MATH1307 Forecasting Assignment 2

Wong Yi Wei S3966890

2023-09-24

1.0 Task 1

1.1.0 Introduction

The aim of this investigation is to analyse and forecast the amount of horizontal solar radiation reaching the ground at a particular location over the globe. We will be using the dataset `data 1` which contains the monthly average horizontal solar radiation and the monthly precipitation series measured between January 1960 and December 2014. We will also be using the dataset `data x` for forecasting purposes, which contains the precipitation measurements (predictor series) for the months January 2015 to December 2016 at the same locations.

To perform these investigations, we will first prep the data for analysis and do further testing through various analysis methods.

1.2.0 Data Description and Preprocessing

The dataset `data 1` is used for analysis and contains the monthly average horizontal solar radiation and the monthly precipitation series measured between January 1960 and December 2014. Furthermore we will also use the dataset `data x` which contains the precipitation measurements (predictor series) for the months January 2015 to December 2016 at the same locations.

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

```
## Rows: 660 Columns: 2
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (2): solar, ppt
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
class(data1)
```

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

```
str(data1)
```

```
## spc_tbl_ [660 × 2] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ solar: num [1:660] 5.05 6.42 10.85 16.93 24.03 ...
##  $ ppt  : num [1:660] 1.333 0.921 0.947 0.615 0.544 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   solar = col_double(),
##   ..   ppt = col_double()
```

```
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
head(data1)
```

```
## # A tibble: 6 × 2
##   solar   ppt
##   <dbl> <dbl>
## 1  5.05 1.33
## 2  6.42 0.921
## 3 10.8  0.947
## 4 16.9  0.615
## 5 24.0  0.544
## 6 26.3  0.703
```

```
datax<-read_csv("data.x.csv")
```

```
## Rows: 24 Columns: 1
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (1): x
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
class(datax)
```

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

```
str(datax)
```

```
## spc_tbl_ [24 × 1] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ x: num [1:24] 0.189 0.697 0.595 0.487 0.262 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   x = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
head(datax)
```

```
## # A tibble: 6 × 1
##       x
##   <dbl>
## 1 0.189
## 2 0.697
## 3 0.595
## 4 0.487
## 5 0.262
## 6 0.809
```

From the code chunk above, we will first import the first dataset into R using the `read_csv` function and assign the dataset with the neme `data1`. Then we will check the class, structure, and view the first few observations of the data with the functions `class`, `str`, and `head`. We can see here that the variables are all in numerical variable type which is what we want for analysis.

Similar to `data1`, we will use the same methods for the second dataset and assign it into the name `datax`. We can also see here that the variables are all in numerical variable type which is what we want for analysis.

```
#Convert data to time series
solar.ts<-ts(data1$solar, start=c(1960,1), frequency = 12)
precip.ts<-ts(data1$ppt, start=c(1960,1), frequency = 12)
data1.ts<-ts(data1, start=c(1960,1), frequency=12)
datax.ts<-ts(datax, start=c(2015,1), frequency = 12)
```

To process the data for further analysis, we must first convert the objects in the data frame `data1` to time series objects. By using the `ts` function, we convert each variables in the dataset into time series objects and name the new objects `solar.ts`, `precip.ts`, `data1.ts`, and `datax.ts`. Since we know that the dataset starts from January 1960 and it contains monthly values, we can write the arguments `start` to 1960 and 1, and `frequency` to 12 for all conversions.

1.3.0 Data Visualisation

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

1.3.1 Solar Radiation Series

```
#Time series plot (Solar Radiation)
plot(solar.ts, main="Time Series Plot for Solar Radiation Series", ylab="Average Solar
         Radiation", xlab="Year")
points(solar.ts, x=time(solar.ts), pch=as.vector(season(solar.ts)))
```

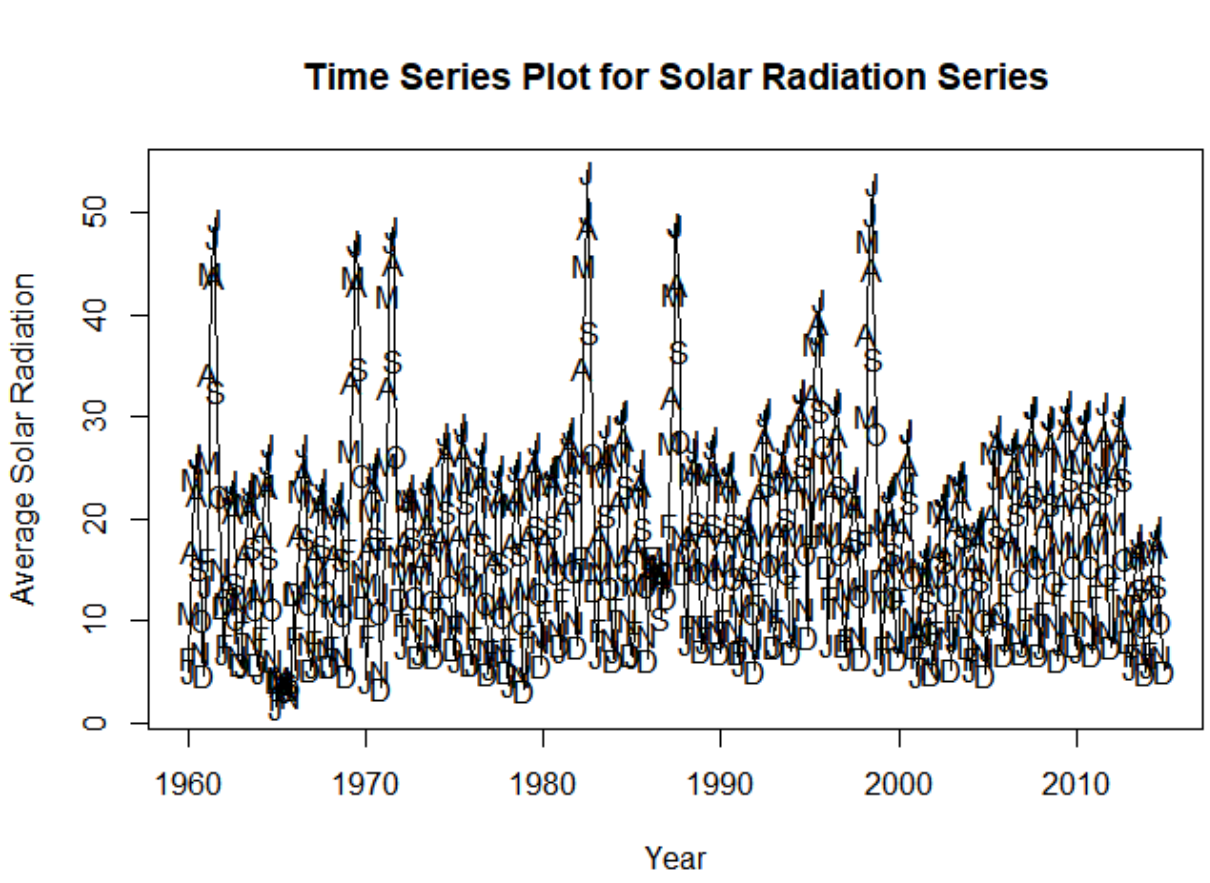

Figure 1

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

- The series has no apparent trends and intervention points
- There is obvious seasonality, with higher values in June and July, and lower values during December and January.
- Since there are seasonality, changing variance and moving average behavior is not obvious.

```
#ACF plot (Solar Radiation)
acf(solar.ts, lag.max = 48, main="Sample ACF Plot for Solar Radiation Series")
```

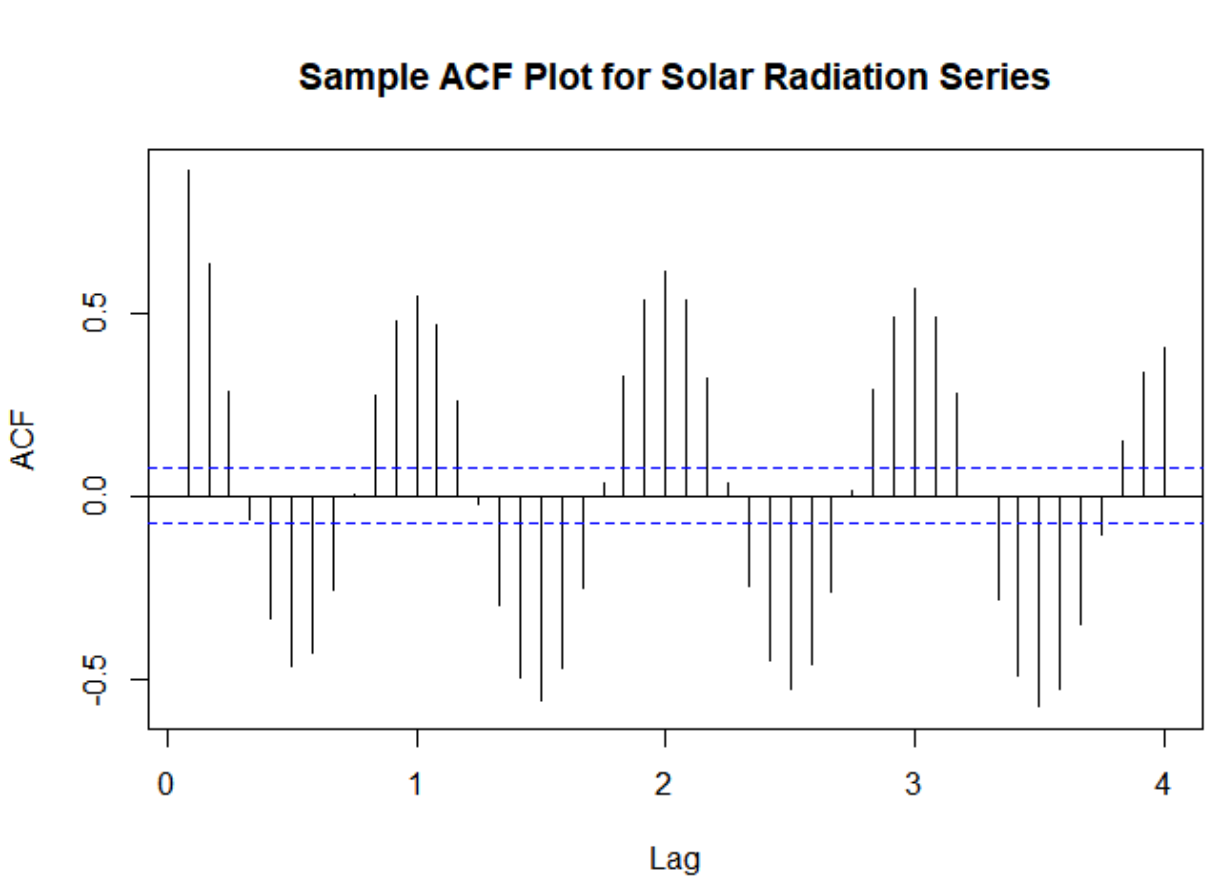

Figure 2

`Figure 2` shows the sample ACF plot for solar radiation series. Here we can observe that it shows strong seasonal patterns and has no apparent trend in the series.

```
#Augmented Dickey-Fuller Test (Solar Radiation)
adf.test(solar.ts, k=ar(solar.ts)$order)
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  solar.ts
## Dickey-Fuller = -4.557, Lag order = 25, p-value = 0.01
## alternative hypothesis: stationary
```

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

1.3.2 Precipitation Series

```
#Time series plot (Precipitation)
plot(precip.ts, main="Time Series Plot for Precipitation Series", ylab="Average Precipitation",
         xlab="Year")
```

```
points(precip.ts, x=time(precip.ts), pch=as.vector(season(precip.ts)))
```

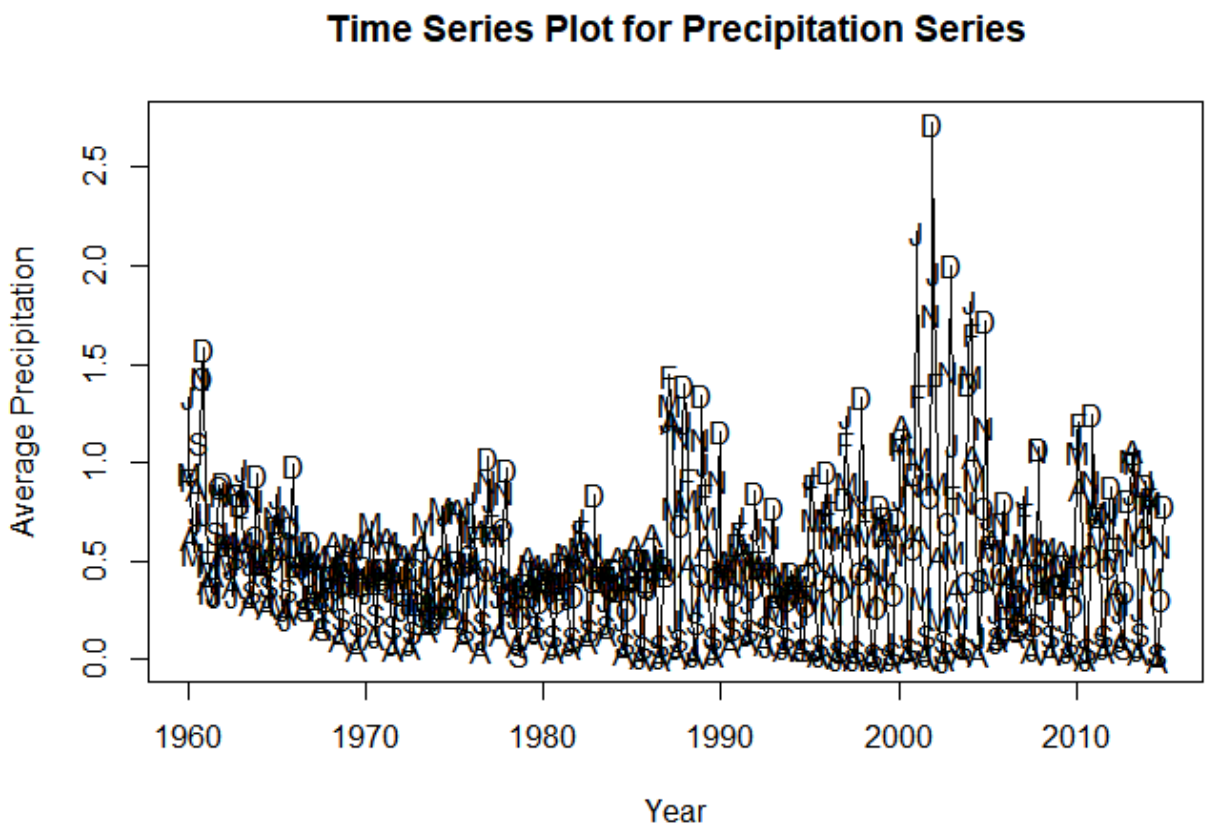

Figure 3

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

- The series has no apparent trends and intervention points
- There is seasonality, with higher values in December, and lower values during April.
- Since there are seasonality, changing variance and moving average behavior is not obvious.

```
#ACF plot (Precipitation)
acf(precip.ts, lag.max = 48, main="Sample ACF Plot for Precipitation Series")
```

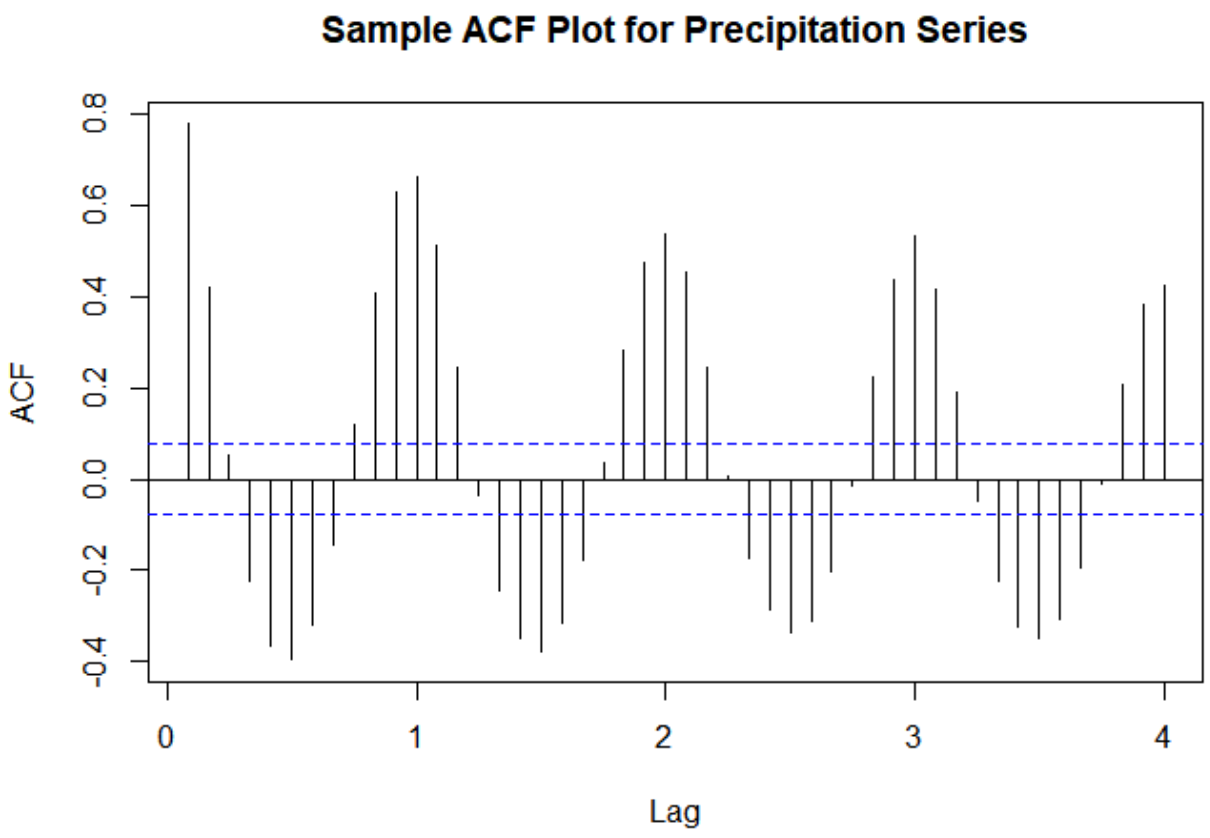

Figure 4

`Figure 4` shows the sample ACF plot for precipitation series. Here we can observe that it shows decaying patterns and suggesting that there could be trend in the series. Furthermore we can see that there are obvious seasonality in the plot.

