)", "Y(1) + Y(2) + S(1)", "Y(1) + Y(2) + Y(3) + S(1)", "Y(1) + Y(2) + Y(3) + s(t) + S(1) + t
rend", "Y(1) + Y(2) + Y(3) + Y(4) + s(t) + S(1) + trend")
p <- c("2.073e-06", "1.688e-06", "6.329e-06", "1.387e-06", "4.286e-06", "0.0008917", "0.00194", "7.469e-05",
"6.317e-05")
rsq <- c("0.6297", "0.6498", "0.6281", "0.6045", "0.6546", "0.4143", "0.4231", "0.6334", "0.6842")
bg <- c("0.521","0.87","0.3165","0.8814","0.9192","0.3783","0.5371","0.05019","0.5751")

s<- data.frame(cbind(modelno, models, p, rsq, bg))
colnames(s)<- c("**Model**", "Description", "**Model p-value**", "**R-squared value**","**Breusch-Godfrey p-v
alue**")

s %>% kbl(caption = "**Model Summary**") %>%
  kable_classic(full_width = F, html_font = "Cambria")
```

Model Summary

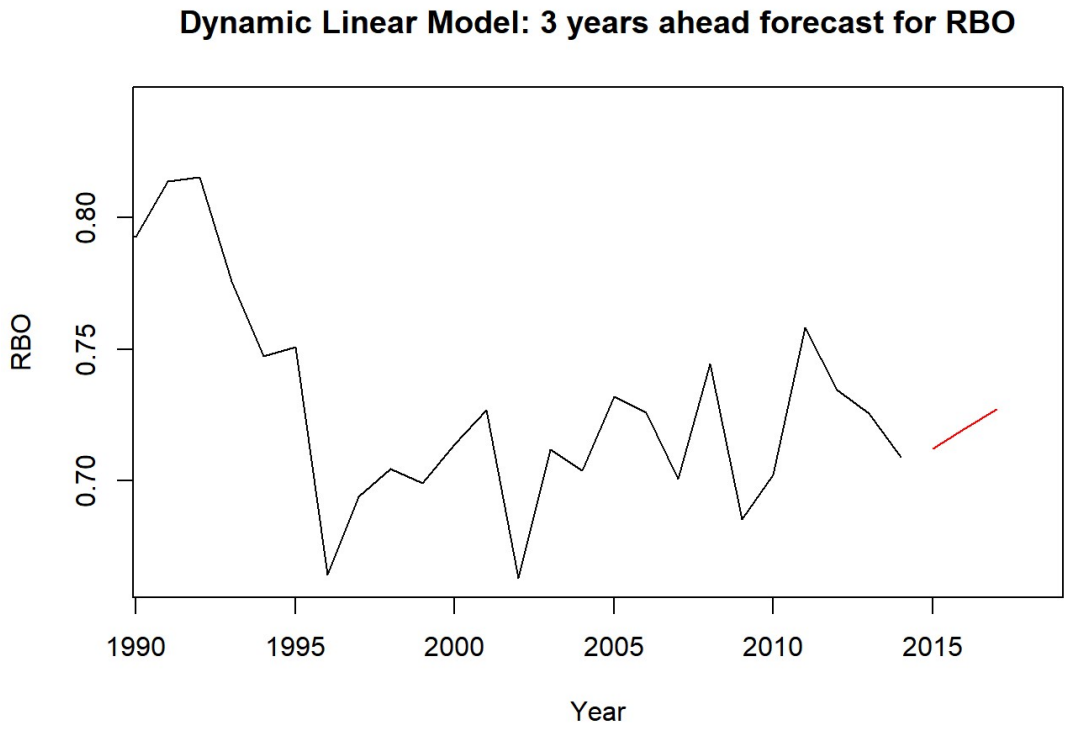
Model	Description	**Model p-value**	**R-squared value**	**Breusch-Godfrey p-value**
Model 1	Y(1) + S(t) + trend	2.073e-06	0.6297	0.521

Model	Description	**Model p-value**	**R-squared value**	**Breusch-Godfrey p-value**
Model 2	Y(2) + S(t), trend	1.688e-06	0.6498	0.87
Model 3	Y(1) + S(t) + S(1), trend	6.329e-06	0.6281	0.3165
Model 4	Y(t) + S(t)	1.387e-06	0.6045	0.8814
Model 5	Y(1) + Y(2) + S(t) + S(1)	4.286e-06	0.6546	0.9192
Model 6	Y(1) + Y(2) + S(1)	0.0008917	0.4143	0.3783
Model 7	Y(1) + Y(2) + Y(3) + S(1)	0.00194	0.4231	0.5371
Model 8	Y(1) + Y(2) + Y(3) + s(t) + S(1) + trend	7.469e-05	0.6334	0.05019
Model 9	Y(1) + Y(2) + Y(3) + Y(4) + s(t) + S(1) + trend	6.317e-05	0.6842	0.5751

Based on the results obtained, Model 9 is selected as the best model from the list as it provides the best compromise between low p-value, relatively high R-squared value, relatively low AIC and BIC, and a relatively high p-value for the Breusch-Godfrey test implying that it does not contain serial correlation. The intercept and the step function S(t) are significant in Model 9.

The forecast for Model 9 is created below, and shows that the RBO is expected to climb between 2015 and 2018 in the 3 years ahead forecast.

