# Assignment 2 MATH1307

2023-09-20

### Analysing and forecasting the horizontal solar radiation levels and analysing the correlation between quarterly Residential Property Price Index (PPI) in Melbourne and quarterly population change over the previous quarter in Victoria

### Van Thai Phan - s3818387

# Introduction

This report consists of two parts. The first part aims to analyse and forecast the amount of horizontal solar radiation reaching the ground at a particular location over the globe. In order to achieve this task, I will apply time series regression method to fit distributed lag models with the monthly precipitation series as an independent explanatory series. Moreover, I will also be using exponential smoothing methods. The expectation is to find the best model to provide an accurate forecast of the amount of horizontal solar radiation.

On the other hand, the second part of the report will analyse the correlation between quarterly Residential Property Price Index (PPI) in Melbourne and quarterly population change over the previous quarter in Victoria between September 2003 and December 2016. The data will be explored and investigated in order to demonstrate whether the correlation between these two series is spurious or not.

# Task 1

# Data description

The first step is to load the data and create time series for solar radiation and precipitation values.

```
Solar_data <- read.csv("/Users/macbookair/Desktop/data1.csv")
head(Solar_data)
```

```
##       solar   ppt
## 1  5.051729 1.333
## 2  6.415832 0.921
## 3 10.847920 0.947
## 4 16.930264 0.615
## 5 24.030797 0.544
## 6 26.298202 0.703
```

```
Solar_x <- read.csv("/Users/macbookair/Desktop/data.x.csv")
Solar_x_ts <- ts(Solar_x)
```

```
solar <- ts(Solar_data$solar, start =c(1960,1), frequency = 12)
head(solar)
```

```
##            Jan       Feb       Mar       Apr       May       Jun
## 1960  5.051729  6.415832 10.847920 16.930264 24.030797 26.298202
```

```
precipit <- ts(Solar_data$ppt, start = c(1960,1), frequency = 12)
head(precipit)
```

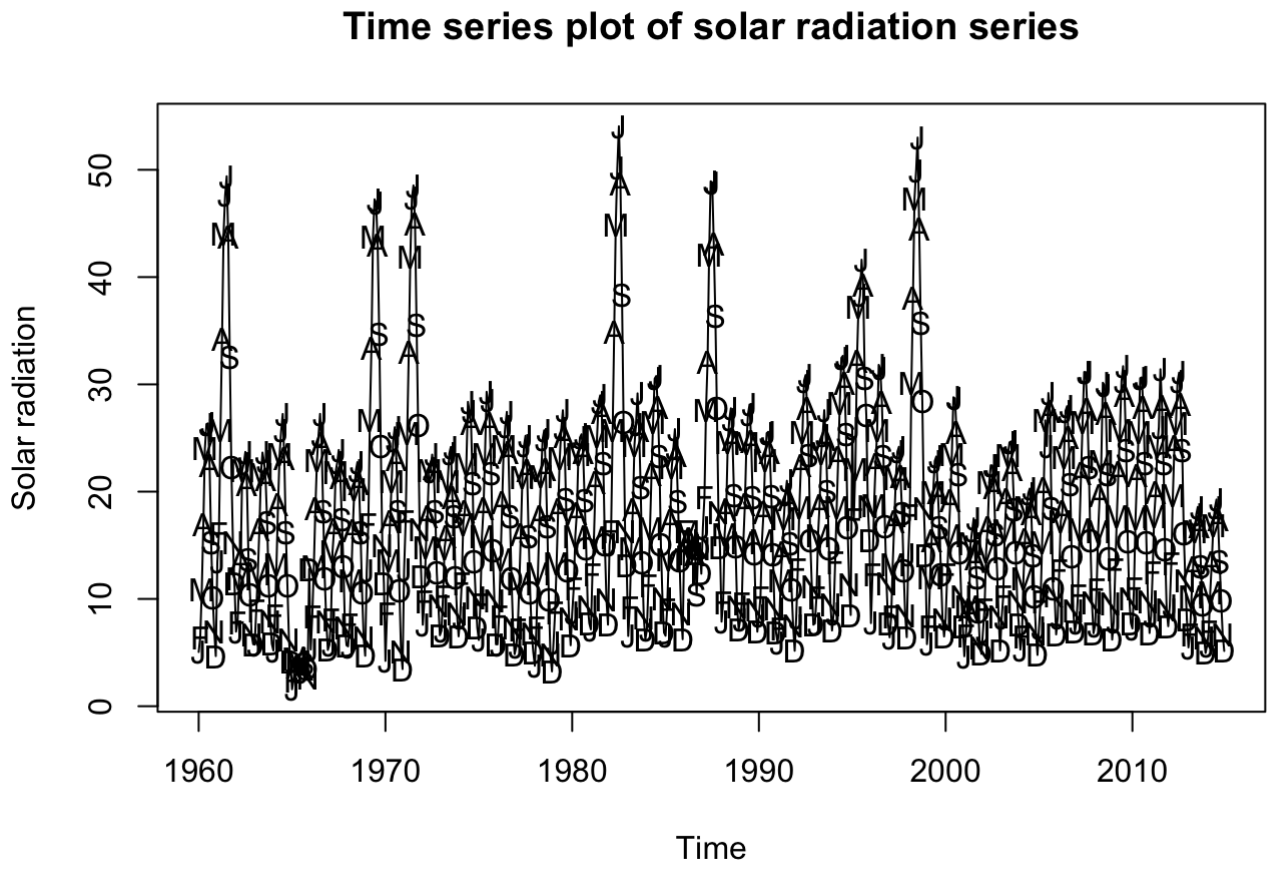
```
##        Jan   Feb   Mar   Apr   May   Jun
## 1960 1.333 0.921 0.947 0.615 0.544 0.703
```

```
Solar_data_ts <- ts(Solar_data, start=c(1960,1), frequency= 12)
head(Solar_data_ts)
```

```
##              solar   ppt
## Jan 1960  5.051729 1.333
## Feb 1960  6.415832 0.921
## Mar 1960 10.847920 0.947
## Apr 1960 16.930264 0.615
## May 1960 24.030797 0.544
## Jun 1960 26.298202 0.703
```

# Data visualisation and exploration

```
plot(solar,  main = "Time series plot of solar radiation series", ylab = "Solar radia
tion", xlab = "Time")
points(solar, x=time(solar), pch = as.vector(season(solar)))
```

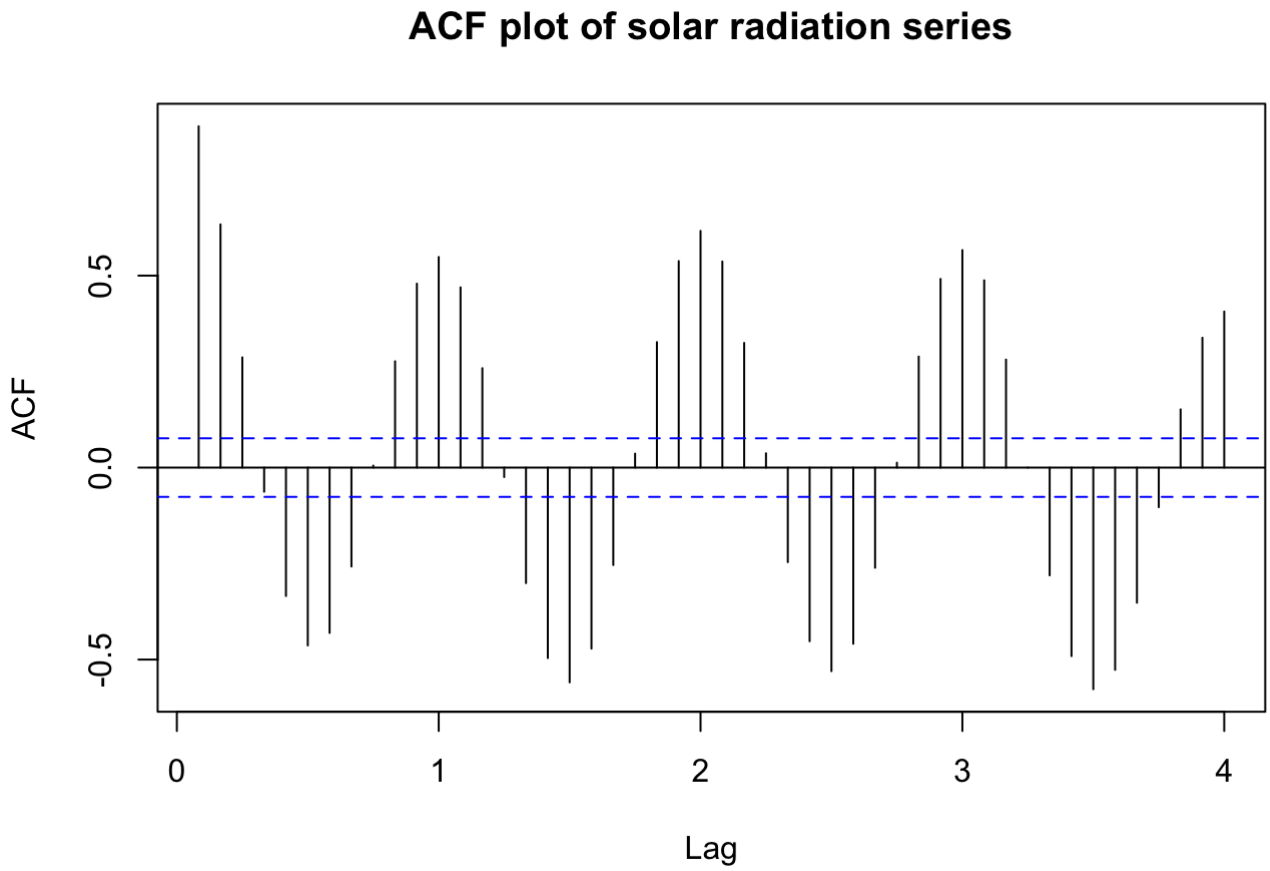



From the above plot, the following characteristics can be observed:

- There is no noticable trend in the series
- Seasonality is obviously present, with the highest values observed in May, June, July, and August while the lowest values are observed in December and January.
- There is no changing variance
- There is an obvious intervention point around 1965 where solar radiation values are extremely low

Next, a ACF plot will be created and a ADF test, along with a PP test will be conducted to further explore the solar radiation time series.

```
acf(solar, lag.max = 48, main="ACF plot of solar radiation series")
```




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

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

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

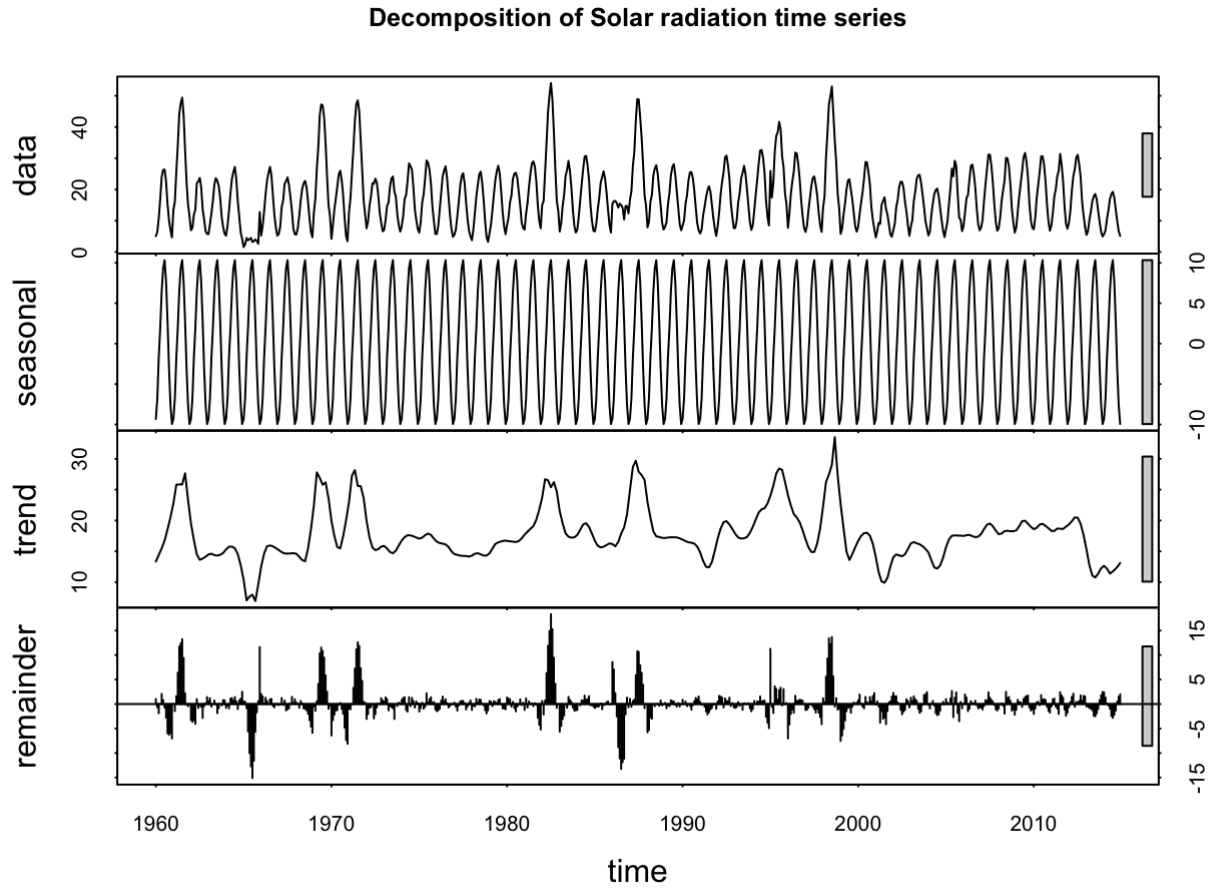
```
pp.test(solar)
```

```
## Warning in pp.test(solar): p-value smaller than printed p-value
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  solar
## Dickey-Fuller Z(alpha) = -138.28, Truncation lag parameter = 6, p-value
## = 0.01
## alternative hypothesis: stationary
```

The ACF plot shows a strong seasonality pattern. The ADF test and the PP test all have p values smaller than 0.05, therefore, it is safe to conclude that the time series is stationary. Next, the time series will be decompsed.