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

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  precip.ts
## Dickey-Fuller = -9.1249, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

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

1.3.3 Scaled Solar Radiation & Precipitation Series

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

```
plot(data1.scaled,plot.type="s",col=c("red","blue"),main="Time Series Plot of Scaled Solar
        Radiation and Precipitation Series")
legend("topleft",lty=1,col=c("red","blue"),c("Solar Radiation Series","Precipitation Series"))
```

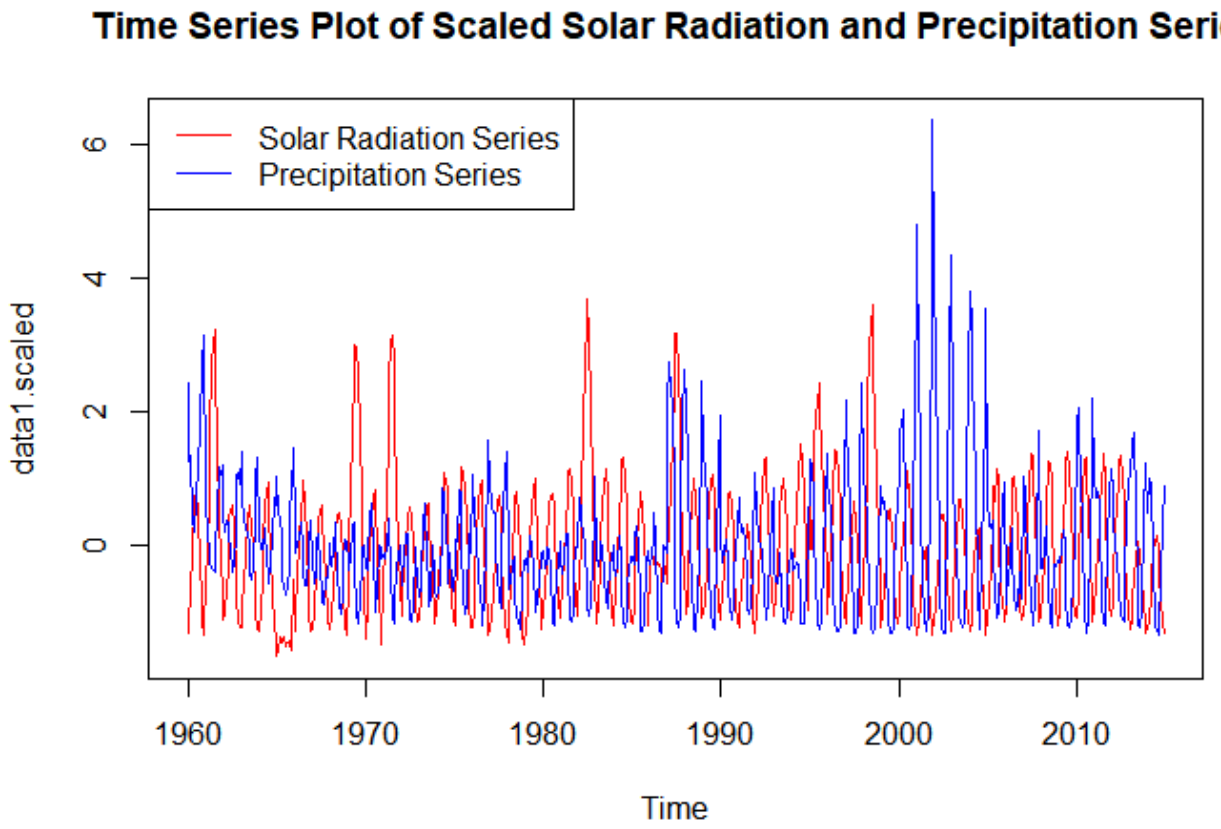

Figure 5

`Figure 5` contains a time series plot of all solar radiation and precipitation series. To compare these data and fit them in a graph, scaling and centering is done to the data by using the `scale` function.

1.4.0 Decomposition

To look into the impact of the components in the time series data, we will use various decomposition methods to have a further insight of the independent seasonal and trend characteristics of the time series data. In our case, we will use the X-12-ARIMA decomposition, as well as the STL decomposition methods to decompose the data. We will not be using the classical decomposition method as there are other methods more efficient than this method.

In the first part, we will use the X-12-ARIMA decomposition as this method is an upgrade from the classical decomposition method as it accounts for day variation, holiday effects and other predictors. We will use the functions `x12` to decompose and visualize the time series data.

1.4.1 Solar Radiation Series

```
solar.x12<-x12(solar.ts)
plot(solar.x12, sa=TRUE,trend=TRUE, main="X-12-ARIMA Decomposition for Solar Radiation Series")
```

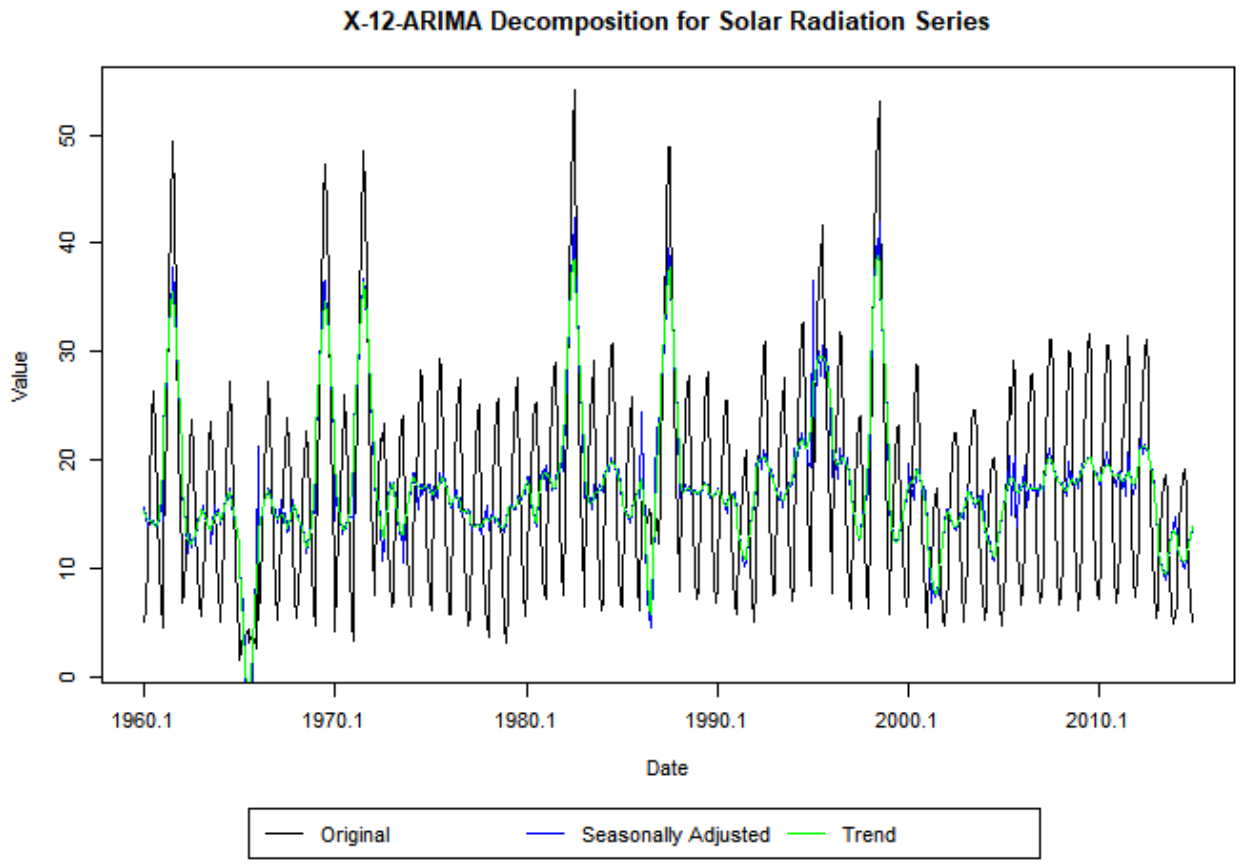

Figure 6

`Figure 6` shows the X12 decomposition of the solar radiation series. The arguments `sa=TRUE` ,and `trend=TRUE` allows us to seasonally adjust the data, and show the trend line. Here we can observe that the seasonally adjusted series does not closely follow the original series of the solar radiation values. Furthermore, the trend line lies closely to the seasonally adjusted series.

```
plotSeasFac(solar.x12)
```

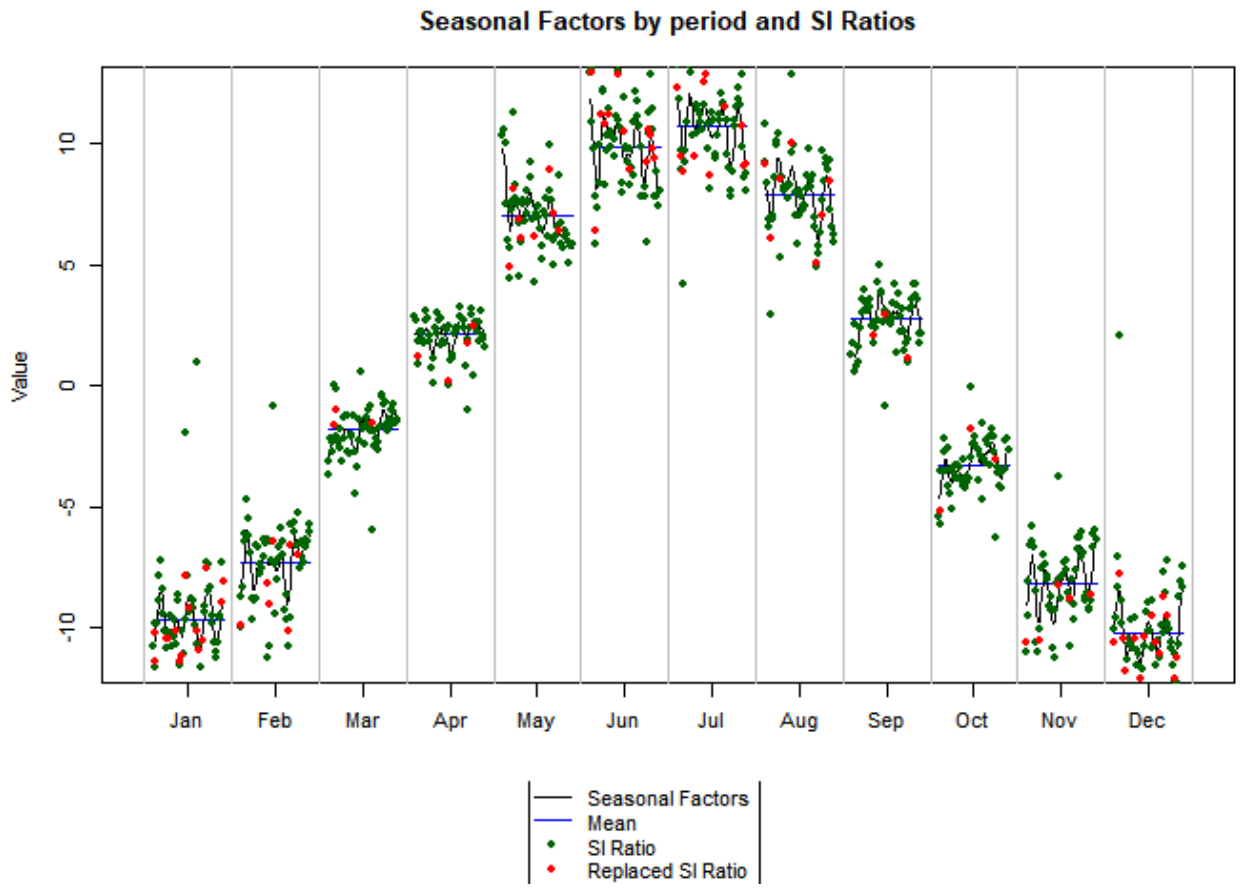

Figure 7

`Figure 7` shows the seasonal factor plots and SI ratios by period for solar radiation series. Here we can observe that for most of the months the seasonal factors do not divert from the mean level. Furthermore, we can observe from the SI ratios that there are little outliers in the plot.

```
solar.stl<-stl(solar.ts,t.window = 15,s.window="periodic",robust=TRUE)
plot(solar.stl)
```

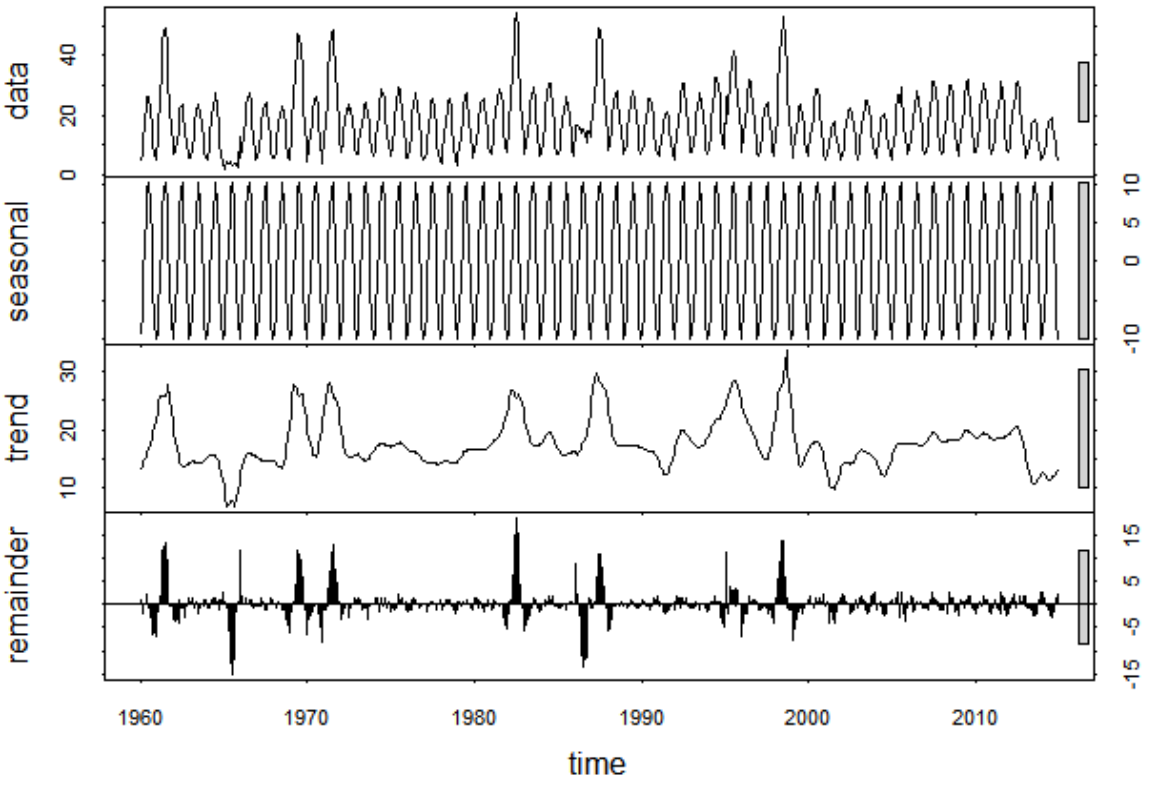

Figure 8

`Figure 8` shows the STL graph for solar radiation, where we can observe that the there is seasonal effect.

1.4.2 Precipitation Series

```
precip.x12<-x12(precip.ts)
plot(precip.x12, sa=TRUE,trend=TRUE, main="X-12-ARIMA Decomposition for Precipitation Series")
```

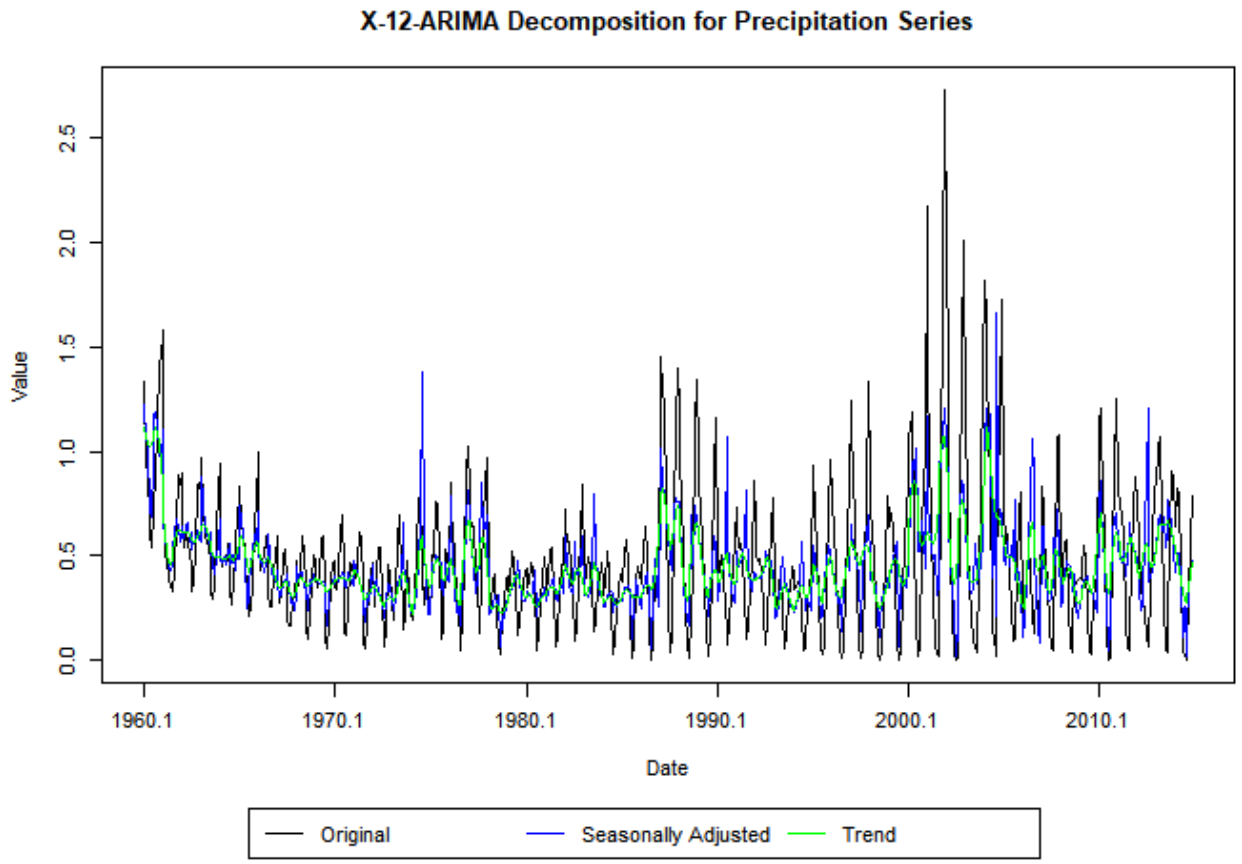

Figure 9

`Figure 9` shows the X12 decomposition of the precipitation series. The arguments `sa=TRUE` ,and `trend=TRUE` allows us to seasonally adjust the data, and show the trend line. Here we can observe that the seasonally adjusted series does not closely follow the original series of the precipitation values. Furthermore, the trend line shows fluctuations from 2000 onwards.

```
plotSeasFac(precip.x12)
```