```
q = 3
n = nrow(model9$model)

rbo.frc = array(NA , (n + q))
rbo.frc[1:n] = Y.t[5:length(Y.t)]

trend = array(NA,q)
trend.start = model9$model[n,"trend(Y.t)"]
trend = seq(trend.start , trend.start + q, 1)

for (i in 1:q){
  data.new = c(1,rbo.frc[n-1+i],rbo.frc[n-2+i],
               rbo.frc[n-3+i],rbo.frc[n-3+i],1,1,trend[i])
  rbo.frc[n+i] = as.vector(model9$coefficients) %*% data.new
}

plot(Y.t,xlim=c(1991,2018),ylab='RBO',xlab='Year',
     main = "Dynamic Linear Model: 3 years ahead forecast for RBO")
lines(ts(rbo.frc[(n+1):(n+q)],start=c(2015,1),frequency = 1),col="red")
```



Conclusion

This report concerned the implementation of various forecasting models in R for three different time series forecasting tasks and validating each of those models to identify the best forecasting model for each application. The selection of models was based on multiple test statistics updating from the model summaries and residuals, including MASE, p-value, AIC, BIC, and the results of serial correlation tests.

The first task involved finding the ideal forecasting model to predict mortality rates in Paris, France based on several climate variables which acted as predictors for the mortality rates time series. The data set provided spanned between the years 2010 till 2020 and was weekly in frequency. Where possible, multivariate models were implemented. Polynomial, Koyck, finite DLM, exponential smoothing, and state space models were implemented and the ideal implementation was found to be either the Simple SES or the ANN state space model, since both model demonstrated similar performance.

The second task involved finding the ideal forecasting model to predict the first flowering day, or FFD, which is a variable that is impacted by several climate conditions such as rainfall, temperature, radiation, and relative humidity. The data set spanned between the years 1984 and 2014 and was annual in frequency. For this task, only univariate models were explored. Polynomial, Koyck, finite DLM, exponential smoothing, and state space models were implemented and the ideal implementation was found to be the Additive Damped Trend Holt SES model due to its relatively low MASE as well displaying improved performance compared to the other models.

Part A of the third task involved finding the ideal forecasting model to predict the Rank-based Order similarity metric, or RBO, which is a variable impacted by climate conditions including temperature, rainfall, radiation, and relative humidity, which act as predictors for RBO. The dataset spanned between the years 1983 and 2014 and was annual in frequency. For this task,

only univariate models were explored. Polynomial, Koyck, finite DLM, exponential smoothing, and state space models were implemented and the ideal implementation was found to be the ARDL model using the temperature variable as a predictor for RBO.

Part B of the third task involved the same data set as part A of task 3, but given the additional context of a drought that took place between 1996 till 2009. This information allows for the implementation of a dynamic linear model with an intervention point set to 1996. Various combinations of Y(t), step functions, trend, as well as their lags were used in the implementations and their summaries and residuals were studied. The ideal dynamic linear model was identified to be the 9th model that was tested, which incorporated the first four lags of Y(t), the step function and its first lag, and trend in the final model.