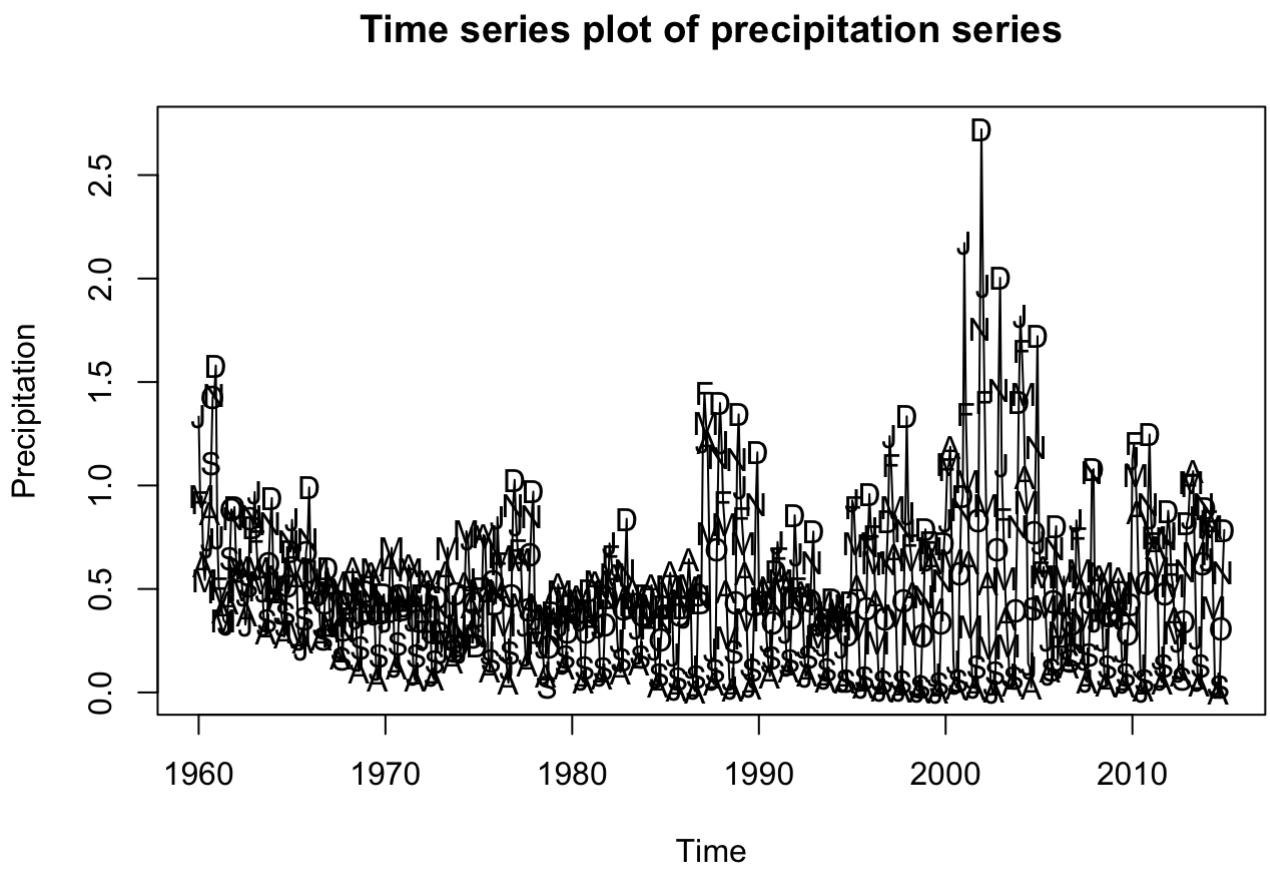
```
solar.decom = stl(solar, t.window=15, s.window="periodic", robust=TRUE)
plot(solar.decom, main="Decomposition of Solar radiation time series")
```




According to the decomposition plot, the Solar radiation time series has strong seasonality but no obvious trend.

Now, the process will be repeated for the Precipitaion time series, which will be used as the predictor for distributed lag models.

```
plot(precipit, main ="Time series plot of precipitation series", ylab="Precipitatio
n", xlab = "Time")
points (precipit, x= time(precipit), pch = as.vector(season(precipit)))
```

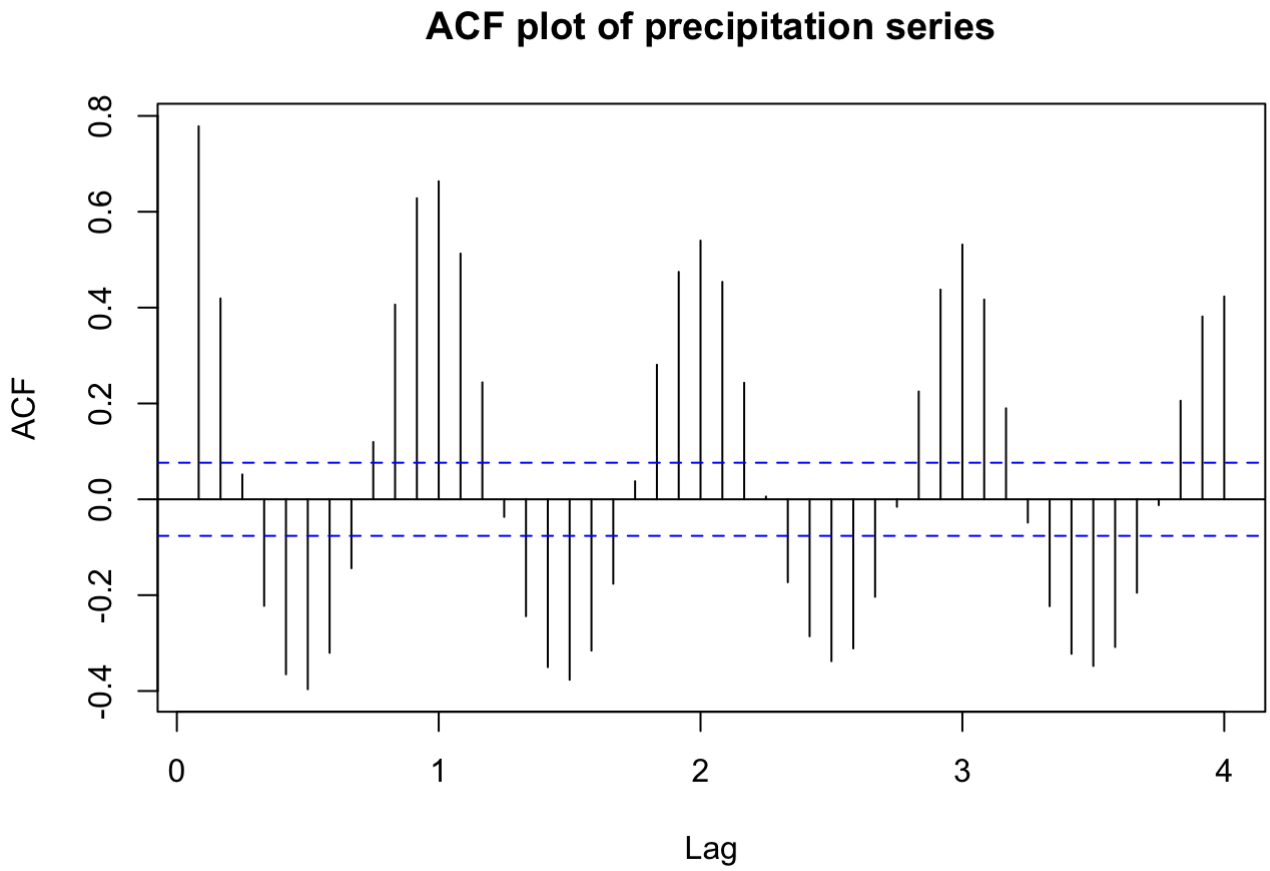



From the above plot, the following characteristics can be observed:

- There is no noticable trend in the series
- Seasonality is obviously present, with the highest values observed in December and January and August while the lowest values are observed in July, August, and September.
- There is no changing variance
- There is no intervention point

Next, a ACF plot will be created and a ADF test, along with a PP test will be conducted to further explore the solar radiation time series.

```
acf(precipit,lag.max = 48, main = "ACF plot of precipitation series")
```




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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  precipit
## Dickey-Fuller = -3.2594, Lag order = 28, p-value = 0.07769
## alternative hypothesis: stationary
```

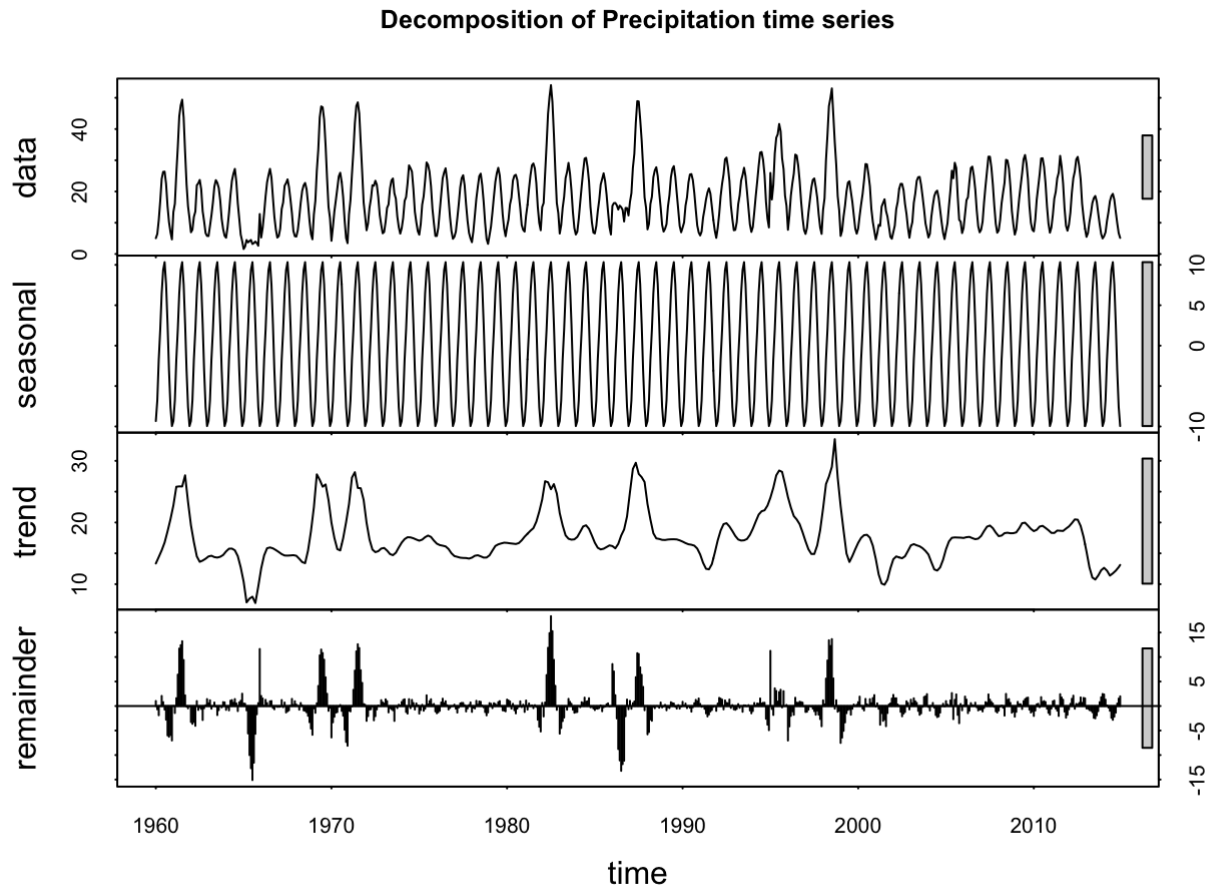
```
pp.test(precipit)
```