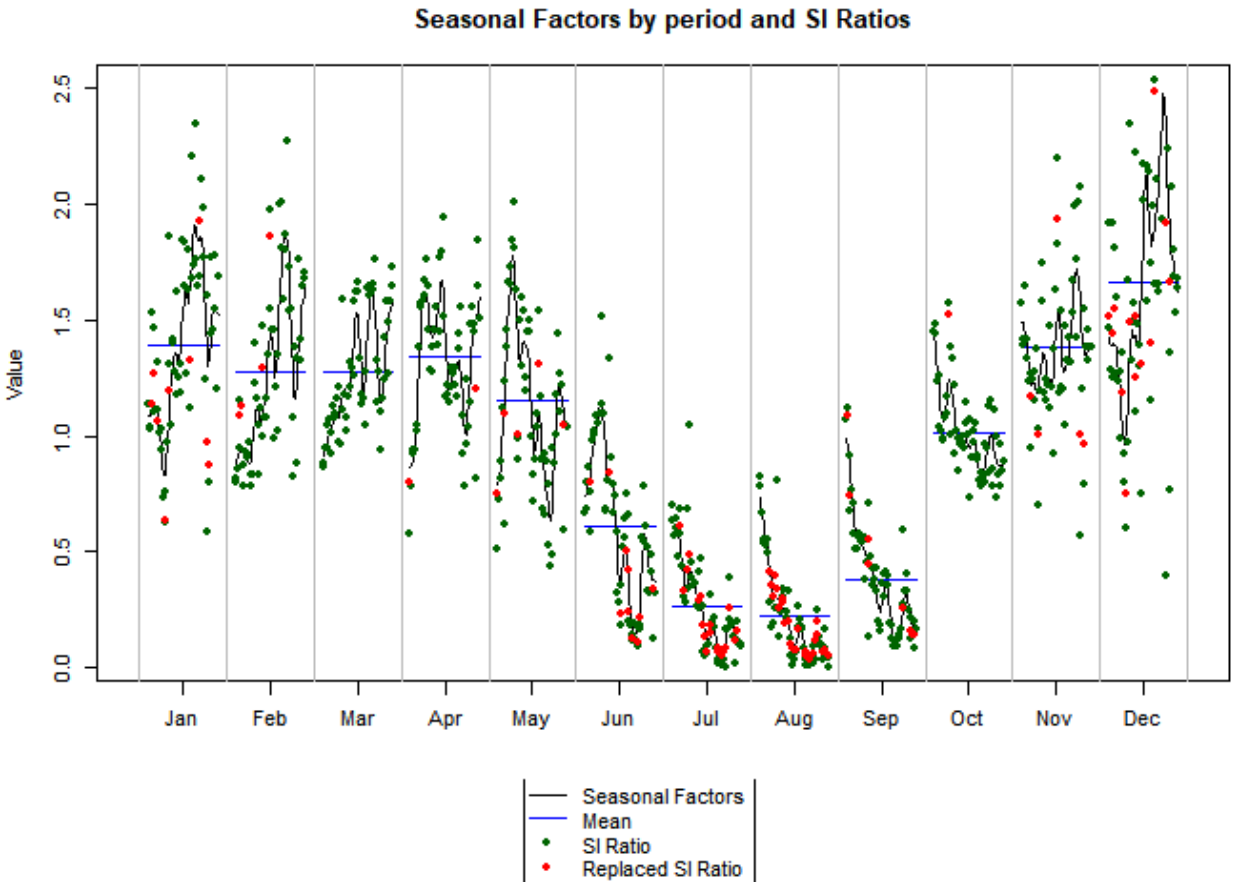

Figure 10

`Figure 10` shows the seasonal factor plots and SI ratios by period for precipitation series. Here we can observe that for most of the months the seasonal factors divert from the mean level. Furthermore, we can observe from the SI ratios that there are many outliers in the plot.

```
precip.stl<-stl(precip.ts,t.window = 15,s.window="periodic",robust=TRUE)
plot(precip.stl)
```

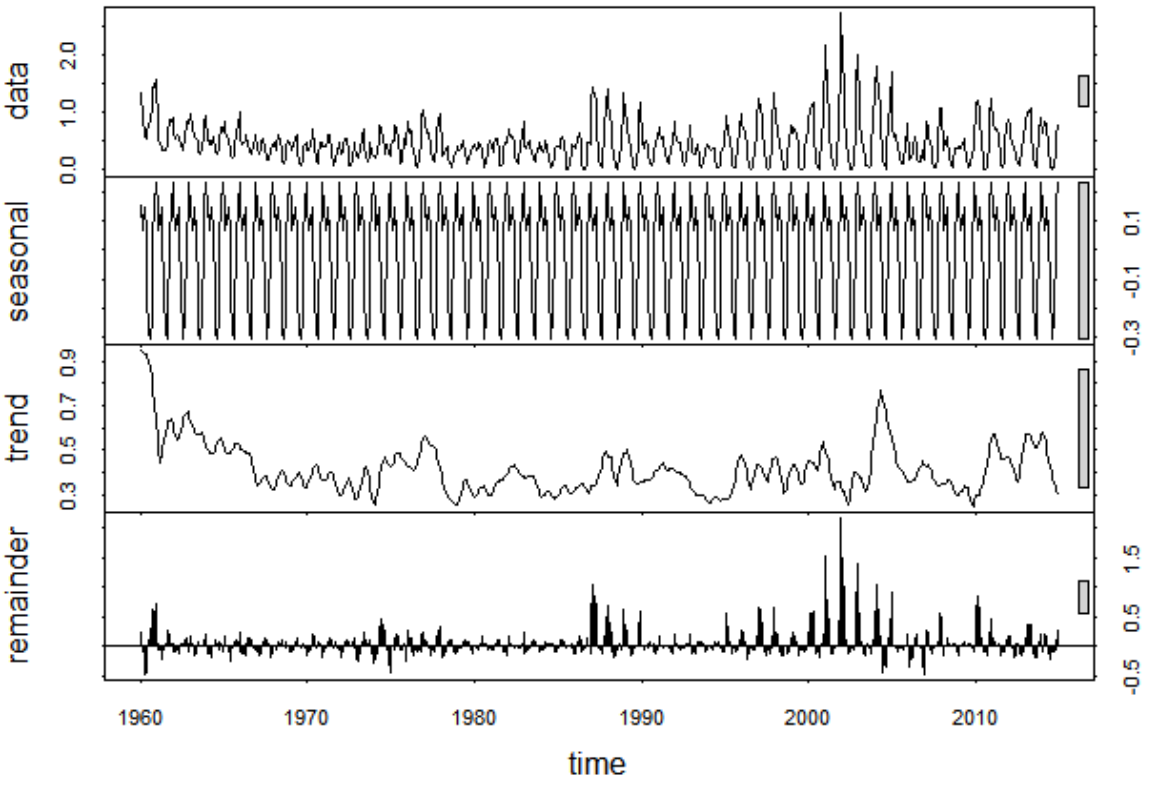

Figure 11

`Figure 11` shows the STL graph for solar radiation, where we can observe that the there is seasonal effect.

1.5.0 Time Series Regression Models

In this section, we will first look at the correlation matrix for solar radiation and precipitation by using the `cor` function.

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

```
##             solar        ppt
## solar  1.0000000 -0.4540277
## ppt   -0.4540277  1.0000000
```

From the correlation matrix output above, we can observe that there is a moderate negative correlation between solar radiation and precipitation. Thus, solar radiation should be used as dependent variable y and precipitation as independent variable x.

1.5.1 Finite Distributed Lag Model

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

```r
for (i in 12){
  model1<-dlm(x=data1$ppt, y=data1$solar, q=i)
  cat("q =",i,"AIC =",AIC(model1$model),"BIC =",BIC(model1$model),"MASE=", MASE(model1)$MASE,
        "\n")
}
```

```
## q = 12 AIC = 4578.787 BIC = 4645.895 MASE= 1.5516
```

```r
model1.2<-dlm(x=data1$ppt, y=data1$solar, q=12)
summary(model1.2)
```

```
##
## Call:
## lm(formula = model.formula, data = design)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.563  -5.239  -0.796   4.137  32.430
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19.5164     1.1151  17.501  < 2e-16 ***
## x.t          -5.8876     1.9508  -3.018  0.00265 **
## x.1           0.9993     2.5647   0.390  0.69694
## x.2           0.4343     2.5571   0.170  0.86520
## x.3           1.8763     2.5580   0.734  0.46352
## x.4           1.7459     2.5587   0.682  0.49529
## x.5           3.3279     2.5601   1.300  0.19410
## x.6           0.7751     2.5617   0.303  0.76230
## x.7           1.7937     2.5615   0.700  0.48402
## x.8           0.2827     2.5593   0.110  0.91207
## x.9          -1.1022     2.5615  -0.430  0.66712
## x.10         -1.9333     2.5508  -0.758  0.44880
## x.11         -0.5613     2.5532  -0.220  0.82605
## x.12         -5.3492     1.9216  -2.784  0.00553 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.181 on 634 degrees of freedom
## Multiple R-squared:  0.3216, Adjusted R-squared:  0.3077
## F-statistic: 23.12 on 13 and 634 DF,  p-value: < 2.2e-16
##
## AIC and BIC values for the model:
##        AIC      BIC
## 1 4578.787 4645.895
```

From the output above, we can see that the recommended amount of lag is 12. Hence, we used the `dlm` function with a argument `q=12` . By using the `summary` function, we can observe that in `model1.2` the

lag weights of the predictors are significant at the 5% statistical level as p-value < 0.05. Furthermore, there is a adjusted R-square value of 0.3077, which accounts for 30.77% of the variability in the data.

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

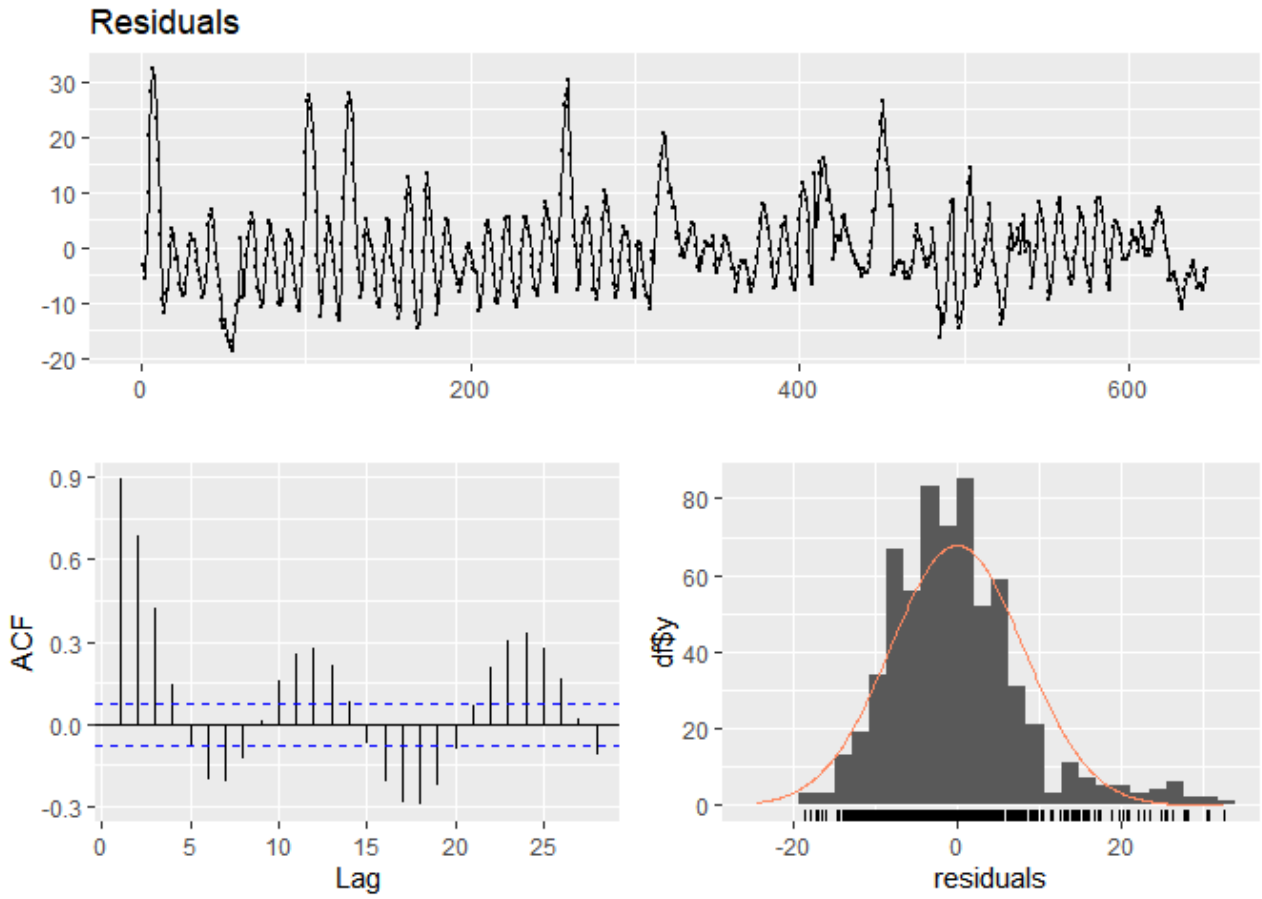

Figure 12

```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 1045.6, df = 10, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 10
```

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

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model1.2$model
## LM test = 527.93, df = 1, p-value < 2.2e-16
```

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

```
##
##  Shapiro-Wilk normality test
##
## data:  model1.2$model$residuals
## W = 0.94168, p-value = 2.922e-15
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 12`. The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are highly significant. Furthermore, from the Shapiro-Wilk test we can conclude that there is enough evidence to reject null hypothesis of normality as p-value < 0.05.

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

```
##      x.t      x.1      x.2      x.3      x.4      x.5      x.6      x.7
## 4.432762 7.774629 7.820758 7.914873 7.941510 7.944820 7.943359 7.929999
##      x.8      x.9     x.10     x.11     x.12
## 7.916836 7.921508 7.867385 7.889225 4.508273
```

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

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

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

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

1.5.2 Polynomial Distributed Lag Model

Polynomial DLM imposes a polynomial shape on the lag distribution to reduce the effect of multicollinearity. For the Polynomial DLM, we will be using precipitation as our independent variable, and solar radiation as our dependent variable.

```
model2.1<-polyDlm(x=as.vector(data1$ppt),y=as.vector(data1$solar),q=12,k=2,show.beta=TRUE)
```

```
## Estimates and t-tests for beta coefficients:
##         Estimate Std. Error t value  P(>|t|)
## beta.0    -1.820      0.393   -4.63 4.41e-06
```

```
## beta.1    -0.402      0.311   -1.29 1.97e-01
## beta.2     0.702      0.261    2.69 7.42e-03
## beta.3     1.490      0.240    6.22 9.28e-10
## beta.4     1.970      0.237    8.31 5.71e-16
## beta.5     2.130      0.239    8.89 6.28e-18
## beta.6     1.970      0.240    8.22 1.18e-15
## beta.7     1.510      0.237    6.36 3.90e-10
## beta.8     0.726      0.232    3.13 1.84e-03
## beta.9    -0.370      0.233   -1.58 1.14e-01
## beta.10   -1.780      0.254   -7.01 5.94e-12
## beta.11   -3.500      0.304  -11.50 4.37e-28
## beta.12   -5.540      0.386  -14.30 1.31e-40
```

```
summary(model2.1)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.689  -5.452  -0.686   4.129  32.797
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 19.22679    1.10784  17.355  < 2e-16 ***
## z.t0        -1.81945    0.39287  -4.631  4.4e-06 ***
## z.t1         1.57478    0.13106  12.015  < 2e-16 ***
## z.t2        -0.15708    0.01019 -15.409  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.175 on 644 degrees of freedom
## Multiple R-squared:  0.3119, Adjusted R-squared:  0.3087
## F-statistic: 97.31 on 3 and 644 DF,  p-value: < 2.2e-16
```

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

In the summary for `model2.1`, the lag weights of the predictors are significant at the 5% statistical level as p-value < 0.05. Furthermore, the adjusted R-squared shows a value of 0.3119, which accounts for 30.77% of the variability in the data.

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

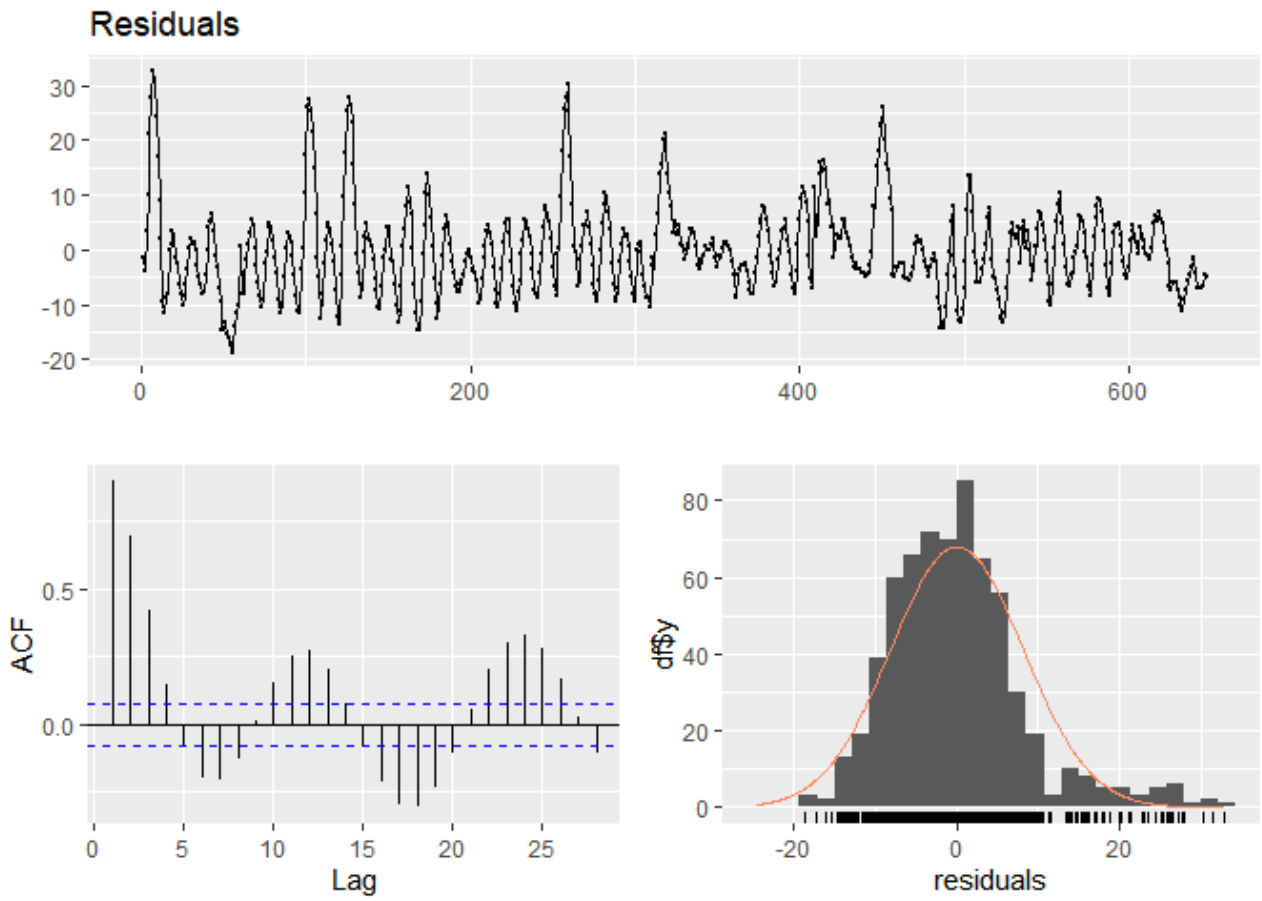

Figure 13

```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 1054.6, df = 10, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 10
```

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

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model2.1$model
## LM test = 525.11, df = 1, p-value < 2.2e-16
```

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

```
##
##  Shapiro-Wilk normality test
##
## data:  model2.1$model$residuals
## W = 0.94141, p-value = 2.679e-15
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 13`. The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are highly significant. Furthermore, from the Shapiro-Wilk test we can conclude that there is enough evidence to reject null hypothesis of normality as p-value < 0.05.