```
## Warning in pp.test(precipit): p-value smaller than printed p-value
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  precipit
## Dickey-Fuller Z(alpha) = -154.28, Truncation lag parameter = 6, p-value
## = 0.01
## alternative hypothesis: stationary
```

The ACF plot shows a strong seasonality pattern. The ADF test and the PP test all have p values smaller than 0.05, therefore, it is safe to conclude that the time series is stationary. Next, the time series will be decompsed.

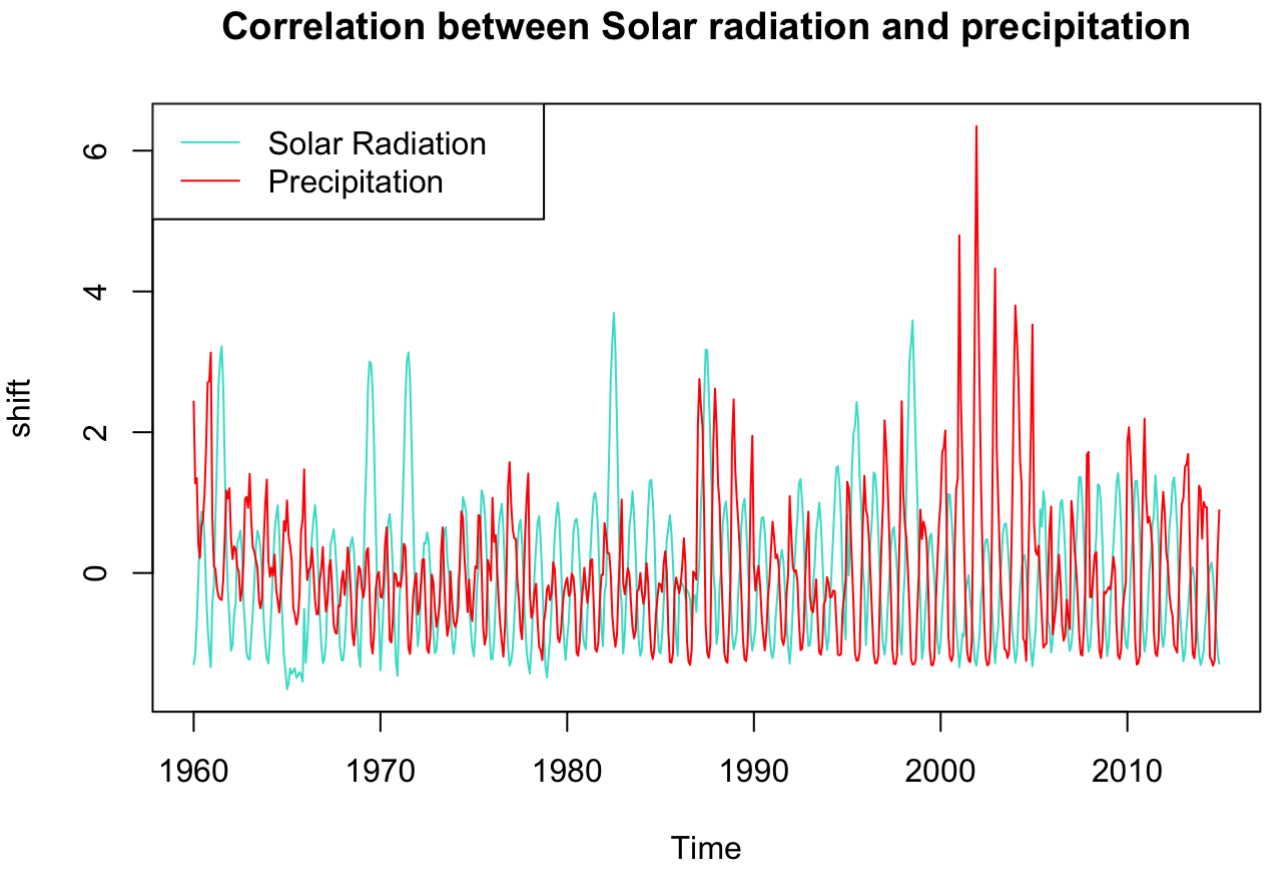
```
precipit.decom = stl(solar, t.window=15, s.window="periodic", robust=TRUE)
plot(precipit.decom, main="Decomposition of Precipitation time series")
```




According to the decomposition plot, the Solar radiation time series has strong seasonality but no obvious trend.

The correlation of the solar radiation series and the precipitation series is shown in the following plot:

```
shift<- scale(Solar_data_ts)
plot(shift, plot.type="s",col=c("turquoise", "red"),main= "Correlation between Solar
radiation and precipitation")
legend("topleft", lty=1, text.width = 14, col = c("turquoise", "red"), c("Solar Radia
tion", "Precipitation"))
```




From the plot, we can see that both series' high values corespond to the other's low values. Thus, indicating a negative correlation. To confirm this, the correlation coefficient is calculated.

```
cor(solar,precipit)
```

```
## [1] -0.4540277
```

The correlation coefficient is -0.4540277, thus confirming that the two series are negatively correlated.

# Time series regression models

# Finite distributed lag model

The first model to be tested is the Finite distributed lag model. In order to select the most accurate model, we will create a loop that calculates AIC, BIC, and MASE. The model with the lowest values will be selected.

```
for (i in 1:12){
  model_a <- dlm(x = Solar_data$ppt, y = Solar_data$solar, q = i)
  cat("q =", i, "AIC =", AIC(model_a$model), "BIC =", BIC(model_a$model), "MASE =", M
ASE(model_a)$MASE, "\n")
}
```

```
## q = 1 AIC = 4728.713 BIC = 4746.676 MASE = 1.688457
## q = 2 AIC = 4712.649 BIC = 4735.095 MASE = 1.675967
## q = 3 AIC = 4688.551 BIC = 4715.478 MASE = 1.662703
## q = 4 AIC = 4663.6 BIC = 4695.003 MASE = 1.646357
## q = 5 AIC = 4644.622 BIC = 4680.499 MASE = 1.613848
## q = 6 AIC = 4637.489 BIC = 4677.837 MASE = 1.607532
## q = 7 AIC = 4632.716 BIC = 4677.532 MASE = 1.607042
## q = 8 AIC = 4625.986 BIC = 4675.267 MASE = 1.604806
## q = 9 AIC = 4615.084 BIC = 4668.827 MASE = 1.593121
## q = 10 AIC = 4602.658 BIC = 4660.858 MASE = 1.577996
## q = 11 AIC = 4590.961 BIC = 4653.617 MASE = 1.562127
## q = 12 AIC = 4578.787 BIC = 4645.895 MASE = 1.5516
```

```
fin_dlm <- dlm(x=Solar_data$ppt, y=Solar_data$solar, q=12)
summary(fin_dlm)
```

```
##
## 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
```

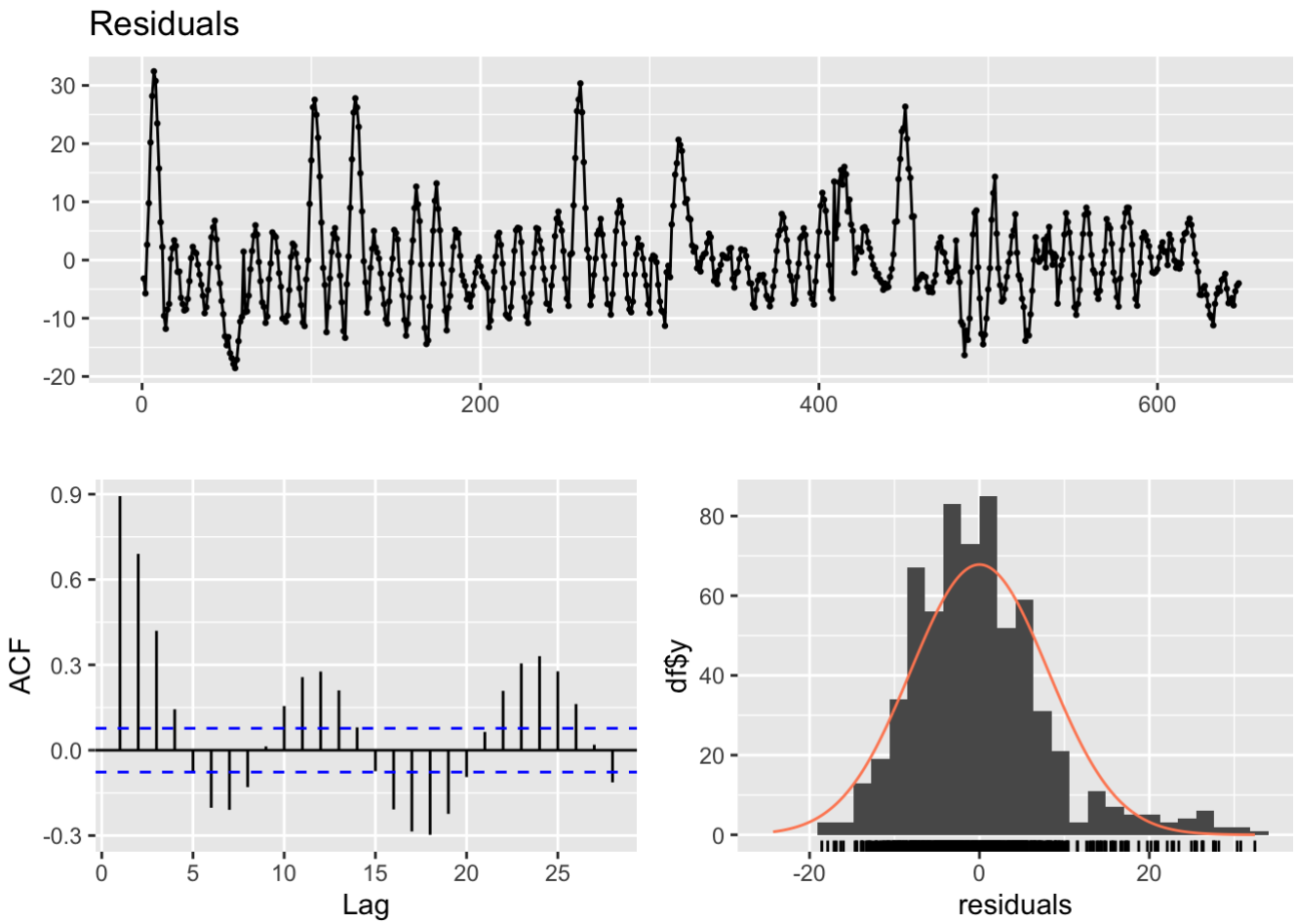
```
vif(fin_dlm$model)
```

```
##      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
```

The AIC, BIC, and MASE values decrease as q value increases, therefore, the finite DLM that was fitted has a number of lags of 12.

The R-squared value is 0.3077, this means that the model is only accountable for 30.7% of the variability in solar radiation, which is low. The F-test gave a p-value < 0.05, thus the model is statistically significant at 5% level. All VIF values are smaller than 10, indicating that the model does not suffer from multicollinearity. Next, a residual check is conducted.

```
checkresiduals(fin_dlm$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 17
##
## data:  Residuals
## LM test = 594.22, df = 17, p-value < 2.2e-16
```

```
shapiro.test(residuals(fin_dlm$model))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(fin_dlm$model)
## W = 0.94168, p-value = 2.922e-15
```

The p-value output for the Breusch-Godfrey test is smaller than 0.05 indicates that there is serial correlation in the residuals. The null hypothesis of normality is rejected by the Shapiro-Wilk test, which also has a p-value smaller than 0.05. The residuals are not normally distributed. Overall, this model is not ideal.

# Polynomial distributed lag model

The polynomial distributed lag model has a q=12 similar to the finite distributed lag model before.

```
poly_dl <- polyDlm(x=as.vector(Solar_data$ppt), y=as.vector(Solar_data$solar), q=12,k
=2)
```

```
## 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(poly_dl)
```

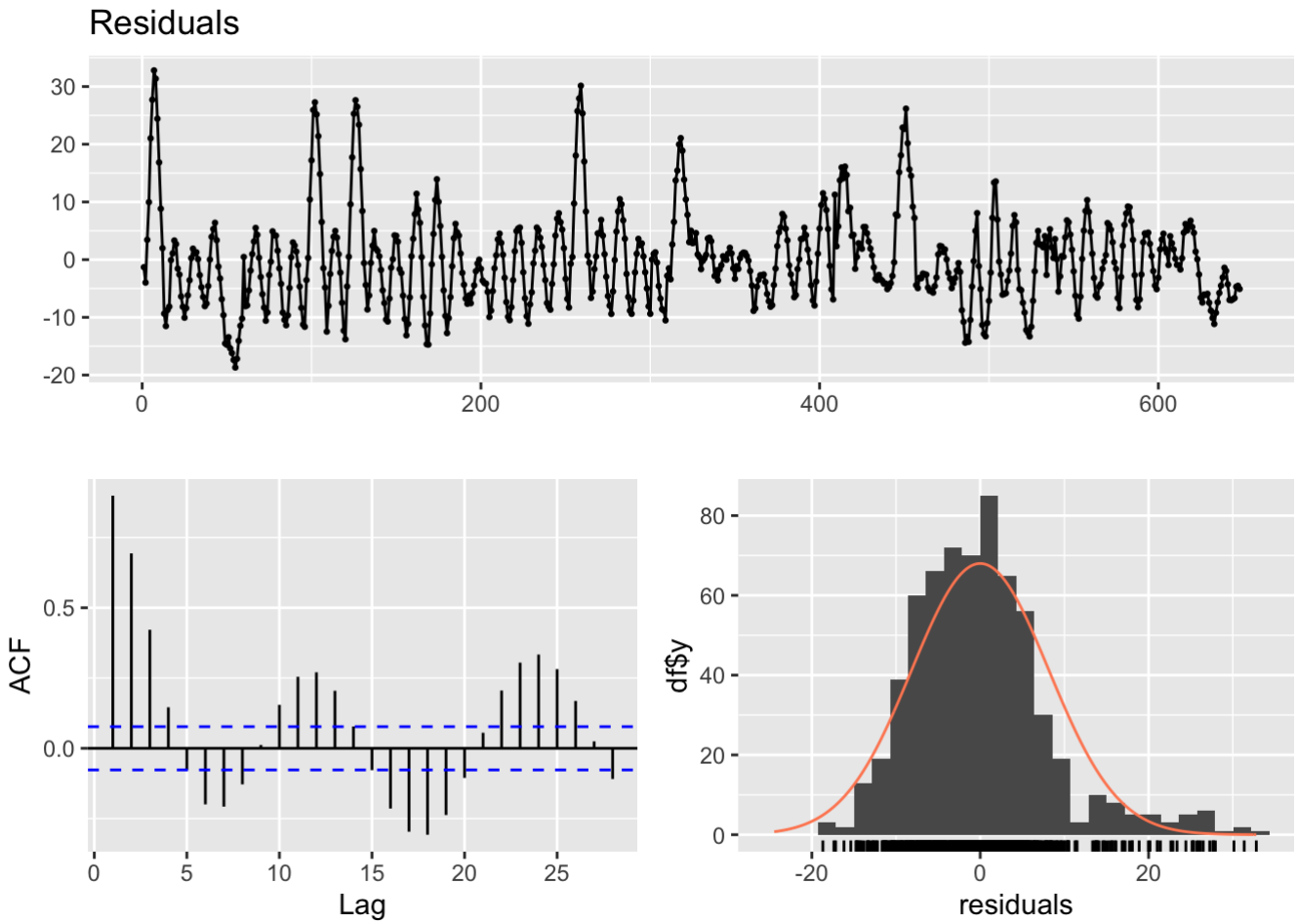
```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.689  -5.452  -0.686   4.129  32.797
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 19.22679    1.10784  17.355  < 2e-16 ***
## z.t0        -1.81945    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
```

```
vif(poly_dl$model)
```

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

The R-squared value is 0.3087, this means that the model is only accountable for 30.8% of the variability in solar radiation, which is low. The F-test gave a p-value < 0.05, thus the model is statistically significant at 5% level. The model suffers from multicollinearity as the VIF values of z.t1 and z.t2 are greater than 10.

```
checkresiduals(poly_dl$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 583.79, df = 10, p-value < 2.2e-16
```

```
shapiro.test(residuals(poly_dl$model))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(poly_dl$model)
## W = 0.94141, p-value = 2.679e-15
```

The p-value output for the Breusch-Godfrey test is smaller than 0.05 indicates that there is serial correlation in the residuals. The null hypothesis of normality is rejected by the Shapiro-Wilk test, which also has a p-value smaller than 0.05. The residuals are not normally distributed.

Overall, the model is not suitable as it has low explainability and suffers from multicollinearity.

# Koyck model

The Koyck model is implemented with precipitation as the predictor.

```
K_model = koyckDlm(x=as.vector(Solar_data$ppt), y= as.vector(Solar_data$solar))
summary(K_model$model, diagnostics=T)
```

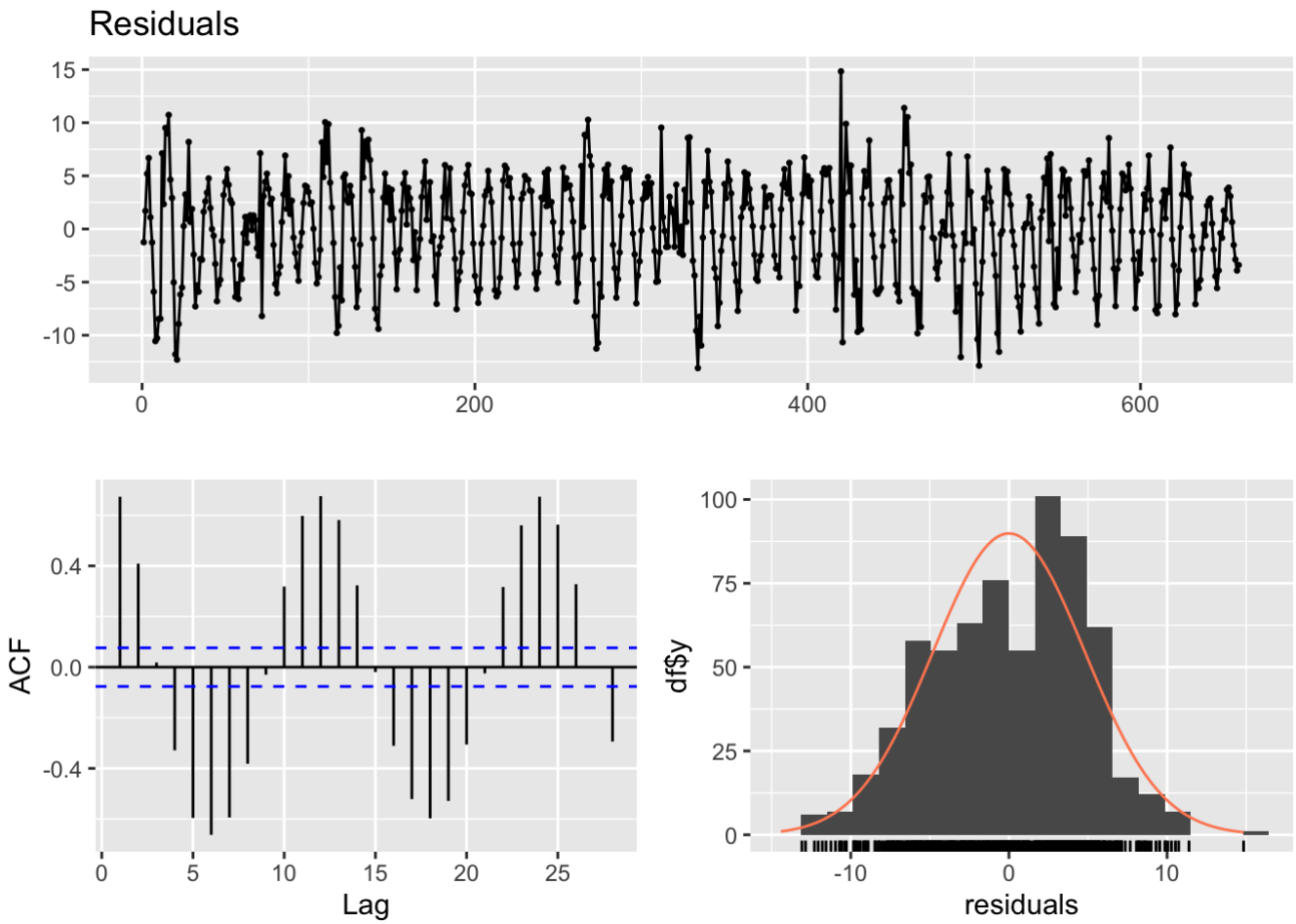
```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.0926  -3.5961   0.3176   3.6103  14.8399
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.23925    0.76549  -2.925  0.00356 **
## Y.1          0.98546    0.02424  40.650  < 2e-16 ***
## X.t          5.34684    0.84383   6.336 4.37e-10 ***
##
## Diagnostic tests:
##                   df1 df2 statistic p-value
## Weak instruments   1 656     710.7  <2e-16 ***
## Wu-Hausman         1 655     146.8  <2e-16 ***
## Sargan             0  NA        NA      NA
## ---
## 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
```

```
vif(K_model$model)
```

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

The R-squared value is 0.7591, this means that the model is accountable for 75.9% of the variability in solar radiation, which is higher than the other tested modesl but still not ideal. The weak instruments test gave a p-value < 0.05, thus the model is statistically significant at 5% level. Similarly, the Wu-Hausman test also produced a p-value smaller than 0.05, thus the correlation between the explanatory variable and the error term is significant at 5%. The model does not suffer from multicollinearity as all VIF values are smaller than 10.

```
checkresiduals(K_model$model)
```




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

```
shapiro.test(residuals(K_model$model))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(K_model$model)
## W = 0.98487, p-value = 2.468e-06
```

The low p-value of the Ljung-Box test suggests that the residuals have serial correlation. However, the Shapiro-Wilk Normality Test has a p-value less than 0.05, which means that the residuals are not normally distributed. Overall, the Koyck model is not ideal.

# Dynamic lag model

The following code creates two dynamic lag models:

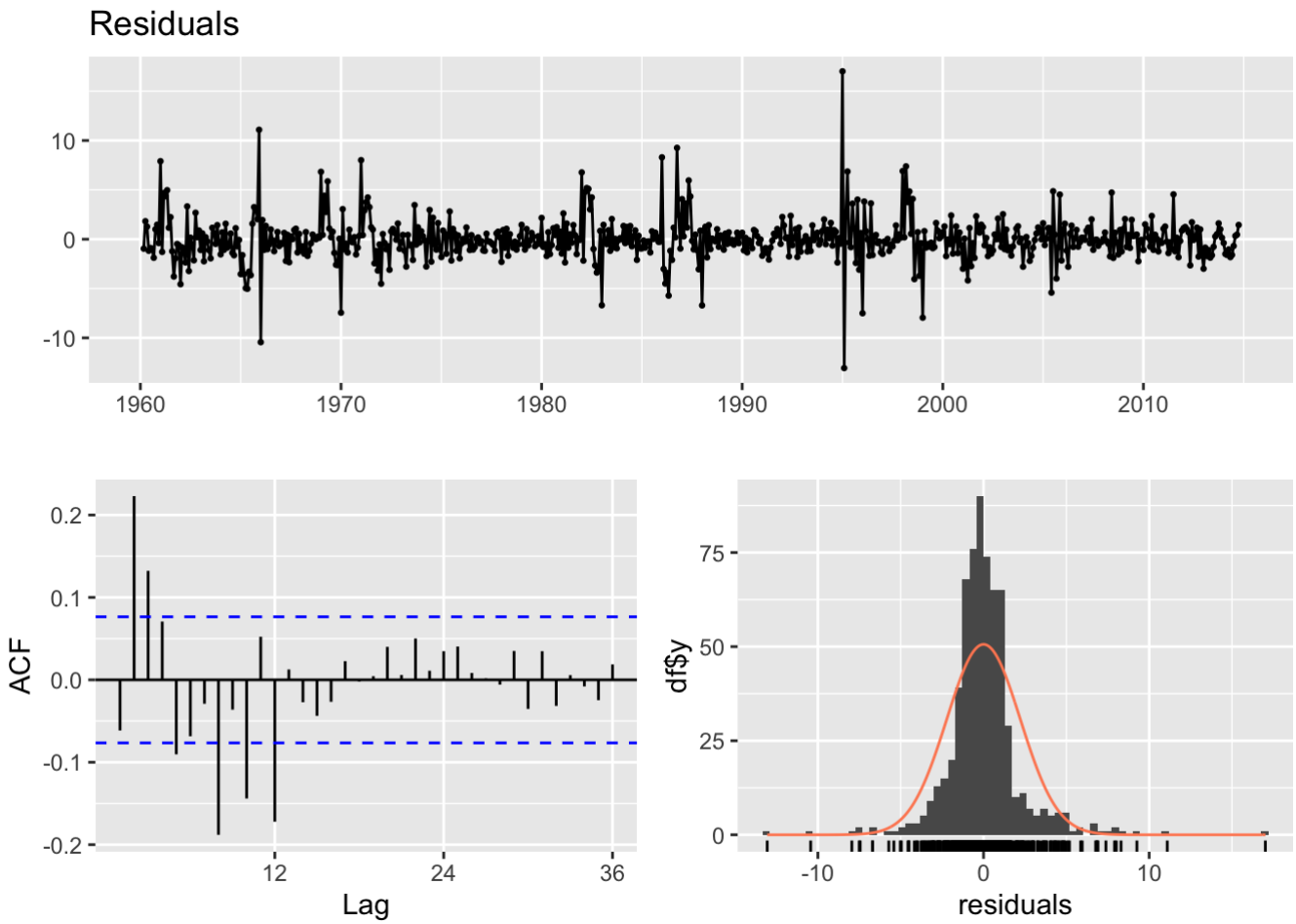
```
Y.t = solar
X.t = precipit
P.t.1 <- stats::lag(X.t, 1)
P.t.2 <- stats::lag(X.t, 2)

dyn_mod.1 = dynlm(Y.t ~ L(Y.t, 1) + L(Y.t, 2) +
              X.t + P.t.1 + P.t.2 +
              trend(Y.t) + season(Y.t))
summary(dyn_mod.1)
```

```
##
## Time series regression with "ts" data:
## Start = 1960(3), End = 2014(10)
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, 1) + L(Y.t, 2) + X.t + P.t.1 + P.t.2 +
##     trend(Y.t) + season(Y.t))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.0585  -1.0150  -0.0997   0.8648  17.0151
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.9078154  0.4525069   4.216 2.84e-05 ***
## L(Y.t, 1)       1.1302165  0.0387004  29.204  < 2e-16 ***
## L(Y.t, 2)      -0.2122329  0.0387690  -5.474 6.32e-08 ***
## X.t            -0.2734789  0.5286760  -0.517 0.605133
## P.t.1          -0.1372028  0.7297513  -0.188 0.850926
## P.t.2           0.3020365  0.5306147   0.569 0.569407
## trend(Y.t)     -0.0007745  0.0055802  -0.139 0.889652
## season(Y.t)Feb  1.3011399  0.4493736   2.895 0.003916 **
## season(Y.t)Mar  3.8962860  0.4754901   8.194 1.38e-15 ***
## season(Y.t)Apr  2.3331103  0.5422689   4.302 1.95e-05 ***
## season(Y.t)May  4.0406188  0.5318166   7.598 1.08e-13 ***
## season(Y.t)Jun  2.0276368  0.5761754   3.519 0.000464 ***
## season(Y.t)Jul  0.7185998  0.5675312   1.266 0.205909
## season(Y.t)Aug -2.7298054  0.5645216  -4.836 1.67e-06 ***
## season(Y.t)Sep -4.4900429  0.5488609  -8.181 1.53e-15 ***
## season(Y.t)Oct -5.1203031  0.5146533  -9.949  < 2e-16 ***
## season(Y.t)Nov -4.2454772  0.4904380  -8.657  < 2e-16 ***
## season(Y.t)Dec -2.2254143  0.4490871  -4.955 9.26e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.251 on 638 degrees of freedom
## Multiple R-squared:  0.9486, Adjusted R-squared:  0.9472
## F-statistic: 692.1 on 17 and 638 DF,  p-value: < 2.2e-16
```