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

```
##      z.t0      z.t1      z.t2
##  5.115134 28.849300 17.496603
```

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

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

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

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

1.5.3 Koyck Distributed Lag Model

In the Koyck DLM similar to Polynomial DLM, we have used the independent predictor precipitation. The Koyck DLM transformation is useful when dealing with infinite DLM.

```
model3.1<-koyckDlm(x=as.vector(data1$ppt),y=as.vector(data1$solar))
summary(model3.1, diagnostics=TRUE)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.0926  -3.5961   0.3176   3.6103  14.8399
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.23925    0.76549  -2.925  0.00356 **
## Y.1          0.98546    0.02424  40.650  < 2e-16 ***
```

```
## X.t          5.34684    0.84383   6.336 4.37e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.814 on 656 degrees of freedom
## Multiple R-Squared: 0.7598,  Adjusted R-squared: 0.7591
## Wald test:  1104 on 2 and 656 DF,  p-value: < 2.2e-16
##
## Diagnostic tests:
##                  df1 df2 statistic       p-value
## Weak instruments   1 656  710.7209 1.191744e-106
## Wu-Hausman         1 655  146.8017  1.248856e-30
##
##                              alpha     beta       phi
## Geometric coefficients:  -154.0203 5.346844 0.9854613
```

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

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

Besides that, we can also observe that in the Wu-Hausman test, there is a significant correlation between the explanatory variable and the error term at 5% level as p-value < 0.05. Furthermore, the adjusted R-squared shows a value of 0.7598, which accounts for 75.98% of the variability in the data.

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

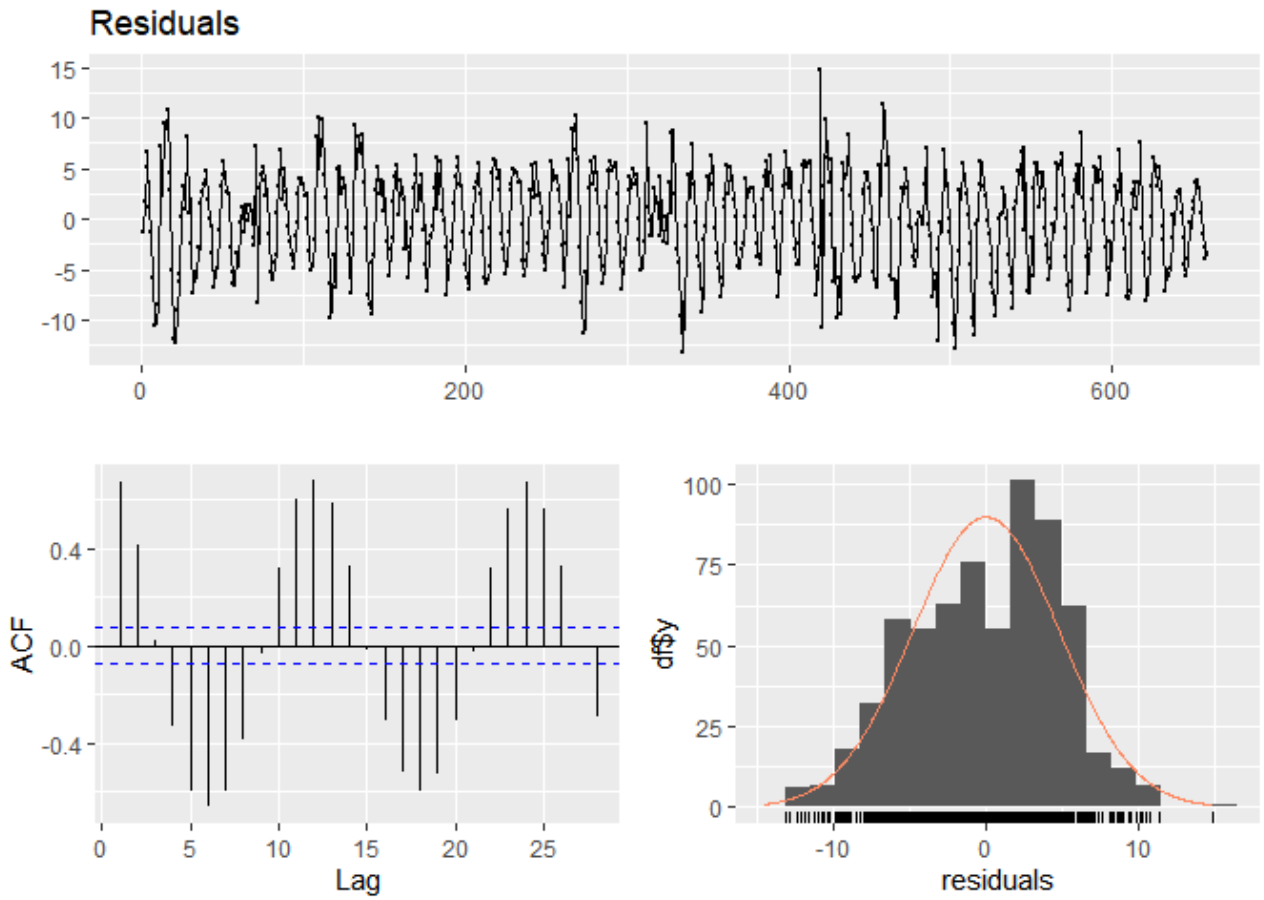

Figure 14

```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 1413.2, df = 10, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 10
```

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

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model3.1$model
## LM test = 387.66, df = 1, p-value < 2.2e-16
```

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

```
##
##  Shapiro-Wilk normality test
##
## data:  model3.1$model$residuals
## W = 0.98487, p-value = 2.468e-06
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 14`. The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are highly significant. Furthermore, from the Shapiro-Wilk test we can conclude that there is enough evidence to reject null hypothesis of normality as p-value < 0.05.

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

```
##      Y.1      X.t
## 1.605001 1.605001
```

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

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

```
## FALSE FALSE
```

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

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

1.5.4 Autoregressive Distributed Lag Model

Autoregressive DLM is normally used when a suitable solution with other DLMs cannot be found for infinite DLM. Similarly to Finite DLM, we will be using the predictor precipitation as the independent variable, and the solar radiation will be the dependent variable. To perform this test, we will first create a loop function for p and q and use the `ardlDLM` function to identify most appropriate number of lags for the DLM.

```
for (i in 1:6){
  for(j in 1:6){
    model4.1 = ardlDlm(x = as.vector(data1$ppt), y = as.vector(data1$solar), p = i , q = j)
    cat("p =", i, "q =", j, "AIC =", AIC(model4.1$model), "BIC =", BIC(model4.1$model), "MASE =",
        MASE(model4.1)$MASE, "\n")
  }
}
```

```
## p = 1 q = 1 AIC = 3712.311 BIC = 3734.765 MASE = 0.8392434
## p = 1 q = 2 AIC = 3239.416 BIC = 3266.352 MASE = 0.4971918
## p = 1 q = 3 AIC = 3143.522 BIC = 3174.936 MASE = 0.4740063
## p = 1 q = 4 AIC = 3138.399 BIC = 3174.288 MASE = 0.4697571
## p = 1 q = 5 AIC = 3100.283 BIC = 3140.644 MASE = 0.450425
## p = 1 q = 6 AIC = 3033.2 BIC = 3078.031 MASE = 0.4239899
## p = 2 q = 1 AIC = 3639.223 BIC = 3666.159 MASE = 0.7834855
## p = 2 q = 2 AIC = 3229.051 BIC = 3260.476 MASE = 0.4951319
## p = 2 q = 3 AIC = 3137.634 BIC = 3173.535 MASE = 0.4738939
## p = 2 q = 4 AIC = 3132.962 BIC = 3173.337 MASE = 0.4702773
## p = 2 q = 5 AIC = 3097.288 BIC = 3142.134 MASE = 0.4503599
## p = 2 q = 6 AIC = 3031.898 BIC = 3081.212 MASE = 0.4235203
## p = 3 q = 1 AIC = 3608.793 BIC = 3640.207 MASE = 0.7572489
## p = 3 q = 2 AIC = 3226.623 BIC = 3262.524 MASE = 0.4955334
## p = 3 q = 3 AIC = 3139.409 BIC = 3179.798 MASE = 0.4737144
## p = 3 q = 4 AIC = 3134.777 BIC = 3179.638 MASE = 0.4701162
## p = 3 q = 5 AIC = 3098.808 BIC = 3148.139 MASE = 0.4502885
## p = 3 q = 6 AIC = 3032.418 BIC = 3086.216 MASE = 0.4227238
## p = 4 q = 1 AIC = 3602.664 BIC = 3638.553 MASE = 0.7580664
## p = 4 q = 2 AIC = 3224.285 BIC = 3264.66 MASE = 0.4959949
## p = 4 q = 3 AIC = 3131.289 BIC = 3176.15 MASE = 0.4695096
## p = 4 q = 4 AIC = 3131.424 BIC = 3180.772 MASE = 0.4665123
## p = 4 q = 5 AIC = 3096.024 BIC = 3149.839 MASE = 0.4479481
## p = 4 q = 6 AIC = 3027.07 BIC = 3085.35 MASE = 0.420076
## p = 5 q = 1 AIC = 3599.402 BIC = 3639.764 MASE = 0.7572617
## p = 5 q = 2 AIC = 3221.853 BIC = 3266.699 MASE = 0.4954501
```

```
## p = 5 q = 3 AIC = 3127.103 BIC = 3176.434 MASE = 0.4675479
## p = 5 q = 4 AIC = 3127.868 BIC = 3181.684 MASE = 0.4651969
## p = 5 q = 5 AIC = 3097.877 BIC = 3156.177 MASE = 0.4479311
## p = 5 q = 6 AIC = 3029.06 BIC = 3091.824 MASE = 0.4201638
## p = 6 q = 1 AIC = 3586.116 BIC = 3630.948 MASE = 0.7503192
## p = 6 q = 2 AIC = 3214.331 BIC = 3263.645 MASE = 0.4913516
## p = 6 q = 3 AIC = 3119.154 BIC = 3172.952 MASE = 0.4653876
## p = 6 q = 4 AIC = 3120.48 BIC = 3178.76 MASE = 0.4633103
## p = 6 q = 5 AIC = 3094.149 BIC = 3156.912 MASE = 0.4473944
## p = 6 q = 6 AIC = 3030.976 BIC = 3098.223 MASE = 0.4205664
```

From the loop function output above, we can see that p = 4, q = 6 contains the lowest AIC value and MASE value, and p = 1, q= 6 contains the lowest BIC value. Hence, we will be using these 2 values for the Autoregressive DLM.

```
model4.2<-ardlDlm(x=as.vector(data1$ppt),y=as.vector(data1$solar),p=1,q=6)
summary(model4.2)
```

```
##
## Time series regression with "ts" data:
## Start = 7, End = 660
##
## Call:
## dynlm(formula = as.formula(model.text), data = data, start = 1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.9649  -1.3262  -0.1447   0.9767  18.9414
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.48394    0.39245   3.781 0.000171 ***
## X.t         -0.90002    0.48413  -1.859 0.063474 .
## X.1          2.01524    0.48482   4.157 3.66e-05 ***
## Y.1          1.21022    0.03748  32.287  < 2e-16 ***
## Y.2          0.03513    0.06025   0.583 0.560063
## Y.3         -0.28327    0.05958  -4.754 2.46e-06 ***
## Y.4         -0.21043    0.05943  -3.541 0.000428 ***
## Y.5         -0.16628    0.06000  -2.771 0.005748 **
## Y.6          0.30180    0.03684   8.192 1.38e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.439 on 645 degrees of freedom
## Multiple R-squared:  0.9391, Adjusted R-squared:  0.9383
## F-statistic:  1243 on 8 and 645 DF,  p-value: < 2.2e-16
```

From the above output for `model4.2`, we can see that the lag weights are significant at the 5% statistical level as p-value < 0.05. Furthermore, the adjusted R-squared shows a value of 0.9391, which accounts for 93.91% of the variability in the data.

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

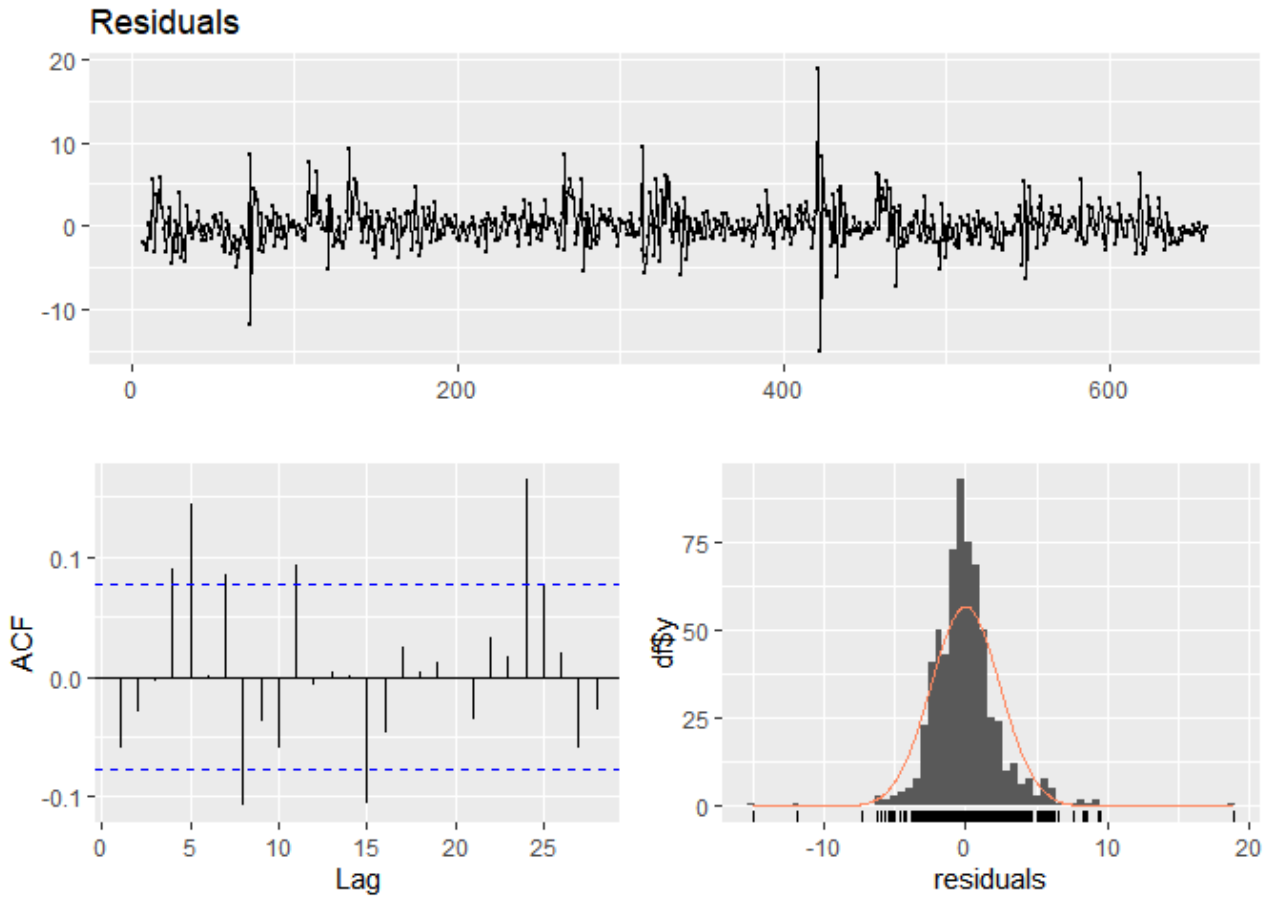

Figure 15

```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 37.751, df = 10, p-value = 4.192e-05
##
## Model df: 0.   Total lags used: 10
```

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

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model4.2$model
## LM test = 19.522, df = 1, p-value = 9.943e-06
```

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

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

```
## data:  model4.2$model$residuals
## W = 0.90301, p-value < 2.2e-16
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 15`. The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that there are no serial correlation in residuals.. Furthermore, from the Shapiro-Wilk test we can conclude that there is enough evidence to reject null hypothesis of normality as p-value < 0.05.

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

```
##       X.t L(X.t, 1) L(y.t, 1) L(y.t, 2) L(y.t, 3) L(y.t, 4) L(y.t, 5) L(y.t, 6)
##  3.220932  3.228270 14.857039 38.336402 37.451794 37.277203 38.080166 14.391011
```

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

```
##       X.t L(X.t, 1) L(y.t, 1) L(y.t, 2) L(y.t, 3) L(y.t, 4) L(y.t, 5) L(y.t, 6)
##     FALSE     FALSE      TRUE      TRUE      TRUE      TRUE      TRUE      TRUE
```

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the `vif` function, we can observe that there the data is suffering from multicollinearity. The reason for this is because the data in the VIF are more than 10.

```
model4.3<-ardlDlm(x=as.vector(data1$ppt),y=as.vector(data1$solar),p=4,q=6)
summary(model4.3)
```

```
##
## Time series regression with "ts" data:
## Start = 7, End = 660
##
## Call:
## dynlm(formula = as.formula(model.text), data = data, start = 1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.3529  -1.2115  -0.1471   1.0354  18.6627
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.73398    0.42506   4.079 5.08e-05 ***
## X.t         -0.61517    0.52239  -1.178 0.239394
## X.1          0.73634    0.73867   0.997 0.319216
## X.2          1.20214    0.75229   1.598 0.110539
## X.3          0.75474    0.74256   1.016 0.309822
```

```
## X.4         -1.42568    0.52932  -2.693 0.007258 **
## Y.1          1.20310    0.03758  32.011  < 2e-16 ***
## Y.2          0.04996    0.06003   0.832 0.405589
## Y.3         -0.27679    0.05931  -4.667 3.73e-06 ***
## Y.4         -0.21731    0.05922  -3.670 0.000263 ***
## Y.5         -0.17880    0.05974  -2.993 0.002869 **
## Y.6          0.30516    0.03679   8.295 6.39e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.422 on 642 degrees of freedom
## Multiple R-squared:  0.9402, Adjusted R-squared:  0.9392
## F-statistic: 917.8 on 11 and 642 DF,  p-value: < 2.2e-16
```

From the above output for `model4.3`, we can see that the lag weights are significant at the 5% statistical level as p-value < 0.05. Furthermore, the adjusted R-squared shows a value of 0.9392, which accounts for 93.92% of the variability in the data.