```
attr(dyn_mod.1,"class") = "lm"
MASE(dyn_mod.1)
```

```
##                 MASE
## dyn_mod.1 0.3630027
```

```
checkresiduals(dyn_mod.1)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 24
##
## data:  Residuals
## LM test = 108.78, df = 24, p-value = 9.215e-13
```

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

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(dyn_mod.1)
## W = 0.86589, p-value < 2.2e-16
```

```
vif(dyn_mod.1)
```

```
##                  GVIF Df GVIF^(1/(2*Df))
## L(Y.t, 1)   18.597096  1        4.312435
## L(Y.t, 2)   18.706138  1        4.325059
## X.t          4.519105  1        2.125819
## P.t.1        8.587424  1        2.930431
## P.t.2        4.544594  1        2.131805
## trend(Y.t)   1.003529  1        1.001763
## season(Y.t) 20.259641 11        1.146548
```

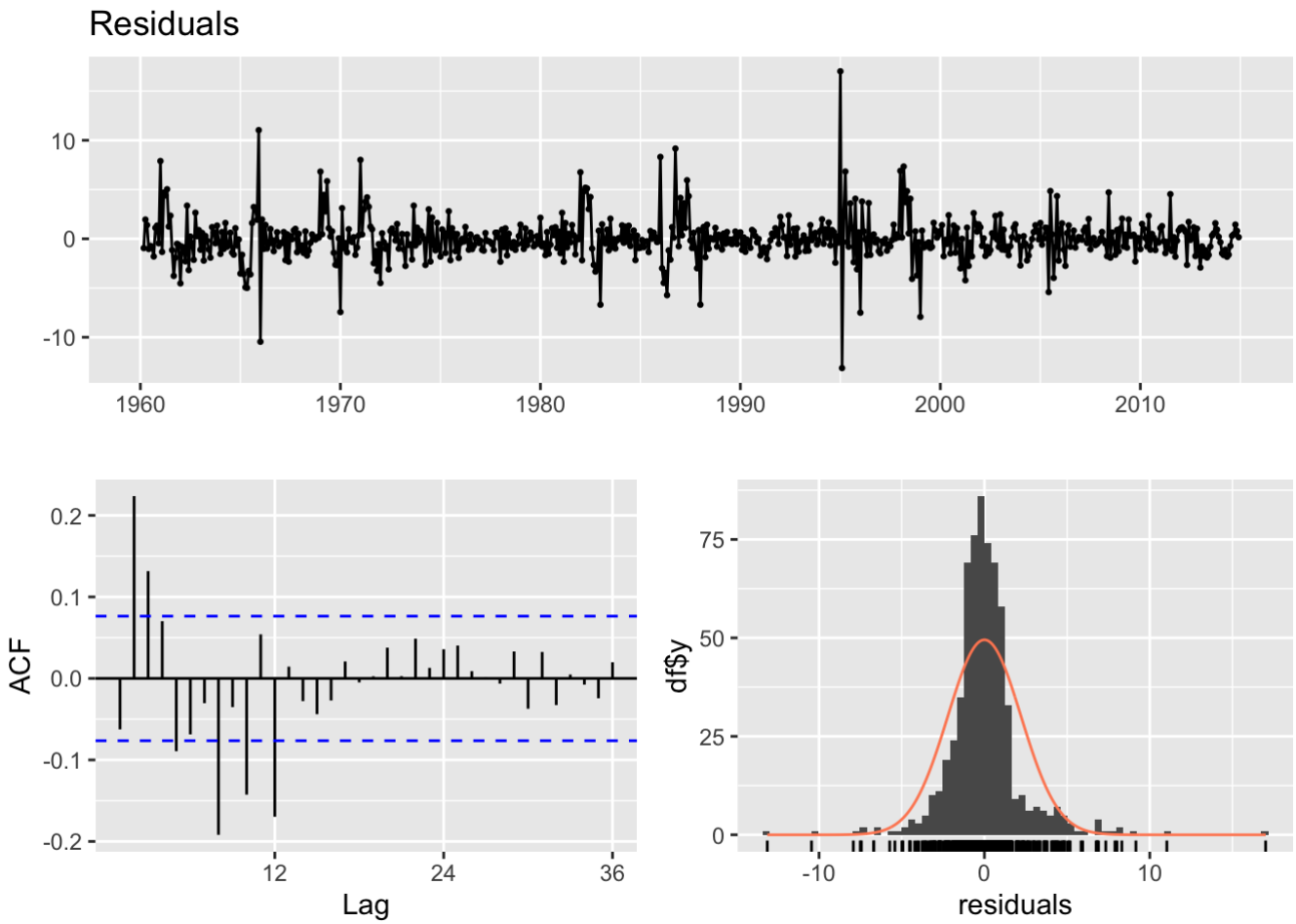
```
dyn_mod.2 = dynlm(Y.t ~ L(Y.t, 1) + L(Y.t, 2) +
              X.t +
              trend(Y.t) + season(Y.t))
summary(dyn_mod.2)
```

```
##
## Time series regression with "ts" data:
## Start = 1960(3), End = 2014(12)
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, 1) + L(Y.t, 2) + X.t + trend(Y.t) +
##     season(Y.t))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.1317  -0.9985  -0.0884   0.8337  17.0093
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.983337   0.436286   4.546 6.54e-06 ***
## L(Y.t, 1)       1.131707   0.038525  29.376  < 2e-16 ***
## L(Y.t, 2)      -0.214438   0.038526  -5.566 3.83e-08 ***
## X.t            -0.243985   0.324990  -0.751 0.453081
## trend(Y.t)     -0.000493   0.005536  -0.089 0.929069
## season(Y.t)Feb  1.299241   0.447869   2.901 0.003848 **
## season(Y.t)Mar  3.862038   0.469610   8.224 1.09e-15 ***
## season(Y.t)Apr  2.245544   0.524586   4.281 2.15e-05 ***
## season(Y.t)May  3.952718   0.512339   7.715 4.64e-14 ***
## season(Y.t)Jun  1.958115   0.564147   3.471 0.000553 ***
## season(Y.t)Jul  0.685603   0.563537   1.217 0.224201
## season(Y.t)Aug -2.680867   0.557469  -4.809 1.89e-06 ***
## season(Y.t)Sep -4.420256   0.529541  -8.347 4.29e-16 ***
## season(Y.t)Oct -5.038937   0.494734 -10.185  < 2e-16 ***
## season(Y.t)Nov -4.206824   0.470104  -8.949  < 2e-16 ***
## season(Y.t)Dec -2.221271   0.445286  -4.988 7.85e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.245 on 642 degrees of freedom
## Multiple R-squared:  0.9488, Adjusted R-squared:  0.9476
## F-statistic: 792.3 on 15 and 642 DF,  p-value: < 2.2e-16
```

```
attr(dyn_mod.2,"class") = "lm"
MASE(dyn_mod.2)
```

```
##                MASE
## dyn_mod.2 0.3633473
```

```
checkresiduals(dyn_mod.2)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 24
##
## data:  Residuals
## LM test = 109.78, df = 24, p-value = 6.173e-13
```

```
shapiro.test(residuals(dyn_mod.2))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(dyn_mod.2)
## W = 0.86658, p-value < 2.2e-16
```

```
vif(dyn_mod.2)
```

```
##                  GVIF Df GVIF^(1/(2*Df))
## L(Y.t, 1)   18.583169  1        4.310820
## L(Y.t, 2)   18.596147  1        4.312325
## X.t          1.719294  1        1.311218
## trend(Y.t)   1.002274  1        1.001136
## season(Y.t) 12.063071 11        1.119843
```

Looking at both of these models, they both have serial correlation due to their Breusch-Godfrey tests' p-value output being less than 0.05. However, looking at the residual plots of both models, it can be seen that there is some improvement in autocorrelation and seasonality compared to previously tested models. It is noteworthy that both models produced an adjusted R-squared value of around 0.947, meaning that they account for 94.7% of the variability in solar radiation. The F-test of both models gave a p-value < 0.05, thus both models are statistically significant at 5% level. The null hypothesis of normality of both models is rejected by their Shapiro-Wilk test results, which all have a p-value smaller than 0.05. They both suffer from multicollinearity as some VIF values are greater than 10. Thus, it can be concluded that the dynamic lag model is not ideal.

# Autoregressive distributed lag models

The autoregressive distributed lag model is implemented. The model with the lowest MASE value will be selected for fitting.

```
for (i in 1:3){
  for(j in 1:3){
    model_b = ardlDlm(x = as.vector(Solar_data$ppt), y = as.vector(Solar_data$solar),
p = i , q = j)
    cat("p =", i, "q =", j, "MASE =", MASE(model_b)$MASE, "\n")
  }
}
```

```
## p = 1 q = 1 MASE = 0.8392434
## p = 1 q = 2 MASE = 0.4971918
## p = 1 q = 3 MASE = 0.4740063
## p = 2 q = 1 MASE = 0.7834855
## p = 2 q = 2 MASE = 0.4951319
## p = 2 q = 3 MASE = 0.4738939
## p = 3 q = 1 MASE = 0.7572489
## p = 3 q = 2 MASE = 0.4955334
## p = 3 q = 3 MASE = 0.4737144
```

Model with p=3 and q=3 has the lowest MASE value. Thus, this is the model that will be fitted and analysed.

```
model_c <- ardlDlm(x=as.vector(Solar_data$ppt), y= as.vector(Solar_data$solar), p=3,
q=3)
summary(model_c)
```

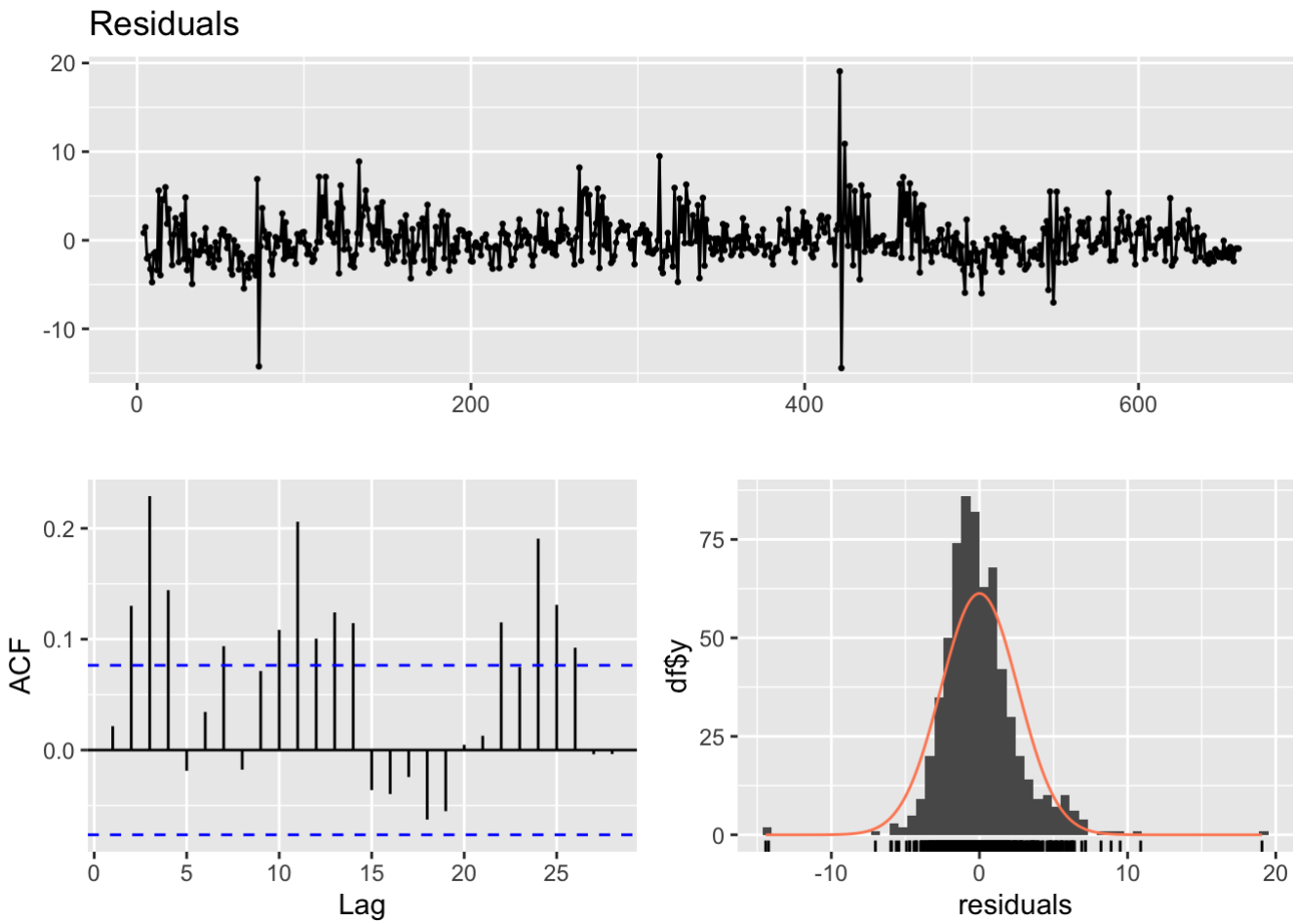
```
##
## Time series regression with "ts" data:
## Start = 4, End = 660
##
## Call:
## dynlm(formula = as.formula(model.text), data = data, start = 1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.4265  -1.5232  -0.2725   1.1582  19.0683
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.09000    0.39589   7.805 2.39e-14 ***
## X.t         -0.55917    0.56261  -0.994   0.3206
## X.1          1.00698    0.79376   1.269   0.2050
## X.2          1.84292    0.79195   2.327   0.0203 *
## X.3         -0.26711    0.56697  -0.471   0.6377
## Y.1          1.26560    0.03696  34.244  < 2e-16 ***
## Y.2         -0.13823    0.06139  -2.252   0.0247 *
## Y.3         -0.35408    0.03644  -9.715  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.619 on 649 degrees of freedom
## Multiple R-squared:  0.9295, Adjusted R-squared:  0.9287
## F-statistic:  1222 on 7 and 649 DF,  p-value: < 2.2e-16
```

```
vif(model_c$model)
```

```
##       X.t L(X.t, 1) L(X.t, 2) L(X.t, 3) L(y.t, 1) L(y.t, 2) L(y.t, 3)
##  3.777687  7.531419  7.514934  3.885577 12.548670 34.621642 12.222617
```

The R-squared value is 0.9287, this means that the model is accountable for 92.87% of the variability in solar radiation, which is relatively high. The F-test gave a p-value < 0.05, thus the model is statistically significant at 5% level. The model suffers from multicollinearity as some VIF values are greater than 10.

```
checkresiduals(model_c$model)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 11
##
## data:  Residuals
## LM test = 131.77, df = 11, p-value < 2.2e-16
```

```
shapiro.test(residuals(model_c$model))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(model_c$model)
## W = 0.91456, p-value < 2.2e-16
```

The p-value output for the Breusch-Godfrey test is smaller than 0.05 indicates that there is serial correlation in the residuals. The null hypothesis of normality is rejected by the Shapiro-Wilk test, which also has a p-value smaller than 0.05. The residuals are not normally distributed. Again, the model is not ideal.