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

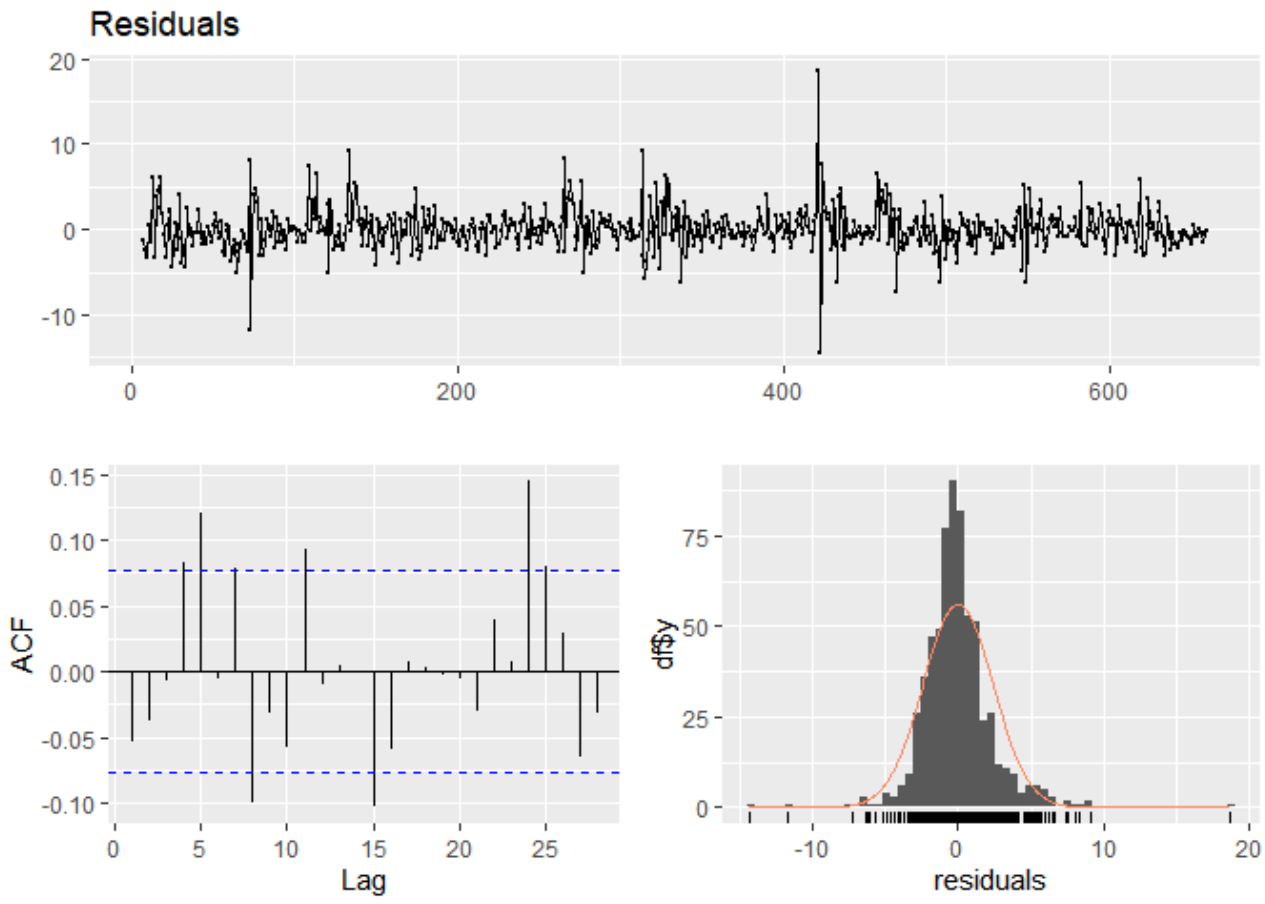

Figure 16

```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 30.681, df = 10, p-value = 0.0006622
```

```
##
## Model df: 0.   Total lags used: 10
```

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

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model4.3$model
## LM test = 17.984, df = 1, p-value = 2.228e-05
```

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

```
##
##  Shapiro-Wilk normality test
##
## data:  model4.3$model$residuals
## W = 0.90665, p-value < 2.2e-16
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 16`. The Breush-Godfrey test is also conducted here using the `bgtest` function, and the Shapiro-Wilk normality test is conducted using the `shapiro.test` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that there are no serial correlation in residuals.. Furthermore, from the Shapiro-Wilk test we can conclude that there is enough evidence to reject null hypothesis of normality as p-value < 0.05.

```
VIF.model4.3<-vif(model4.2$model)
VIF.model4.3
```

```
##       X.t L(X.t, 1) L(y.t, 1) L(y.t, 2) L(y.t, 3) L(y.t, 4) L(y.t, 5) L(y.t, 6)
##  3.220932  3.228270 14.857039 38.336402 37.451794 37.277203 38.080166 14.391011
```

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

```
##       X.t L(X.t, 1) L(y.t, 1) L(y.t, 2) L(y.t, 3) L(y.t, 4) L(y.t, 5) L(y.t, 6)
##     FALSE     FALSE      TRUE      TRUE      TRUE      TRUE      TRUE      TRUE
```

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the `vif` function, we can observe that there the data is suffering from multicollinearity. The reason for this is because the data in the VIF are more than 10.

Again, similarly to the previous models, this model is not successful at capturing the autocorrelation and seasonality in the series. Thus, all models in the time series regression models are not suitable for further

analysis.

1.6.0 Dynamic Linear Models

Since there are no intervention points as analyzed in the earlier sections for `solar` and `ppt`, we will not fit the dynamic linear models as the models are for assessing the effect of the intervention on the time series.

1.7.0 Exponential Smoothing Methods

Now we will use the exponential smoothing method as we have found seasonality in the solar radiation series. To start the test, we will first create a loop to store the measures for exponential, seasonality, and damped.

```
expo = c(T,F)
seas<-c("additive","multiplicative")
damped<-c(T,F)
exp<-expand.grid(expo, seas, damped)
exp<-exp[-c(1,5),]
f.aic<-array(NA, 4)
f.mase<-array(NA, 4)
levels<-array(NA, dim=c(4,3))

for (i in 1:4){
  hw<-hw(solar.ts, expo = exp[i,1], seas = toString(exp[i,2], damped = exp[i,3]))
  f.aic[i]<-hw$model$aic
  f.mase[i]<-accuracy(hw)[6]
  levels[i,1]<-exp[i,1]
  levels[i,2]<-toString(exp[i,2])
  levels[i,3]<-exp[i,3]
}

model5<-data.frame(levels, f.mase, f.aic)
colnames(model5)= c("exponential","seasonality","damped","MASE", "AIC")
model5<-arrange(model5,MASE)
model5
```

```
##   exponential    seasonality damped      MASE      AIC
## 1       FALSE multiplicative   TRUE 0.2233077 6648.746
## 2        TRUE multiplicative   TRUE 0.2320404 6584.208
## 3       FALSE       additive   TRUE 0.2471600 5434.708
## 4       FALSE       additive  FALSE 0.2471600 5434.708
```

From the output above, here is what we observed:

- The model with exponential=FALSE, seasonal=multiplicative, and damped=TRUE has the lowest MASE of 0.2233077 and has a AIC value of 6648.746.

- The model with exponential=TRUE, seasonal=multiplicative, and damped=TRUE has the second lowest MASE of 0.2320404 and has a AIC value of 6584.208.
- The model with exponential=FALSE, seasonal=additive, and damped=TRUE has the lowest AIC of 5434.708 and has a MASE of 0.2471600.
- The model with exponential=FALSE, seasonal=additive, and damped=FALSE has the lowest AIC of 5434.708 and has a MASE of 0.2471600.

Thus, we will now compare the lowest MASE and AIC models with their residual plots.

```
model5.1<-
         hw(solar.ts,exponential=FALSE,seasonal="multiplicative",damped=TRUE,h=2*frequency(solar.ts))
checkresiduals(model5.1)
```

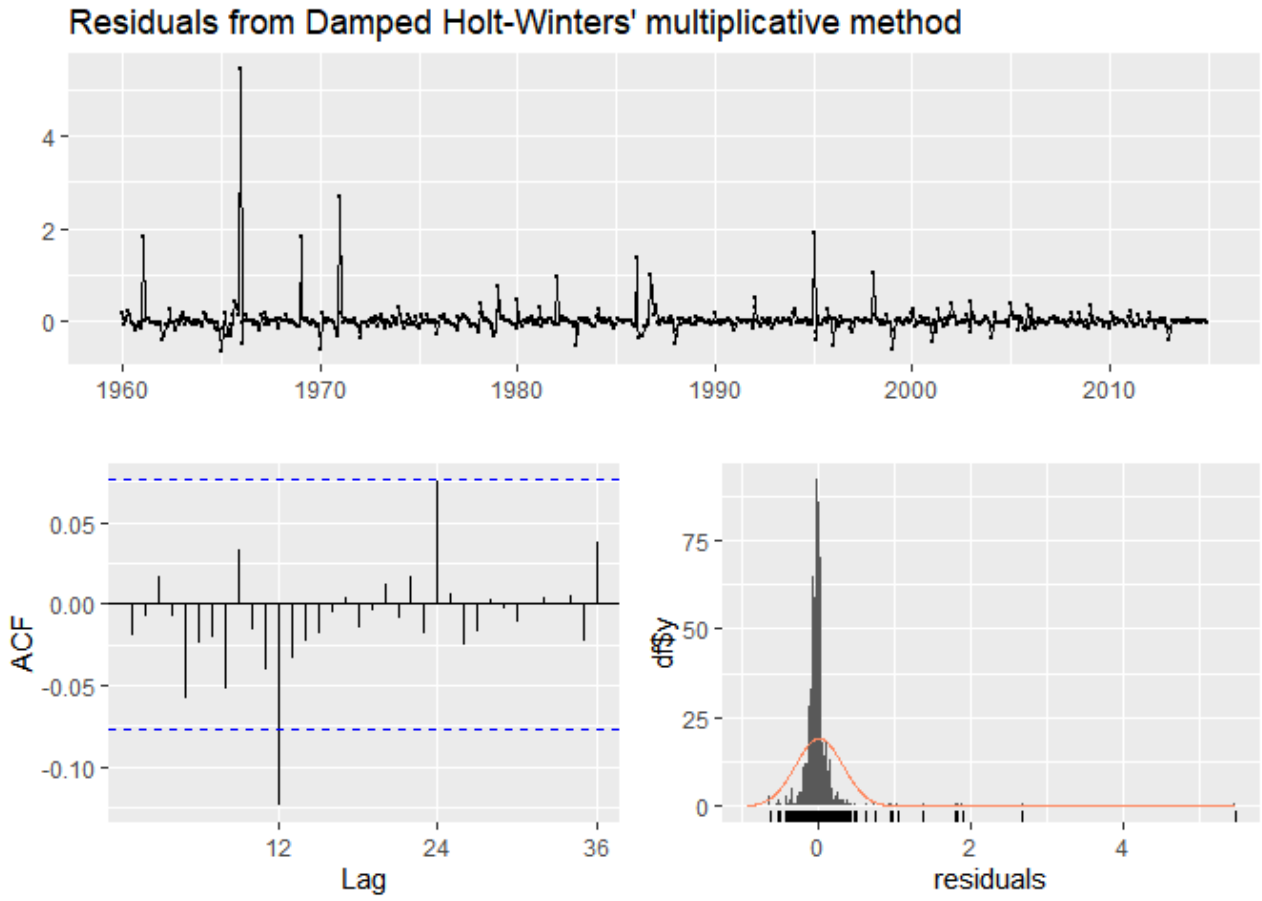

Figure 17

```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt-Winters' multiplicative method
## Q* = 23.57, df = 24, p-value = 0.4864
##
## Model df: 0.   Total lags used: 24
```

`Figure 17` above shows the residual plot for `model5.1`. Here is what we can observe from the output:

- Based on the p-values, we cannot reject the hypothesis of stationary series as the p-value is greater that 0.05.
- There are significant lags observed in lag 12 at the ACF plot.

```
model5.2<-
        hw(solar.ts,exponential=TRUE,seasonal="multiplicative",damped=TRUE,h=2*frequency(solar.ts))
checkresiduals(model5.2)
```

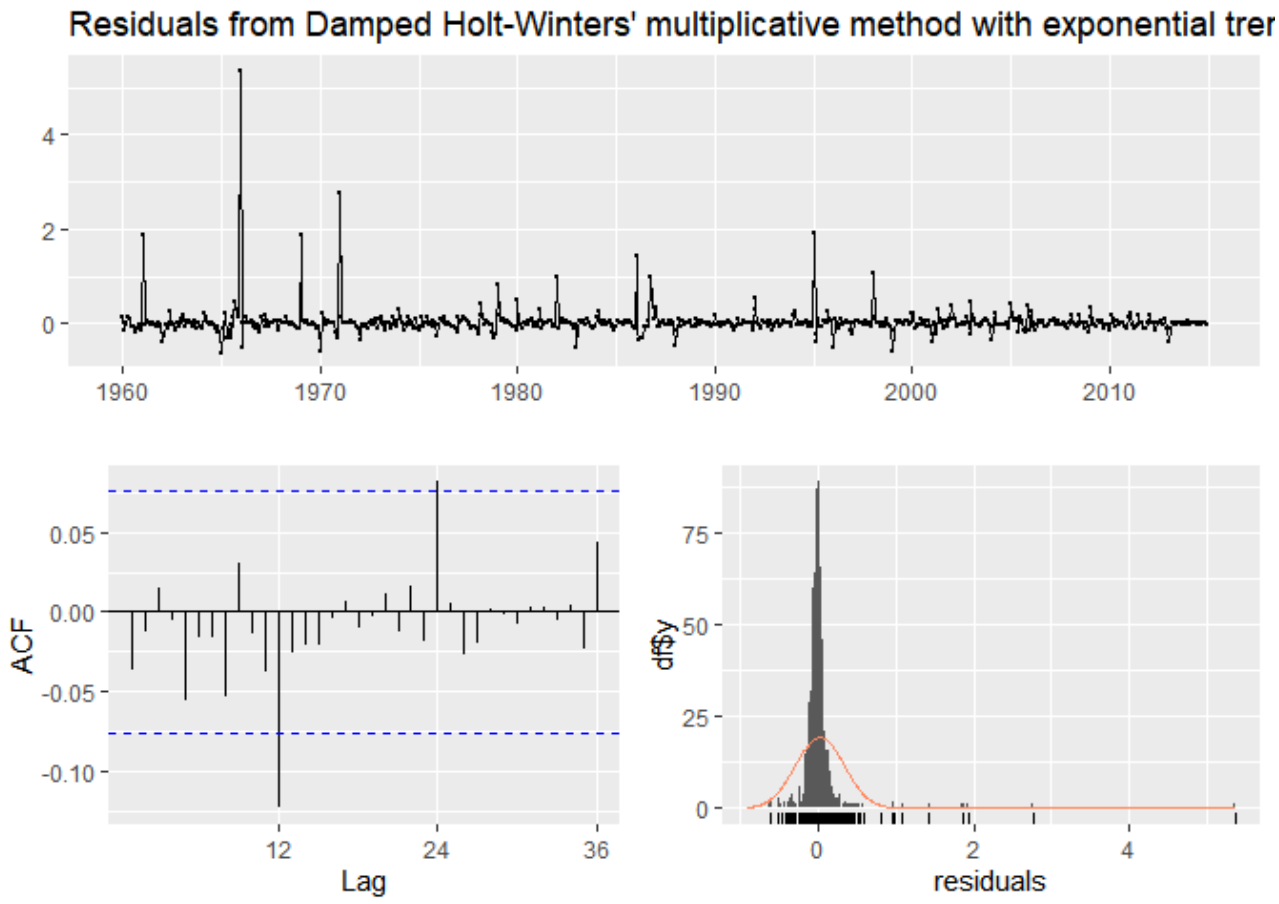

Figure 18

```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt-Winters' multiplicative method with exponential trend
## Q* = 23.862, df = 24, p-value = 0.4695
##
## Model df: 0.   Total lags used: 24
```

`Figure 18` above shows the residual plot for `model5.2` . Here is what we can observe from the output:

- Based on the p-values, we cannot reject the hypothesis of stationary series as the p-value is greater that 0.05.
- There are significant lags observed in lag 12 and 24 at the ACF plot.

```
model5.3<-hw(solar.ts,exponential=FALSE,seasonal="additive",damped=TRUE,h=2*frequency(solar.ts))
```

```
checkresiduals(model5.3)
```

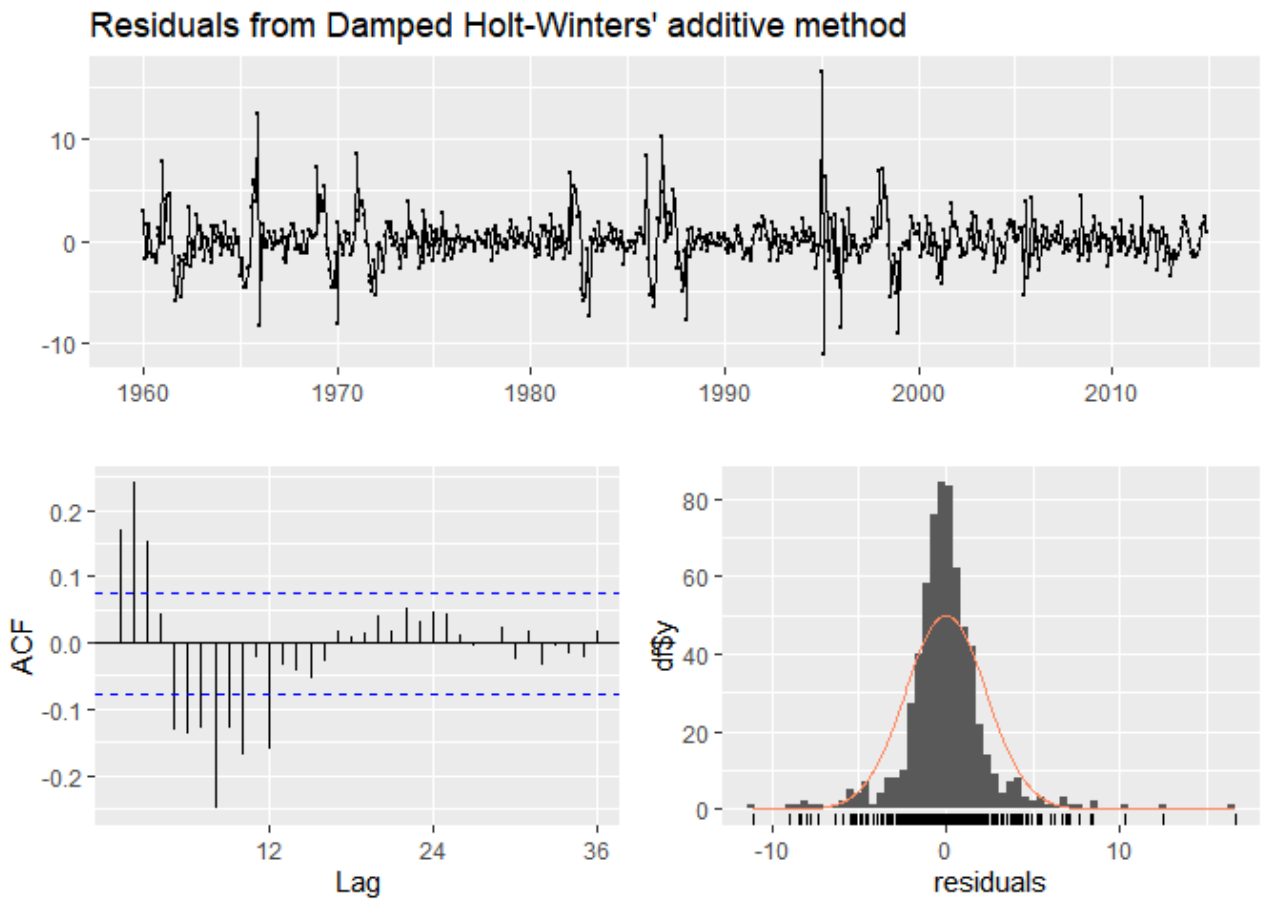

Figure 19

```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt-Winters' additive method
## Q* = 210.76, df = 24, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 24
```