Overall, none of the tested time series regression models is ideal. A dataframe is created to store the MASE, AIC, and BIC values of different models. The dataframe is designed so that the accuracy values of future models will be added. First, we have to check the classes of the models.

```
models_list <- list(fin_dlm = fin_dlm$model, poly_dl = poly_dl$model, K_model = K_mod
el$model, model_c = model_c$model)
model_classes <- sapply(models_list, class)
print(model_classes)
```

```
## $fin_dlm
## [1] "lm"
##
## $poly_dl
## [1] "lm"
##
## $K_model
## [1] "ivreg"
##
## $model_c
## [1] "dynlm" "lm"
```

From this, we can see that we have to transform “K_model” to “lm” class. Thus, we have the following codes to create the dataframe “accuracy_table”:

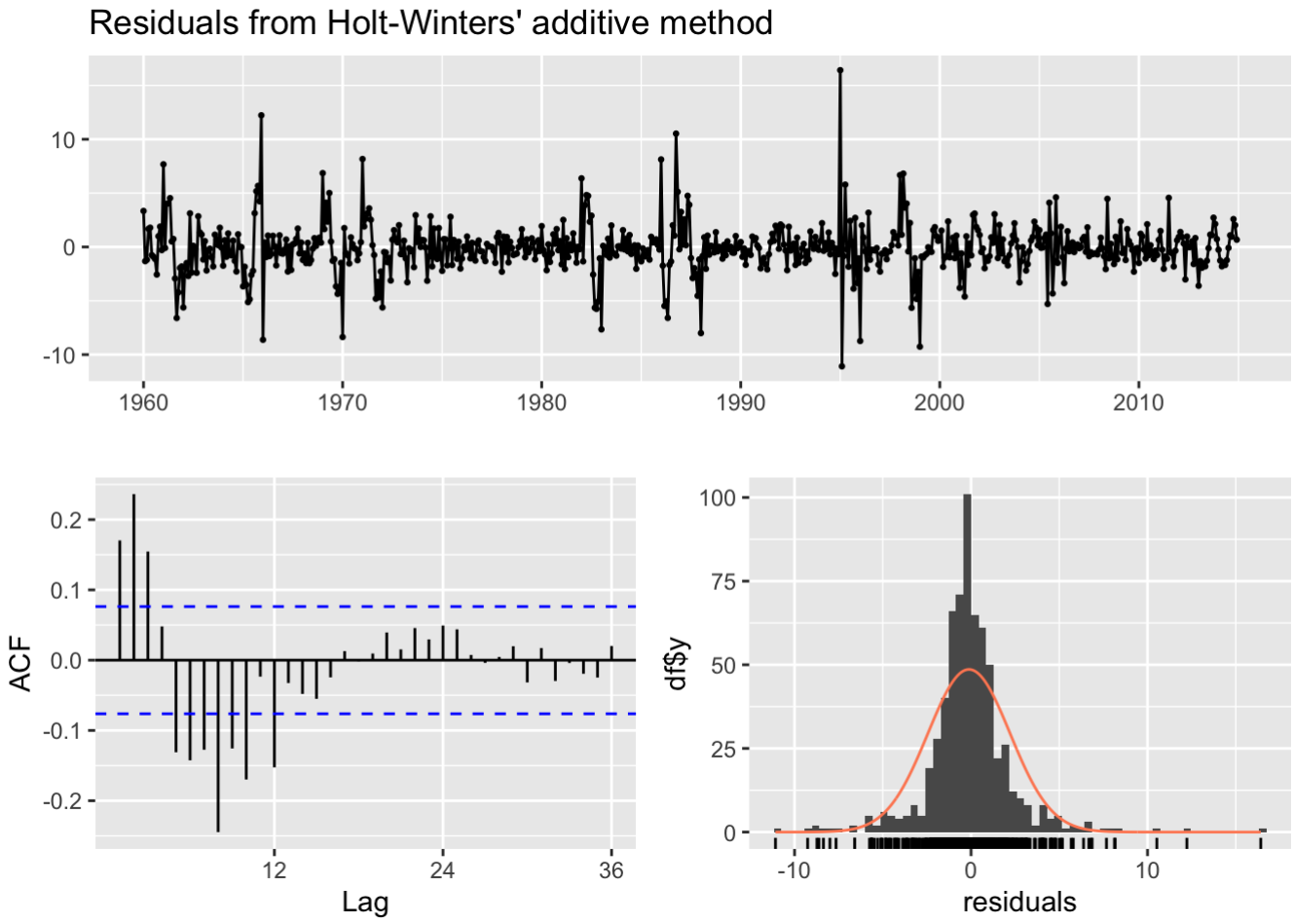
```
model_c <- ardlDlm(x=as.vector(Solar_data$ppt), y= as.vector(Solar_data$solar), p=3,
q=3)
models <- c("fin_dlm", "poly_dl", "K_model", "model_c")
attr(K_model$model, "class") = "lm"
aic <- AIC(fin_dlm$model, poly_dl$model, K_model$model, model_c$model)$AIC
bic <- BIC(fin_dlm$model, poly_dl$model, K_model$model, model_c$model)$BIC
MASE <- MASE(fin_dlm, poly_dl, K_model, model_c)$MASE
accuracy_table <- data.frame(models, MASE, aic, bic)
colnames(accuracy_table) <- c("Model", "MASE", "AIC", "BIC")
head(accuracy_table)
```

```
##     Model      MASE      AIC      BIC
## 1 fin_dlm 1.5516004 4578.787 4645.895
## 2 poly_dl 1.5630351 4567.969 4590.339
## 3 K_model 1.0324829 3946.476 3964.439
## 4 model_c 0.4737144 3139.409 3179.798
```

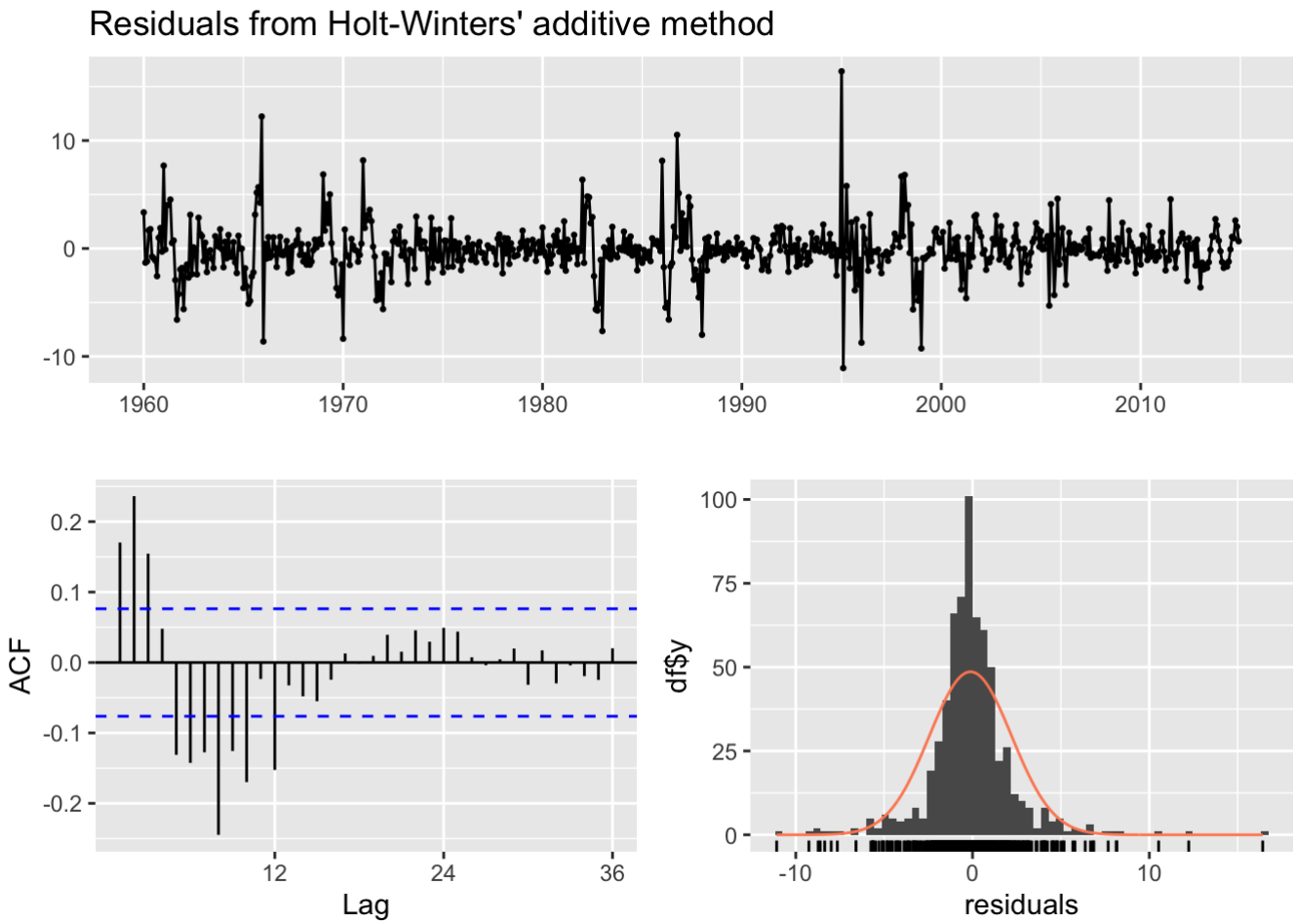
# Exponential smoothing

The next method that will be tested is exponential smoothing. There are 6 models that include additive or multiplicative seasonality that need to be considered.

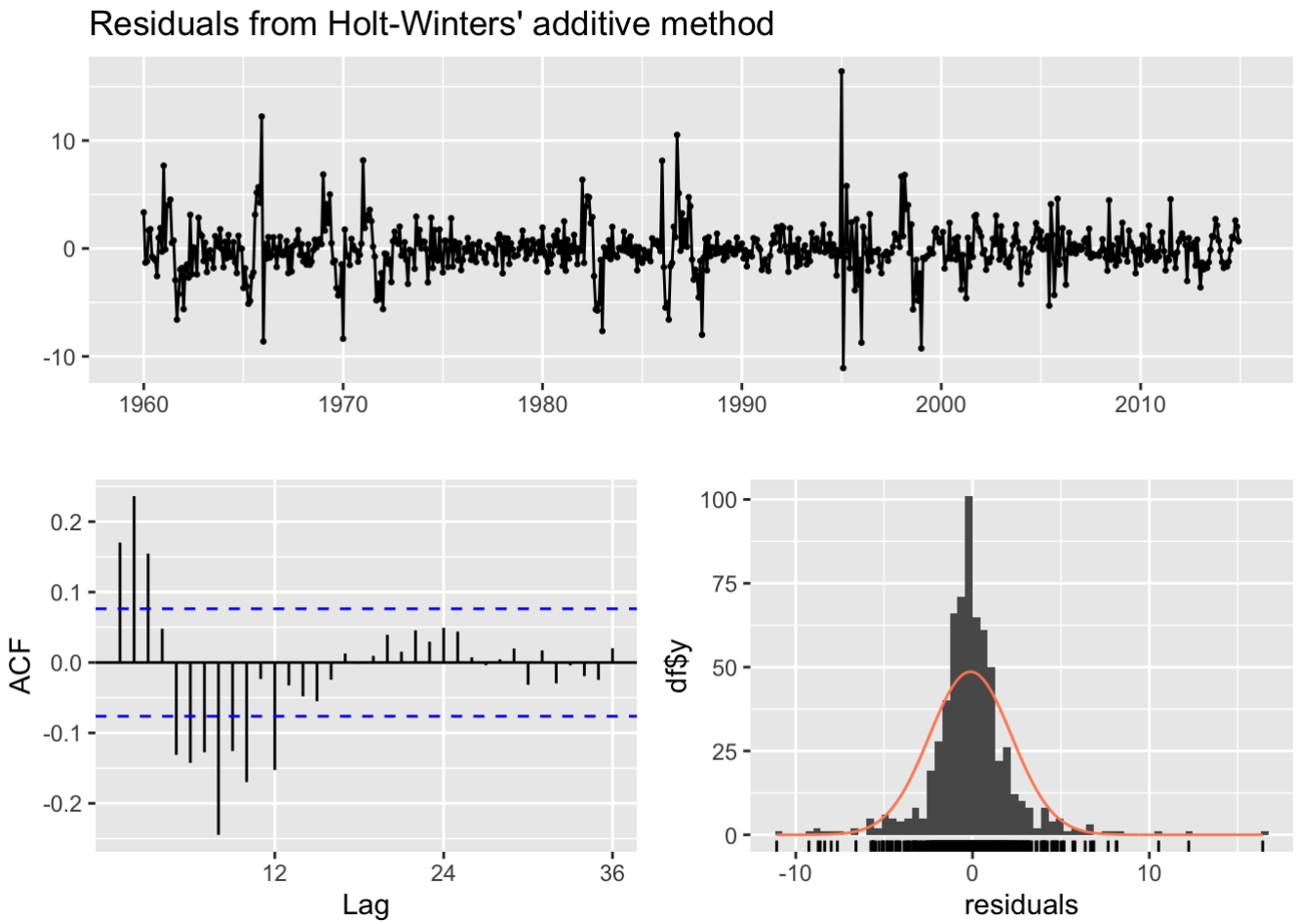
```
exponential = c(T,F)
seasonality <- c("additive","multiplicative")
damped <- c(T,F)
expa <- expand.grid(exponential, seasonality, damped)
expa <- expa[-c(3,4),]
f_aic <- array(NA, 6)
f_bic <- array(NA, 6)
f_mase <- array(NA, 6)
levels <- array(NA, dim=c(6,3))
for (i in 1:6){
  holt_winters <- hw(solar, ES = expa[i,1], seasonal = toString(expa[i,2], damped = e
xpa[i,3]))
  f_aic[i] <- holt_winters$model$aic
  f_bic[i] <- holt_winters$model$bic
  f_mase[i] <- accuracy(holt_winters)[6]
  levels[i,1] <- expa[i,1]
  levels[i,2] <- toString(expa[i,2])
  levels[i,3] <- expa[i,3]
  checkresiduals(holt_winters)
}
```