`Figure 19` above shows the residual plot for `model5.3` . Here is what we can observe from the output:

- Based on the p-values, we reject the hypothesis of stationary series as the p-value is less than that 0.05.
- There are many significant lags observed in the ACF plot.

```
model5.4<-hw(solar.ts,exponential=FALSE,seasonal="additive",damped=FALSE,h=2*frequency(solar.ts))
checkresiduals(model5.4)
```

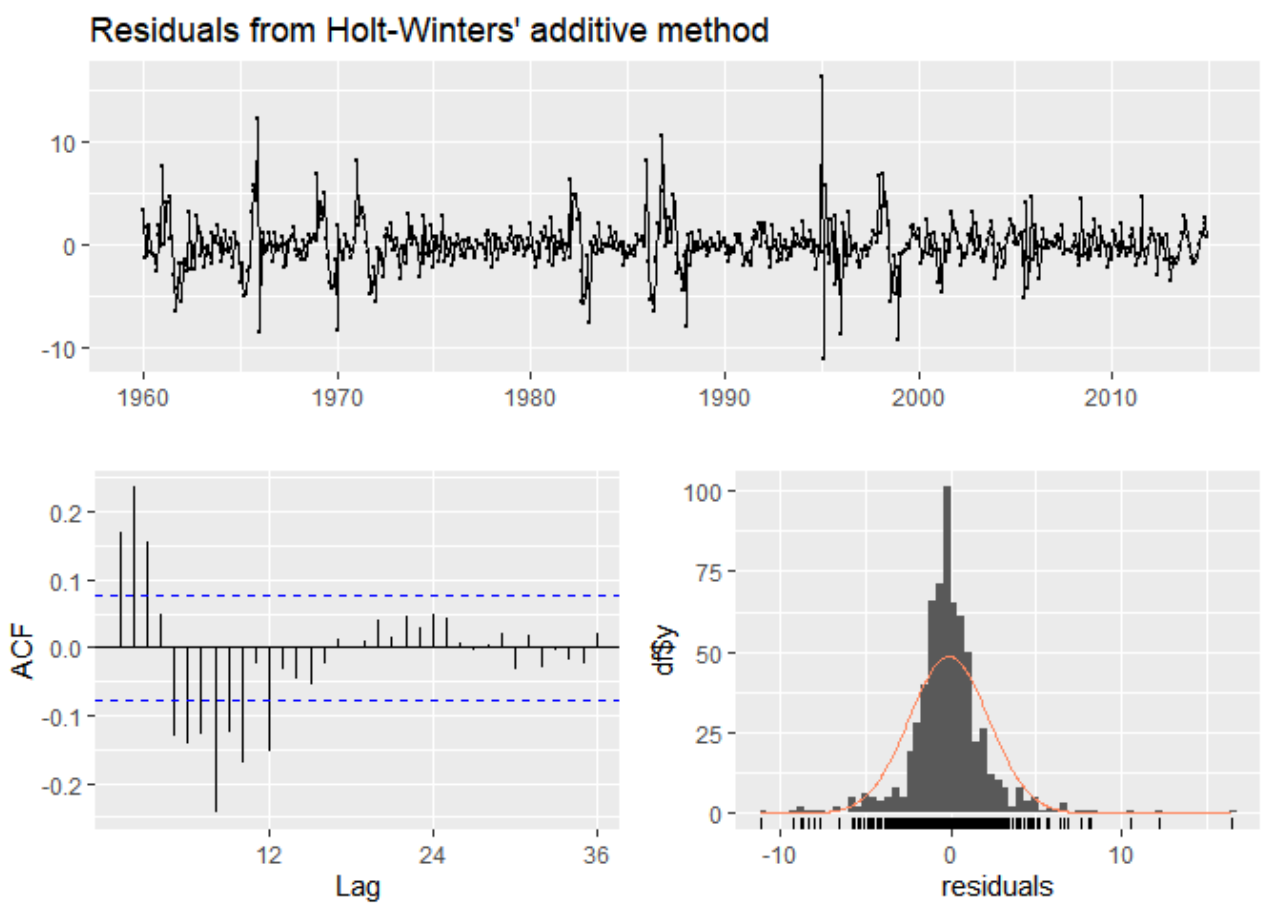

Figure 20

```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' additive method
## Q* = 205.55, df = 24, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 24
```

`Figure 20` above shows the residual plot for `model5.4`. Here is what we can observe from the output:

- Based on the p-values, we reject the hypothesis of stationary series as the p-value is less than that 0.05.
- There are many significant lags observed in the ACF plot.

Comparing between `Figure 17` to `Figure 20`, we can make a decision to select `model5.1` for exponential smoothing modeling as it has the lowest MASE value among all, and has the least significant lags.

1.8.0 State-Space Models

From previous analysis, we concluded that seasonality is leading cause of variations in the time series. Therefore, state-space models with seasonality components should be used to fit the series. We will first create a loop to fit the state-space models.

```
ssmodels<-c("AAA","MAM","MMM")
ssdamped<-c(TRUE,FALSE)
ssexp<-expand.grid(ssmodels,ssdamped)
ss.f.aic<-array(NA,6)
ss.f.mase<-array(NA,6)
sslevels<-array(NA,dim=c(6,2))

for(i in 1:6){
  ets <- ets(solar.ts, model = toString(ssexp[i, 1]), damped = ssexp[i,2])
  ss.f.aic[i] <- ets$aic
  ss.f.mase[i] <- accuracy(ets)[6]
  sslevels[i,1] <- toString(ssexp[i,1])
  sslevels[i,2] <- ssexp[i,2]
}

model6<-data.frame(sslevels,ss.f.mase,ss.f.aic)
colnames(model6)<-c("Models","Damped","MASE","AIC")
model6<-arrange(model6,MASE)
model6
```

```
##   Models Damped      MASE      AIC
## 1    AAA   TRUE 0.2461797 5428.422
## 2    AAA  FALSE 0.2471600 5434.708
## 3    MMM   TRUE 0.3201193 5995.550
## 4    MAM   TRUE 0.3222574 5953.502
## 5    MAM  FALSE 0.3721664 6105.959
## 6    MMM  FALSE 0.5292151 6670.168
```

From the output above, here is what we observed:

- The model AAA, damped=TRUE has the lowest MASE value and lowest AIC value.
- The model AAA, damped=FALSE has the second lowest MASE value and the second lowest AIC value.

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

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

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

From the auto ETS model output above we are recommended to use the model AAA with damped=TRUE, which is consistent with our output in `model6`. Now, we will now compare the AAA models with damped= `TRUE` and `FALSE` with their residual plots.

```
model6.1<-ets(solar.ts,model="AAA",damped=TRUE)
checkresiduals(model6.1)
```

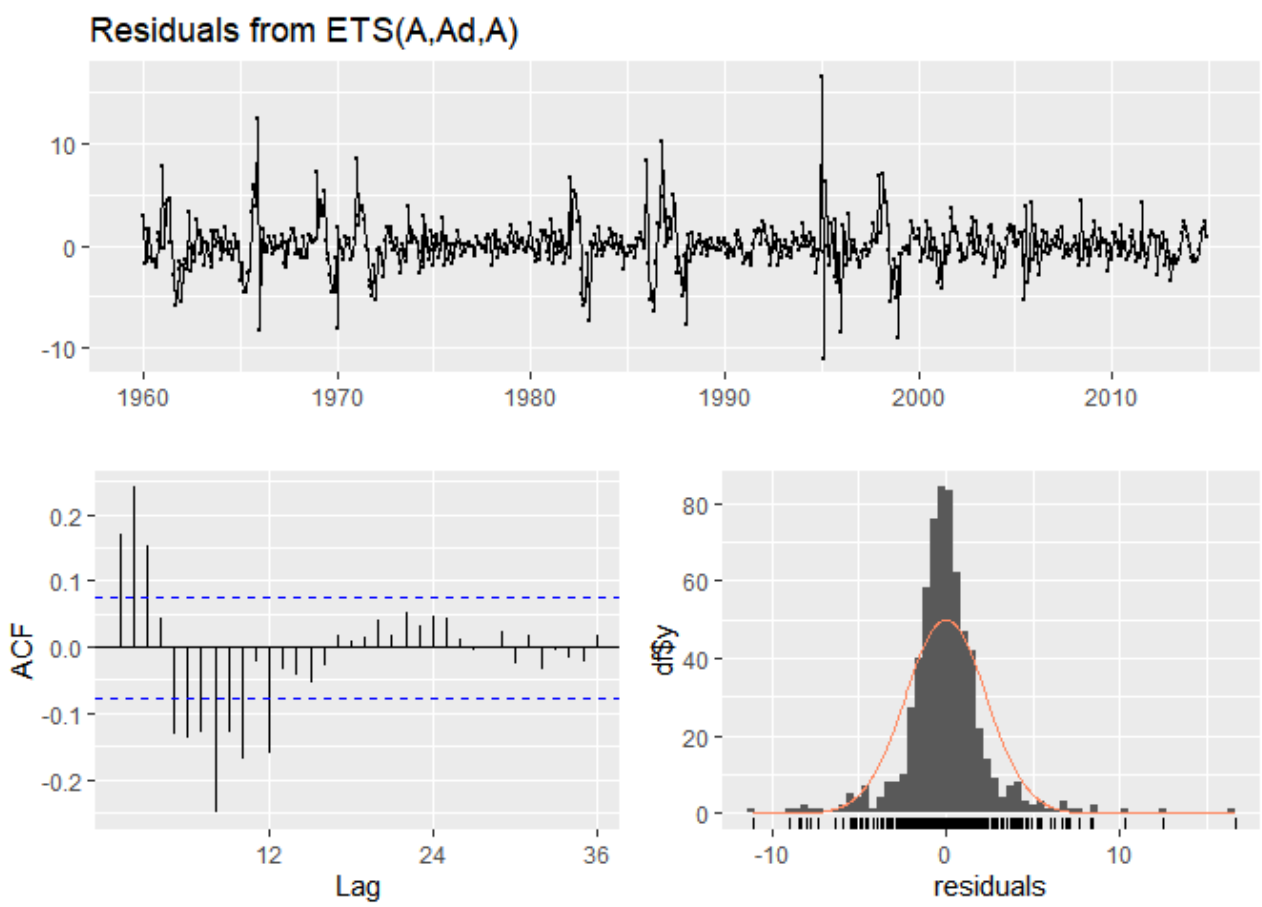

Figure 21

```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,Ad,A)
## Q* = 210.76, df = 24, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 24
```

`Figure 21` above shows the residual plot for `model6.1`. Here is what we can observe from the output:

- Based on the p-values, we reject the hypothesis of stationary series as the p-value is less than 0.05.
- There are many significant lags observed in the ACF plot.

```
model6.2<-ets(solar.ts,model="AAA",damped=FALSE)
checkresiduals(model6.2)
```

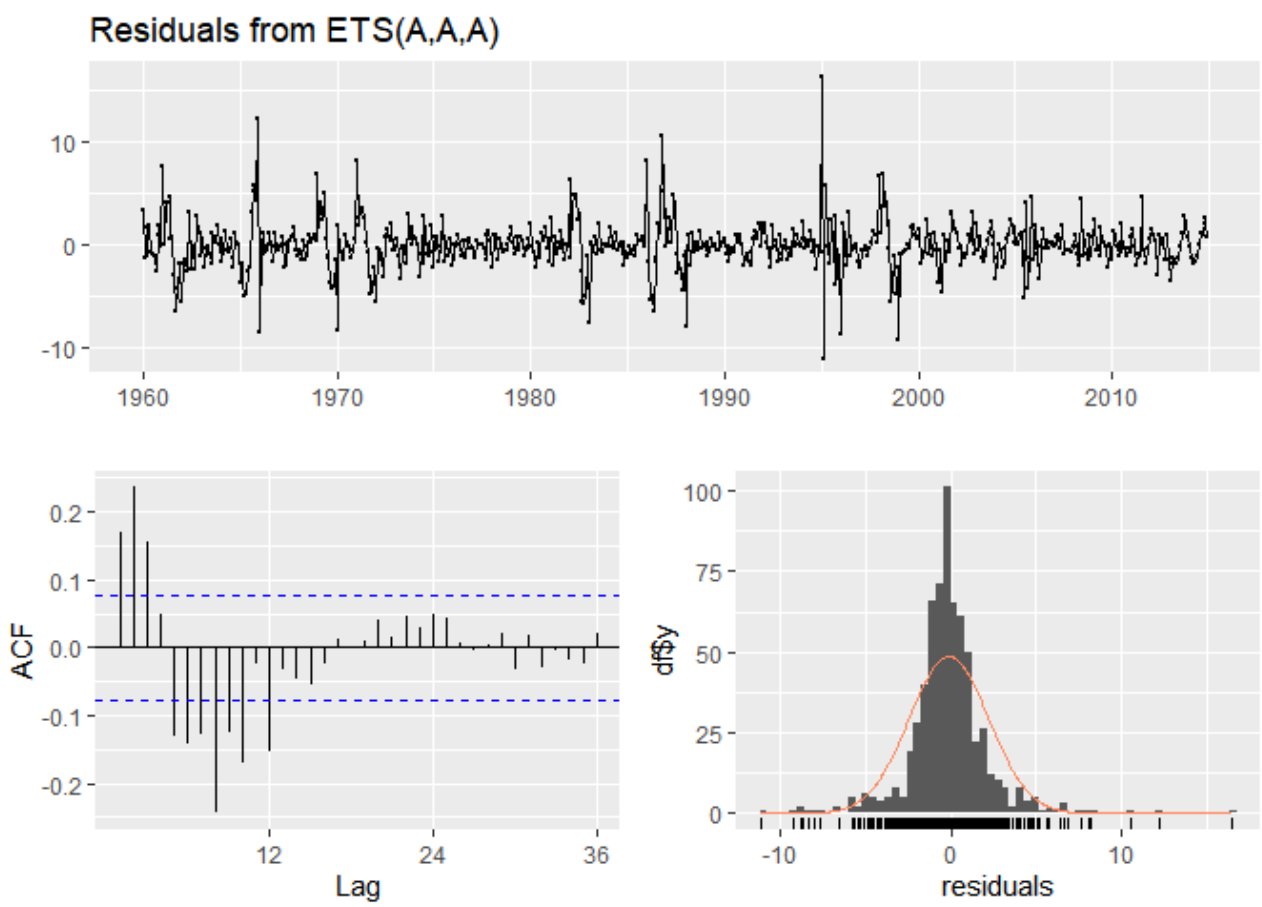

Figure 22

```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,A,A)
## Q* = 205.55, df = 24, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 24
```

`Figure 22` above shows the residual plot for `model6.2`. Here is what we can observe from the output:

- Based on the p-values, we reject the hypothesis of stationary series as the p-value is less than 0.05.
- There are many significant lags observed in the ACF plot.

1.9.0 Model Comparison and Selection

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

```
model.tsrm<-c("Finite DLM","Polynomial DLM","Koyck DLM","Autoregressive DLM (P1Q6)",
              "Autoregressive DLM (P4Q6)")

mase.tsrm<-MASE(model1.2,model2.1,model3.1,model4.2,model4.3)

aic.tsrm<-c(AIC(model1.2),AIC(model2.1),AIC(model3.1),AIC(model4.2),AIC(model4.3))
```

```
## [1] 4578.787
## [1] 4567.969
## [1] 3946.476
## [1] 3033.2
## [1] 3027.07
```

```
bic.tsrm<-c(BIC(model1.2),BIC(model2.1),BIC(model3.1),BIC(model4.2),BIC(model4.3))
```

```
## [1] 4645.895
## [1] 4590.339
## [1] 3964.439
## [1] 3078.031
## [1] 3085.35
```

```
tsrm<-data.frame(model.tsrm,mase.tsrm,aic.tsrm,bic.tsrm)
tsrm<-tsrm[-2]
colnames(tsrm)<-c("Model","MASE","AIC","BIC")

model.ss<-c("HW Multiplicative","HW Multiplicative Exponential","HW Additive Damped",
            "HW Additive","ETS (A,Ad,A)","ETS(A,A,A)")

mase.ss<-c(accuracy(model5.1)[,6],accuracy(model5.2)[,6],accuracy(model5.3)[,6],
           accuracy(model5.4)[,6],accuracy(model6.1)[,6],accuracy(model6.2)[,6])

aic.ss<-c(model5.1$model$aic,model5.2$model$aic,model5.3$model$aic,model5.4$model$aic,
          model6.1$aic,model6.2$aic)

bic.ss<-c(model5.1$model$bic,model5.2$model$bic,model5.3$model$bic,model5.4$model$bic,
          model6.1$bic,model6.2$bic)

ss<-data.frame(model.ss,mase.ss,aic.ss,bic.ss)
colnames(ss)<-c("Model","MASE","AIC","BIC")

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

```
##                           Model      MASE      AIC      BIC
## 1  HW Multiplicative Exponential 0.2032009 6365.329 6446.190
## 2               HW Multiplicative 0.2035619 6366.701 6447.561
## 3             HW Additive Damped 0.2461797 5428.422 5509.282
## 4                   ETS (A,Ad,A) 0.2461797 5428.422 5509.282
## 5                    HW Additive 0.2471600 5434.708 5511.076
## 6                     ETS(A,A,A) 0.2471600 5434.708 5511.076
## 7      Autoregressive DLM (P4Q6) 0.4200760 3027.070 3085.350
## 8      Autoregressive DLM (P1Q6) 0.4239899 3033.200 3078.031
## 9                      Koyck DLM 1.0324829 3946.476 3964.439
```

```
## 10                   Finite DLM 1.5516004 4578.787 4645.895
## 11               Polynomial DLM 1.5630351 4567.969 4590.339
```

Based on the `compare` output, we can observe that HW Multiplicative Exponential, and HW Multiplicative have the lowest MASE values. Furthermore, with respective of the diagnostics checking from the previous sections, we shall use the first 4 models from the output for further analysis.

1.10.0 Forecasting

In this section, we will give the best 2 years ahead forecast by using the models Holt-Winters Multiplicative seasonality case with exponential, Holt-Winters Multiplicative seasonality case,Holt-Winters Additive Damped, and ETS(A,Ad,A), and we will use `datax` to compare the data for January 2015 to December 2016.

```
fit1<-hw(solar.ts,exponential=TRUE,seasonal="multiplicative",damped=TRUE,h=2*frequency(solar.ts))
fit2<-hw(solar.ts,exponential=FALSE,seasonal="multiplicative",damped=TRUE,h=2*frequency(solar.ts))
fit3<-hw(solar.ts,seasonal="additive",damped=TRUE,h=2*frequency(solar.ts))
fit4<-ets(solar.ts,model="AAA",damped=TRUE)
plot(forecast.ets(fit4),main="2 Years Forecast for Solar Radiation",ylab="Solar
         Radiation",xlab="Year")
lines(fitted(fit1),col="red",lty=1)
lines(fitted(fit2),col="green",lty=1)
lines(fitted(fit3),col="cyan",lty=1)
lines(fitted(fit4),col="brown",lty=1)
lines(fit1$mean,col="red")
lines(fit2$mean,col="green")
lines(fit3$mean,col="cyan")
lines(fit4$mean,col="brown")
legend("bottomleft",cex=0.7,lty=1, pch=1, col=c("black","red","green","cyan","brown"),
       c("data","Holt Winters' Multiplicative Exponential", "Holt Winters' Multiplicative",
         "Holt Winters' Additive Damped", "ETS(A,Ad,A)"))
```

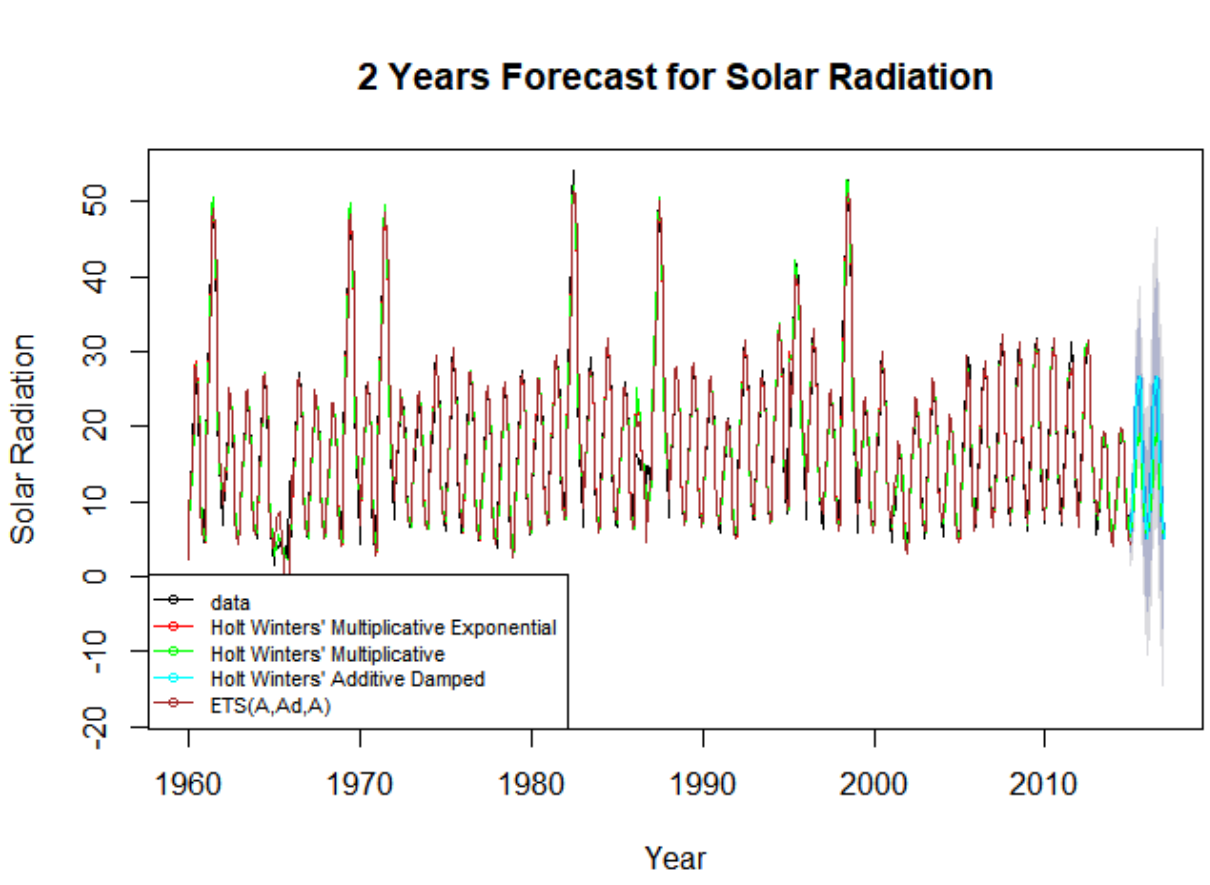

Figure 23

Based on previous analysis, we will also choose Holt Winters Multiplicative seasonality case with exponential for forecasting.

```
model.frc<-
         hw(solar.ts,exponential=TRUE,seasonal="multiplicative",damped=TRUE,h=2*frequency(solar.ts))
upper.95<-model.frc$upper[,2]
lower.95<-model.frc$lower[,2]
datax.frc<-datax.ts
centre<-model.frc$mean
plot(solar.ts, ylim=c(-20, 100))
lines(centre, col = "blue")
lines(upper.95, col = "green")
lines(lower.95, col = "green")
lines(datax.frc,col = "red")
legend("bottomleft", lty=1, cex=0.7, pch=1,
       col=c("black","blue","green","red"),
       c("Data","Forecasts","95% Lower and Upper Bounds","Precipitation Measurements"))
```

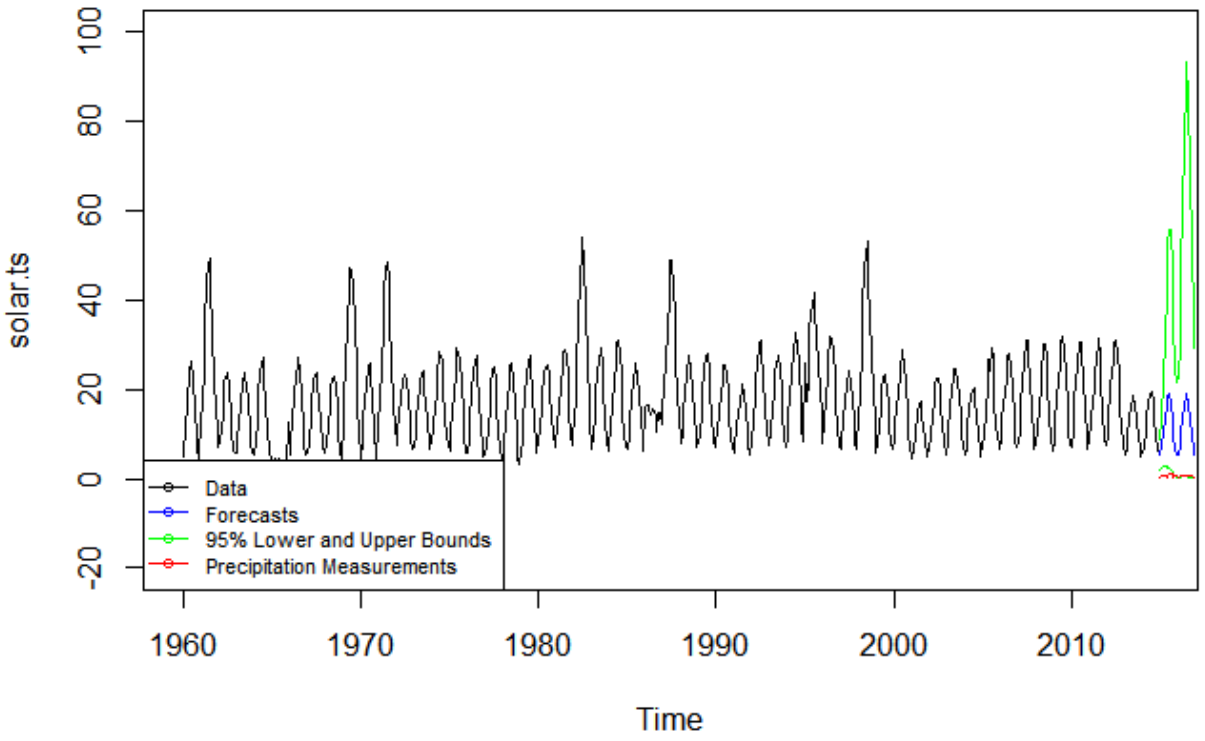

Figure 24

`Figure 24` shows the forecasts and 95% confidence interval for the solar radiation series. Since solar radiation is positive numbers, we will ignore the lower bound of the 95% confidence interval.

1.11.0 Conclusion

In conclusion, the exponential smoothing method Holt Winters Multiplicative seasonality case with exponential is the most suitable model for forecasting the solar radiation series.

2.0 Task 2

2.1.0 Introduction

The aim of this investigation is to analyse the correlation between Residential Property Price Index (PPI) in Melbourne and quarterly population change over the previous quarter in Victoria between September 2003 and December 2016., and to analyse whether the correlation between the two series is spurious or not. We will be using the dataset `data 2` which contains the information on the quarterly Residential Property Price Index (PPI) in Melbourne and quarterly population change over the previous quarter in Victoria between September 2003 and December 2016.

To perform these investigations, we will first prep the data for analysis and do further testing through various analysis methods.

2.2.0 Data Description and Preprocessing

The dataset `data 2` is used for analysis and contains the information on the quarterly Residential Property Price Index (PPI) in Melbourne and quarterly population change over the previous quarter in Victoria between September 2003 and December 2016.

```
data2<-read_csv("data2.csv")
```

```
## Rows: 54 Columns: 3
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (1): Quarter
## dbl (2): price, change
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
class(data2)
```

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

```
str(data2)
```

```
## spc_tbl_ [54 × 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Quarter: chr [1:54] "Sep-2003" "Dec-2003" "Mar-2004" "Jun-2004" ...
##  $ price  : num [1:54] 60.7 62.1 60.8 60.9 60.9 62.4 62.5 63.2 63.1 64 ...
##  $ change : num [1:54] 14017 12350 17894 9079 16210 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Quarter = col_character(),
##   ..   price = col_double(),
##   ..   change = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
head(data2)
```

```
## # A tibble: 6 × 3
##   Quarter  price change
##   <chr>    <dbl>  <dbl>
## 1 Sep-2003  60.7  14017
## 2 Dec-2003  62.1  12350
## 3 Mar-2004  60.8  17894
## 4 Jun-2004  60.9   9079
```

```
## 5 Sep-2004  60.9  16210
## 6 Dec-2004  62.4  13788
```

From the code chunk above, we will first import the first dataset into R using the `read_csv` function and assign the dataset with the neme `data2`. Then we will check the class, structure, and view the first few observations of the data with the functions `class`, `str`, and `head`. We can see here that the variables are all in numerical variable type which is what we want for analysis.

```
data2.ts<-ts(data2[,2:3],start=c(2003,3),frequency=4)
ppi.ts<-ts(data2$price,start=c(2003,3),frequency=4)
change.ts<-ts(data2$change,start=c(2003,3),frequency=4)
```

To process the data for further analysis, we must first convert the objects in the data frame `data1` to time series objects. By using the `ts` function, we convert each variables in the dataset into time series objects and name the new objects `data2.ts`, `ppi.ts`, and `change.ts`. Since we know that the dataset starts from September 2003 and it contains quarter values, we can write the arguments `start` to 2003 and 3, and `frequency` to 4 for all conversions.

2.3.0 Data Visualisation

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

```
plot(data2.ts,main="Time Series Plot for Residential PPI and Population Change",yax.flip=T)
```

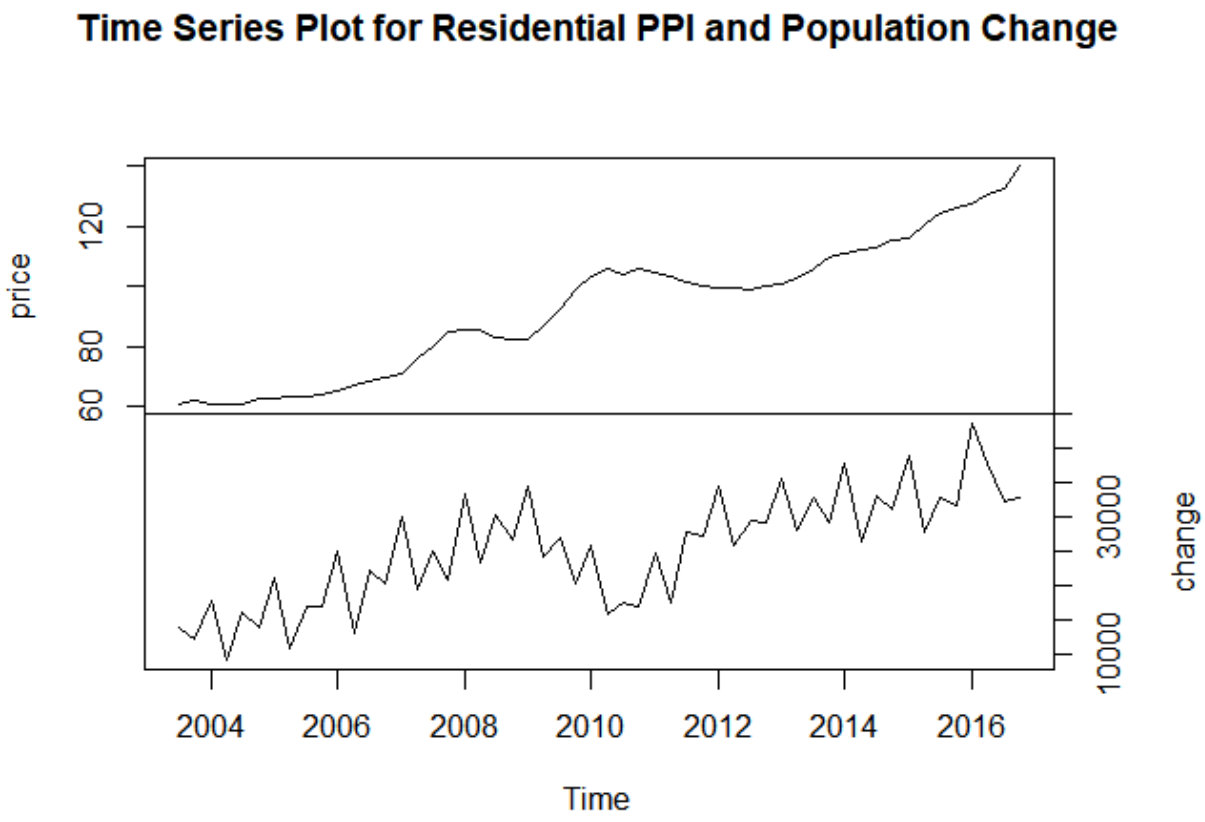

Figure 25

By `Figure 25`, which shows the time series plot of Residential Property Price Index and Population Change, we can make the following observations:

- The series has an upward trend over the time period.
- There is obvious seasonality, with higher values in June and July, and lower values during December and January.
- In both the price and change data there seems to be a correlation in their trends.

The sample CCF is displayed as follows:

```
ccf(as.vector(ppi.ts),as.vector(change.ts),ylab="CCF",main="Sample CCF between Residential PPI
          and Population Change")
```

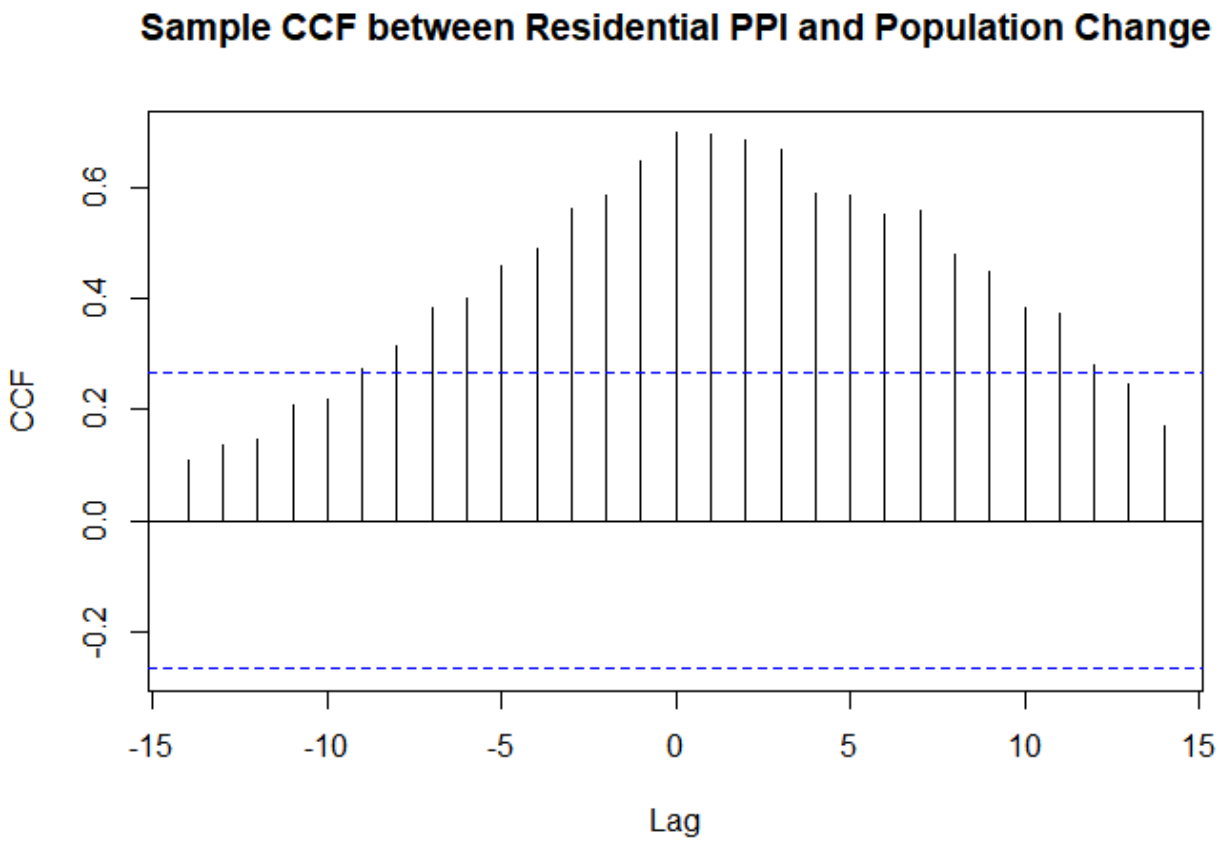

Figure 26

In `Figure 26` above, we can observe that a significant number of lags are significantly different from zero according to the *1.96/n* rule. This suggests that there is a strong relationship between Residential Property Price Index and Population Change. The nonstationarity in both the series are more likely the cause of the spurious correlations found between the two series.

2.4.0 Existence of Non-Stationary Test

In this step, we have to ensure that both Residential Property Price Index and Population Change are stationary. To test the existence of non-stationary, we will conduct the test first by plotting the ACF and PACF plots using the functions `ACF` and `PACF`. Then, we shall conduct the Augmented Dickey-Fuller test (ADF) to test the hypothesis. The Augmented Dickey-Fuller test will use the function `adf.test`.