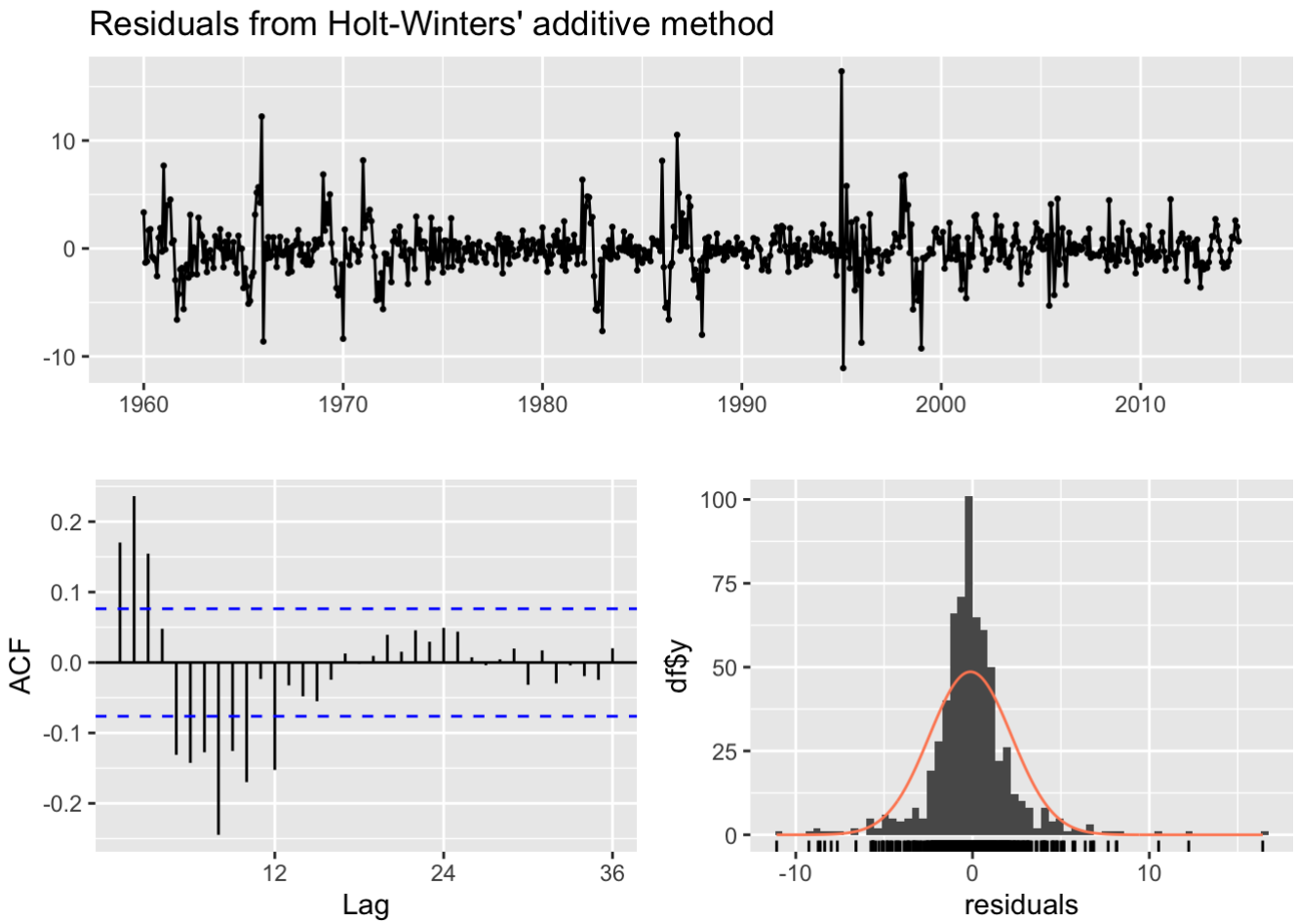

```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' additive method
## Q* = 205.55, df = 24, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 24
```



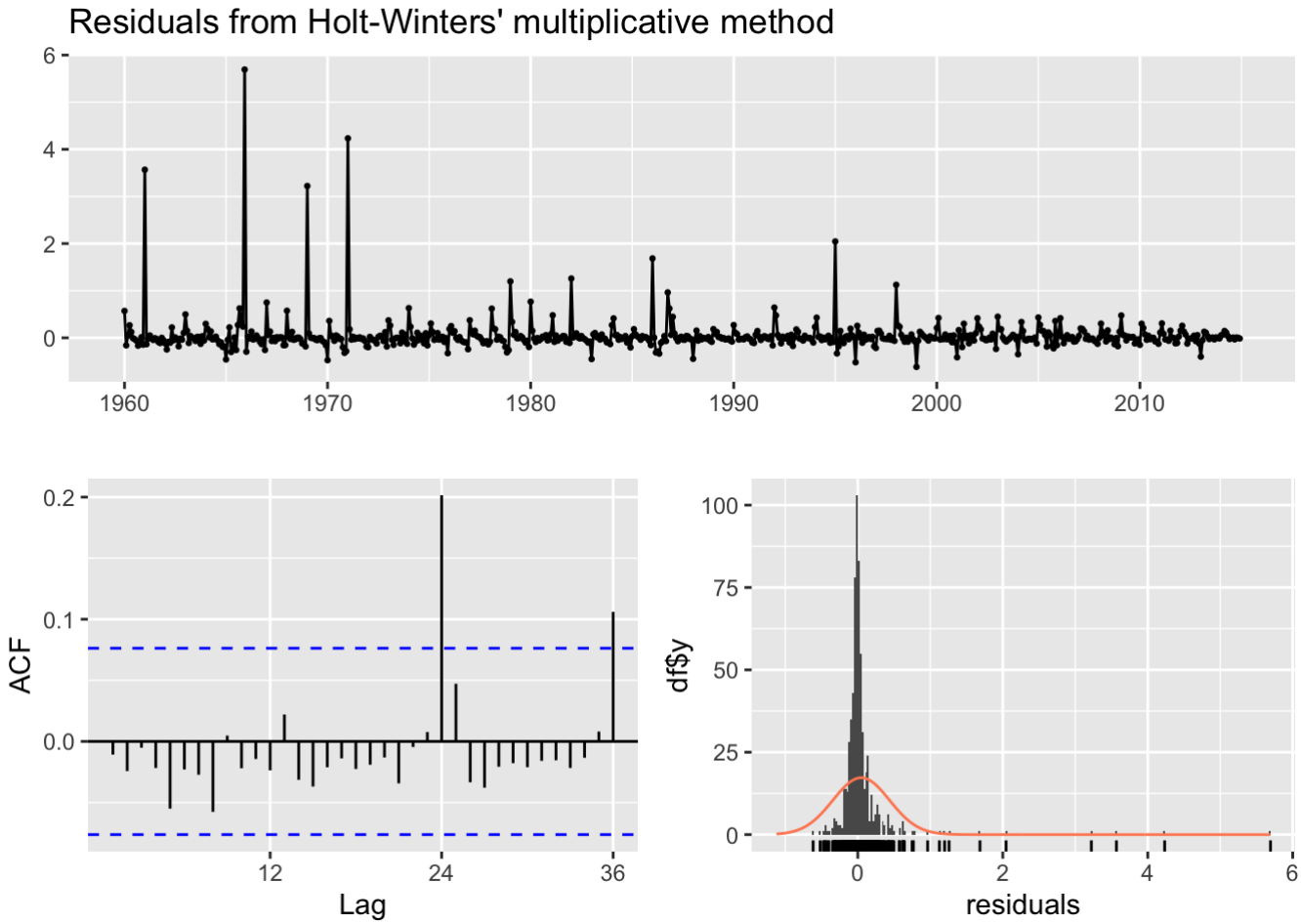

```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' additive method
## Q* = 205.55, df = 24, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 24
```



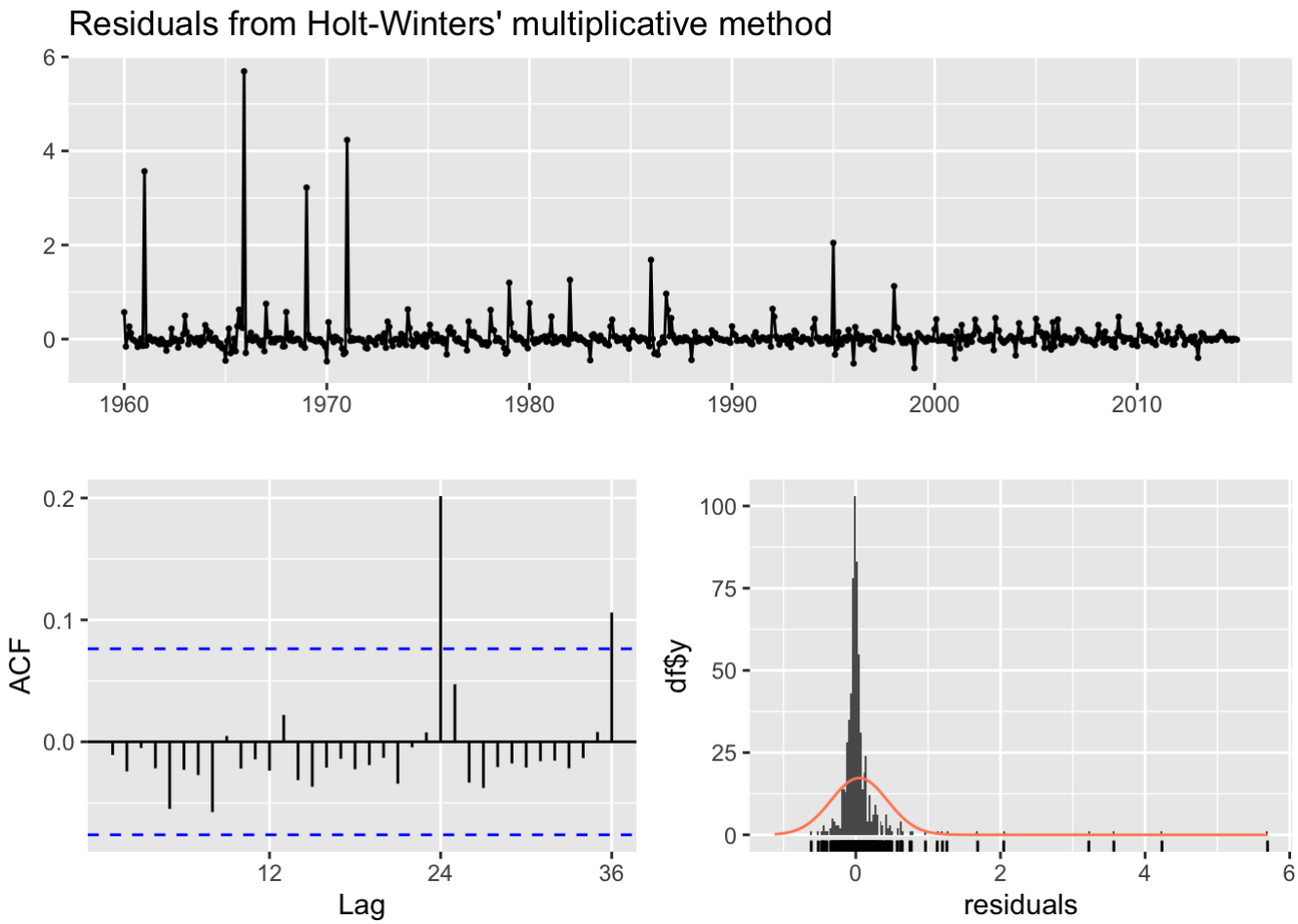

```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' additive method
## Q* = 205.55, df = 24, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 24
```




```
##
##  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
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' multiplicative method
## Q* = 38.585, df = 24, p-value = 0.03017
##
## Model df: 0.   Total lags used: 24
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' multiplicative method
## Q* = 38.585, df = 24, p-value = 0.03017
##
## Model df: 0.   Total lags used: 24
```

The exponential smoothing method still has autocorrelation judging from the p-values of the Ljung-Box tests as they are all less than 0.05. However, when looking at the residual plot of the Holt-Winter's multiplicative method, there is definitely a clear improvement in autocorrelation and seasonality compared to previously tested models.