```
par(mfrow=c(1,2))
acf(ppi.ts,main="Sample ACF for Residential PPI Series",cex.main=0.65)
acf(change.ts,main="Sample ACF for Population Change",cex.main=0.65)
```

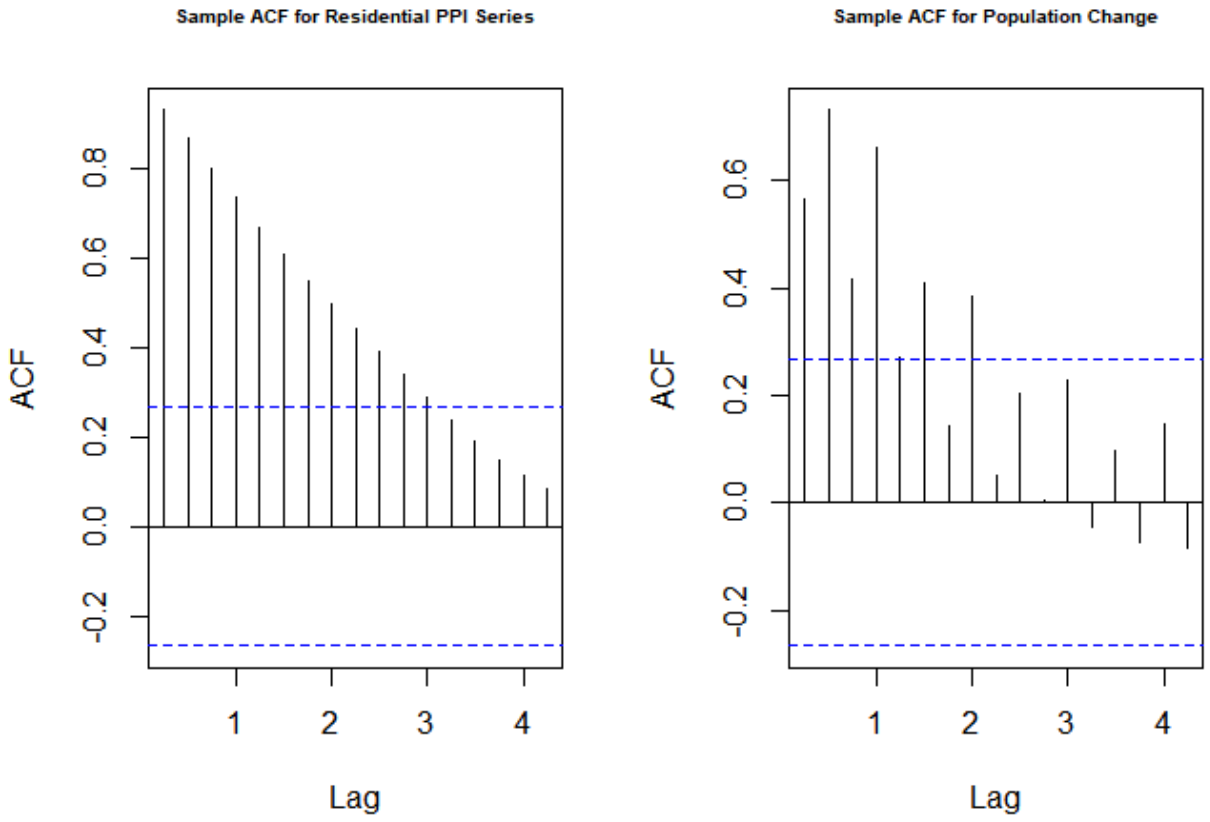

Figure 27

`Figure 27` above shows the ACF plot for Residential PPI and Population Change. Here we can observe that there is an decaying pattern for both series and most lags are above the 95% confidence interval in the ACF plot.

```
par(mfrow=c(1,2))
pacf(ppi.ts,main="Sample PACF for Residential PPI Series",cex.main=0.65)
pacf(change.ts,main="Sample PACF for Population Change",cex.main=0.65)
```

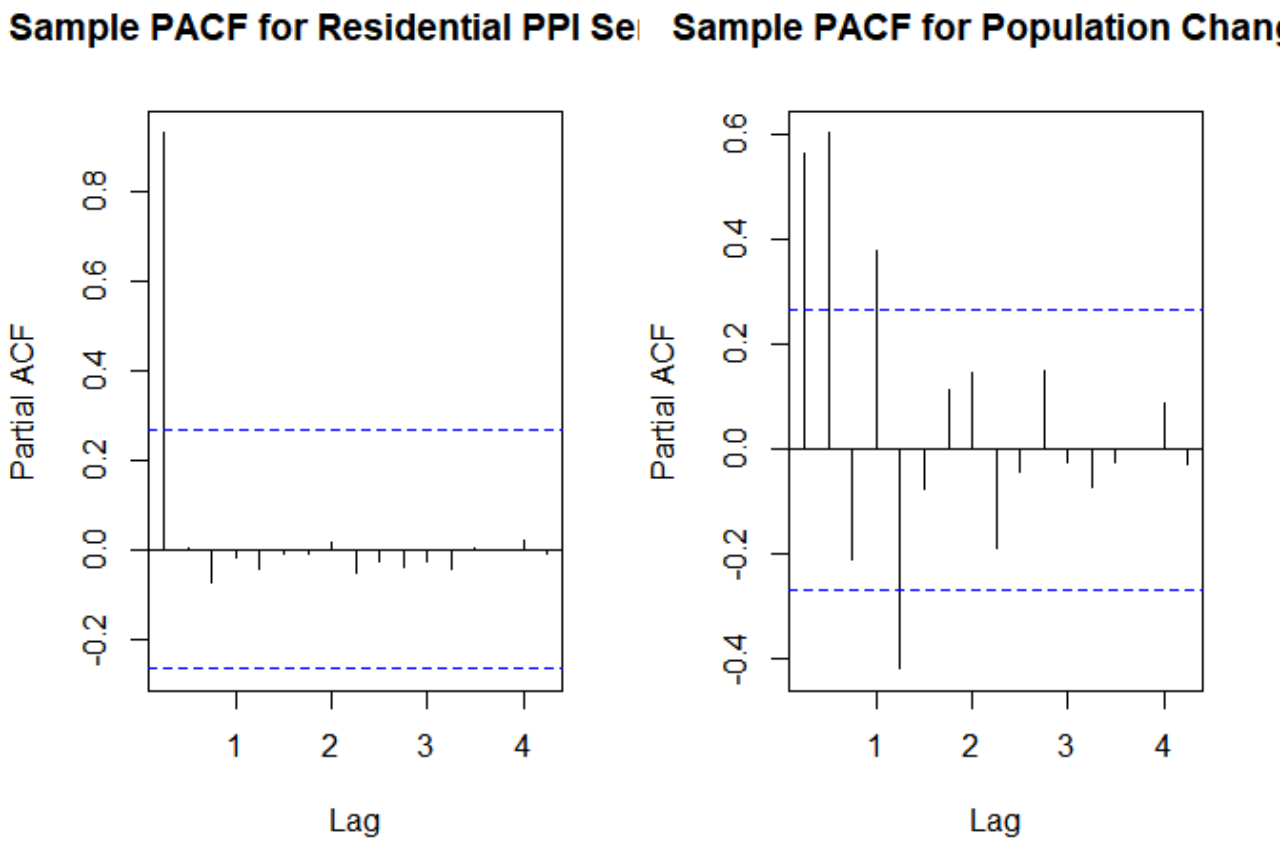

Figure 28

`Figure 28` above shows the ACF plot for Residential PPI and Population Change. In the PACF plot, we can observe that the first lag for both series is significant, giving evidence that the time series is non-stationary. However, further testing needs to be conducted to be sure.

```
#Augmented Dickey-Fuller Test
adf.test(ppi.ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ppi.ts
## Dickey-Fuller = -1.3264, Lag order = 3, p-value = 0.8458
## alternative hypothesis: stationary
```

```
adf.test(change.ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  change.ts
## Dickey-Fuller = -1.603, Lag order = 3, p-value = 0.7344
## alternative hypothesis: stationary
```

In the above output, we used the ADF test for both series. For both series, we can conclude that the series are non stationary at 5% level of significance as p value is greater than 0.05. Since both series are non-stationary, we have to convert the series into a stationary series before proceeding with further analysis. In the next steps, we will use ordinary differencing operation as it's main aim is to deal with non-stationary.

```
ppi.diff<-diff(ppi.ts,differences=3)
change.diff<-diff(change.ts,differences=3)

par(mfrow=c(1,2))
acf(ppi.diff,main="Sample ACF for Residential PPI Series",cex.main=0.65)
acf(change.diff,main="Sample ACF for Population Change",cex.main=0.65)
```

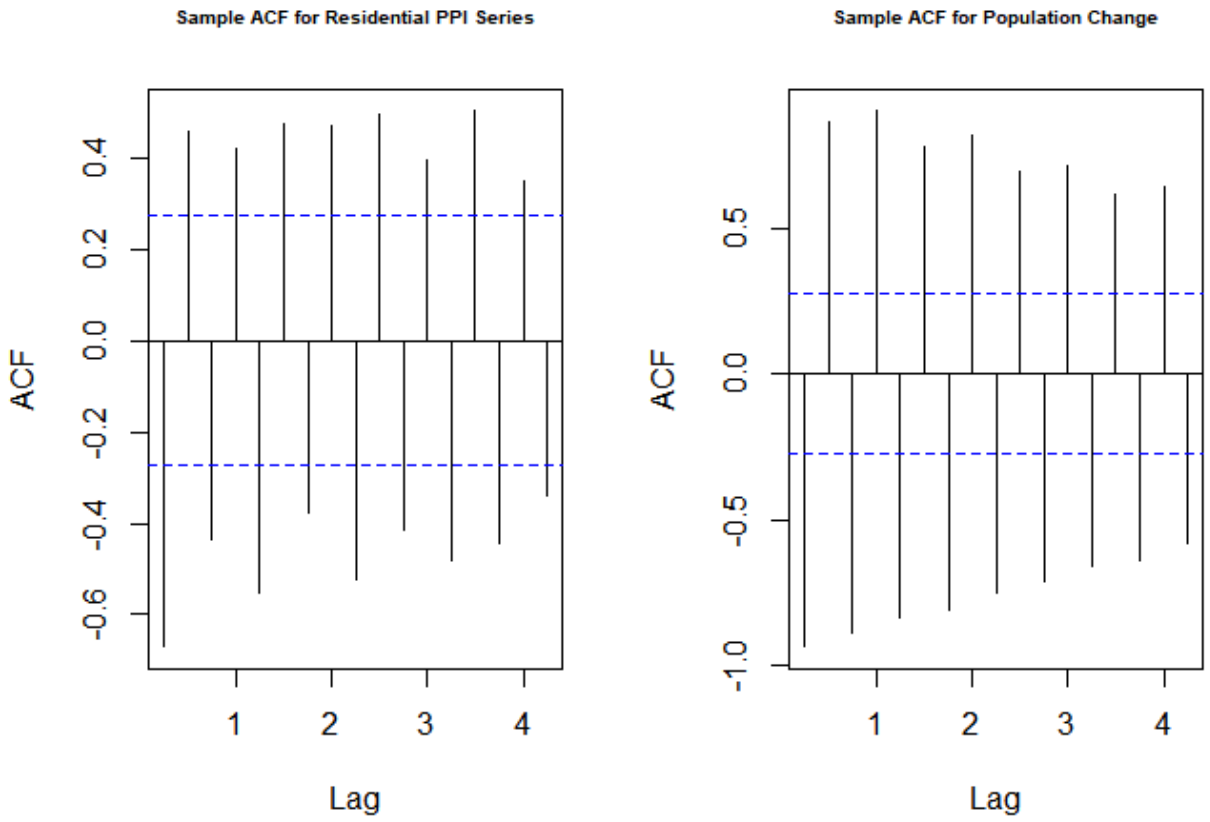

Figure 29

`Figure 29` above shows the ACF plot for differenced Residential PPI and Population Change. Here we can observe that there is no decaying pattern for both series and all lags are above the 95% confidence interval in the ACF plot.

```
par(mfrow=c(1,2))
pacf(ppi.diff,main="Sample PACF for Residential PPI Series",cex.main=0.65)
pacf(change.diff,main="Sample PACF for Population Change",cex.main=0.65)
```

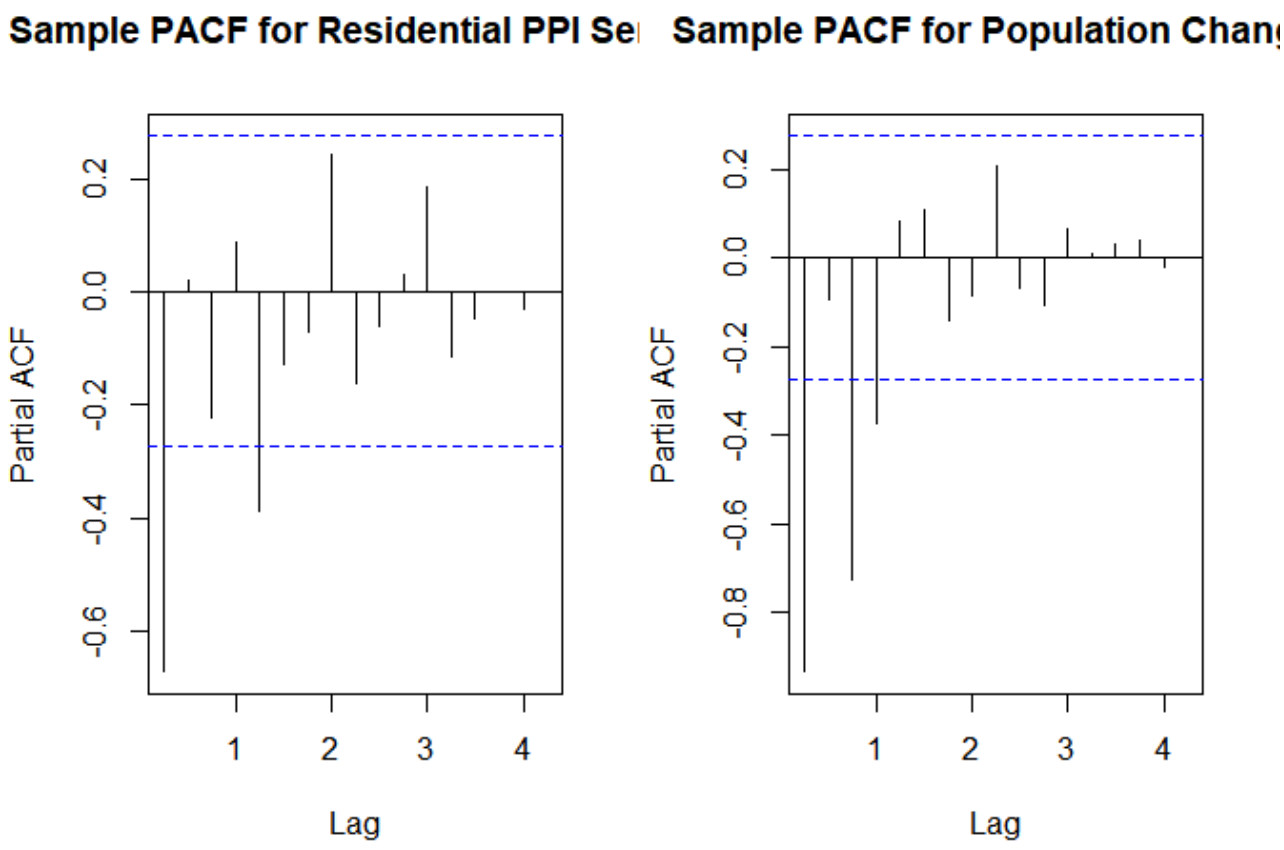

Figure 30

`Figure 30` above shows the ACF plot for differenced Residential PPI and Population Change . In the PACF plot, we can observe that the first lag for both series is significant, giving evidence that the time series is non-stationary. However, further testing needs to be conducted to be sure.

```
#Augmented Dickey-Fuller Test
adf.test(ppi.diff)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ppi.diff
## Dickey-Fuller = -3.9847, Lag order = 3, p-value = 0.01709
## alternative hypothesis: stationary
```

```
adf.test(change.diff)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  change.diff
## Dickey-Fuller = -11.253, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

In the above output, we used the ADF test for both series. For both series, we can conclude that the series are stationary at 5% level of significance as p value is less than 0.05. This means that we have successfully dealt with non-stationary and can proceed with the prewhitening process.

2.5.0 Prewhitening

In the previous sections we observed that the data was strongly autocorrelated and is difficult to assess the dependence between the two process. Now we will use the prewhitening process to disentangle the linear association between Residential PPI and Population Change.

```
prewhiten(as.vector(ppi.diff),as.vector(change.diff),ylab="CCF",main="Sample CCF after
          prewhitening of the Residential PPI and Population Change")
```

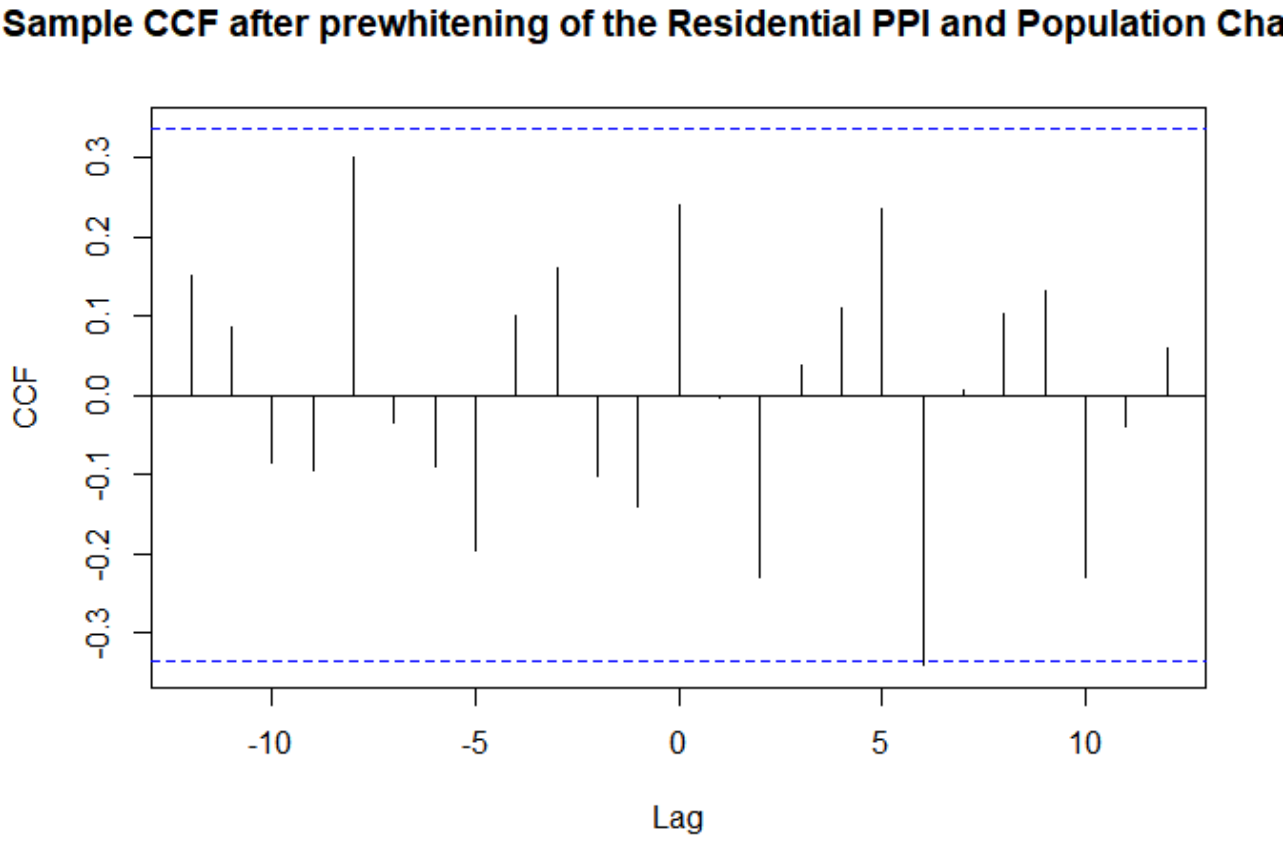

Figure 31

In `Figure 31` above shows the sample CCF after prewhitening of the Residential PPI and Population Change. Here we can observe that there is no significant correlation between both series. The significant correlations in the above plots can be related to the false alarm rate of the CCF.

2.6.0 Conclusion

In conclusion, it seems that Residential Property Price Change and Population Change are largely uncorrelated, and the strong cross-correlation pattern found between the raw data series is indeed spurious.