The new accuracy values of the exponential smoothing models is added to the "accuracy_table" data frame. The models can be identified in this format: trend (multiplicative or additive), seasonality (multiplicative or additive) and if the trend is damped (damped) or not damped (ND).

```
newvalues <- data.frame(levels, f_mase, f_aic, f_bic)
colnames(newvalues) <- c("Trend", "Seasonality", "damped", "MASE", "AIC", "BIC")
newvalues$Trend <- factor(newvalues$Trend, levels = c(T,F), labels = c("multiplicativ
e","additive"))
newvalues$damped <- factor(newvalues$damped, levels = c(T,F), labels = c("damped","N
D"))

newvalues <- unite(newvalues, col = "Model", c("Trend","Seasonality","damped"))
accuracy_table <- rbind(accuracy_table, newvalues)
accuracy_table
```

```
##                                Model      MASE      AIC      BIC
## 1                            fin_dlm 1.5516004 4578.787 4645.895
## 2                            poly_dl 1.5630351 4567.969 4590.339
## 3                            K_model 1.0324829 3946.476 3964.439
## 4                            model_c 0.4737144 3139.409 3179.798
## 5    multiplicative_additive_damped 0.2471600 5434.708 5511.076
## 6          additive_additive_damped 0.2471600 5434.708 5511.076
## 7        multiplicative_additive_ND 0.2471600 5434.708 5511.076
## 8              additive_additive_ND 0.2471600 5434.708 5511.076
## 9  multiplicative_multiplicative_ND 0.2233077 6648.746 6725.114
## 10       additive_multiplicative_ND 0.2233077 6648.746 6725.114
```

# State-space models

Excluding the models that are prohibited in R due to stability issues, there are 8 state-space models that can be implemented.

```
vlist <- c("AAA", "MAA", "MAM", "MMM")
damp <- c(T,F)
ets_models <- expand.grid(vlist, damp)
ets_aic <- array(NA, 8)
ets_mase <- array(NA,8)
ets_bic <- array(NA,8)
mod <- array(NA, dim=c(8,2))
for (i in 1:8){
  ets <- ets(solar, model = toString(ets_models[i, 1]), damped = ets_models[i,2])
  ets_aic[i] <- ets$aic
  ets_bic[i] <- ets$bic
  ets_mase[i] <- accuracy(ets)[6]
  mod[i,1] <- toString(ets_models[i,1])
  mod[i,2] <- ets_models[i,2]
}
```

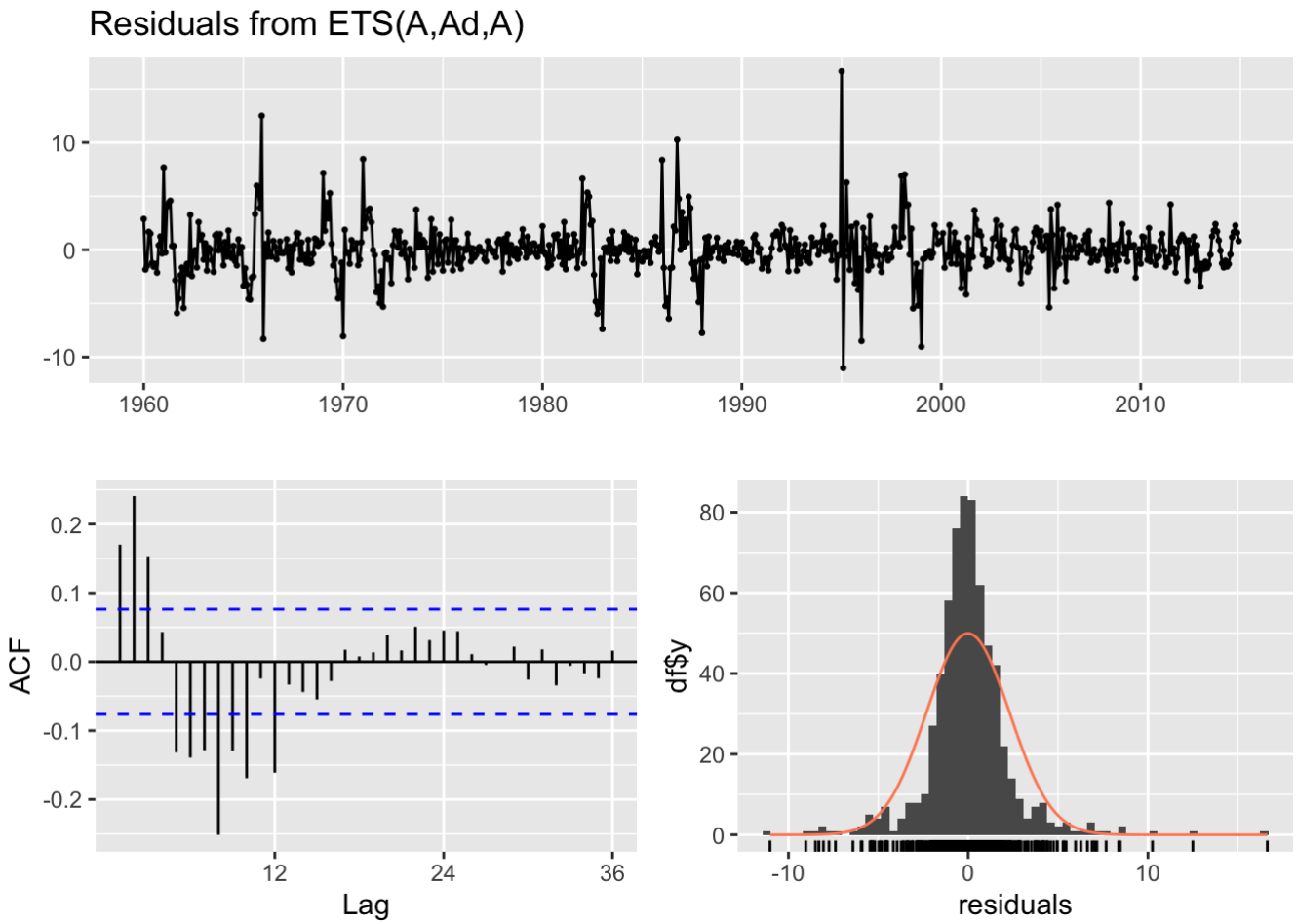
Next, we use the “auto-fit” function to find the ideal model according to the software.

```
auto_ets_solar <- ets(solar)
summary(auto_ets_solar)
```

```
## ETS(A,Ad,A)
##
## Call:
##  ets(y = solar)
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 1e-04
##     gamma = 1e-04
##     phi   = 0.9388
##
##   Initial states:
##     l = 11.154
##     b = 0.7632
##     s = -10.4919 -8.137 -3.348 2.5794 8.08 11.1219
##            9.9586 6.9916 1.9573 -1.8565 -7.1607 -9.6946
##
##   sigma:  2.3446
##
##      AIC     AICc      BIC
## 5428.422 5429.489 5509.282
##
## Training set error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.01091357 2.314163 1.498521 -1.468083 12.44796 0.2461797
##                    ACF1
## Training set 0.1700724
```

We then check the residuals of this model

```
checkresiduals(auto_ets_solar)
```




```
##
##  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
```

```
shapiro.test(residuals(auto_ets_solar))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(auto_ets_solar)
## W = 0.88756, p-value < 2.2e-16
```

Both the p-value output of the Ljung-Box test and the Shapiro-Wilk normality test are less than 0.05. Thus, we can see that the model has autocorrelation and is not normally distributed. Therefore, it is not ideal. However, it is noteworthy that the model's MASE value is quite low at 0.2461797 so it is worth considering as a viable forecasting model. The model's accuracy values are added into the "accuracy_table" data frame.

```
calculate <- data.frame(mod, ets_mase, ets_aic, ets_bic)
calculate$X2 <- factor(calculate$X2, levels = c(T,F), labels = c("Damped","ND"))
calculate <- unite(calculate, "Model", c("X1","X2"))
colnames(calculate) <- c("Model", "MASE", "AIC", "BIC")
accuracy_table <- rbind(accuracy_table,calculate)

accuracy_table <- arrange(accuracy_table, MASE)
kable(accuracy_table, caption = "Different forecasting models sorted by MASE (Ascendi
ng)")
```

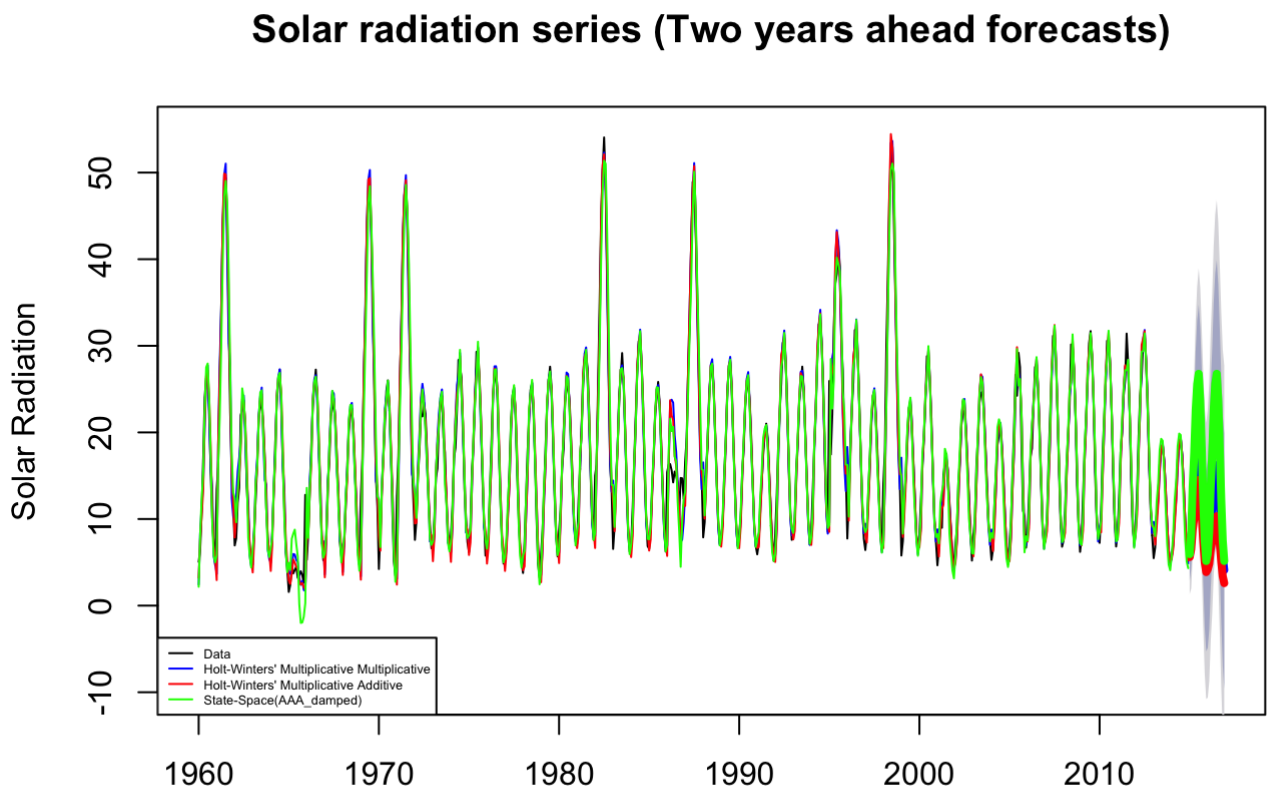
Different forecasting models sorted by MASE (Ascending)

| Model | MASE | AIC | BIC |
|---|---|---|---|
| multiplicative_multiplicative_ND | 0.2233077 | 6648.746 | 6725.114 |
| additive_multiplicative_ND | 0.2233077 | 6648.746 | 6725.114 |
| AAA_Damped | 0.2461797 | 5428.422 | 5509.282 |
| multiplicative_additive_damped | 0.2471600 | 5434.708 | 5511.076 |
| additive_additive_damped | 0.2471600 | 5434.708 | 5511.076 |
| multiplicative_additive_ND | 0.2471600 | 5434.708 | 5511.076 |
| additive_additive_ND | 0.2471600 | 5434.708 | 5511.076 |
| AAA_ND | 0.2471600 | 5434.708 | 5511.076 |
| MMM_Damped | 0.3201193 | 5995.550 | 6076.410 |
| MAM_Damped | 0.3222574 | 5953.502 | 6034.363 |
| MAM_ND | 0.3721664 | 6105.959 | 6182.327 |
| MAA_Damped | 0.3798095 | 6469.079 | 6549.940 |
| model_c | 0.4737144 | 3139.409 | 3179.798 |
| MAA_ND | 0.4748561 | 7602.755 | 7679.123 |
| MMM_ND | 0.5292151 | 6670.168 | 6746.536 |
| K_model | 1.0324829 | 3946.476 | 3964.439 |
| fin_dlm | 1.5516004 | 4578.787 | 4645.895 |
| poly_dl | 1.5630351 | 4567.969 | 4590.339 |

# Forecasting

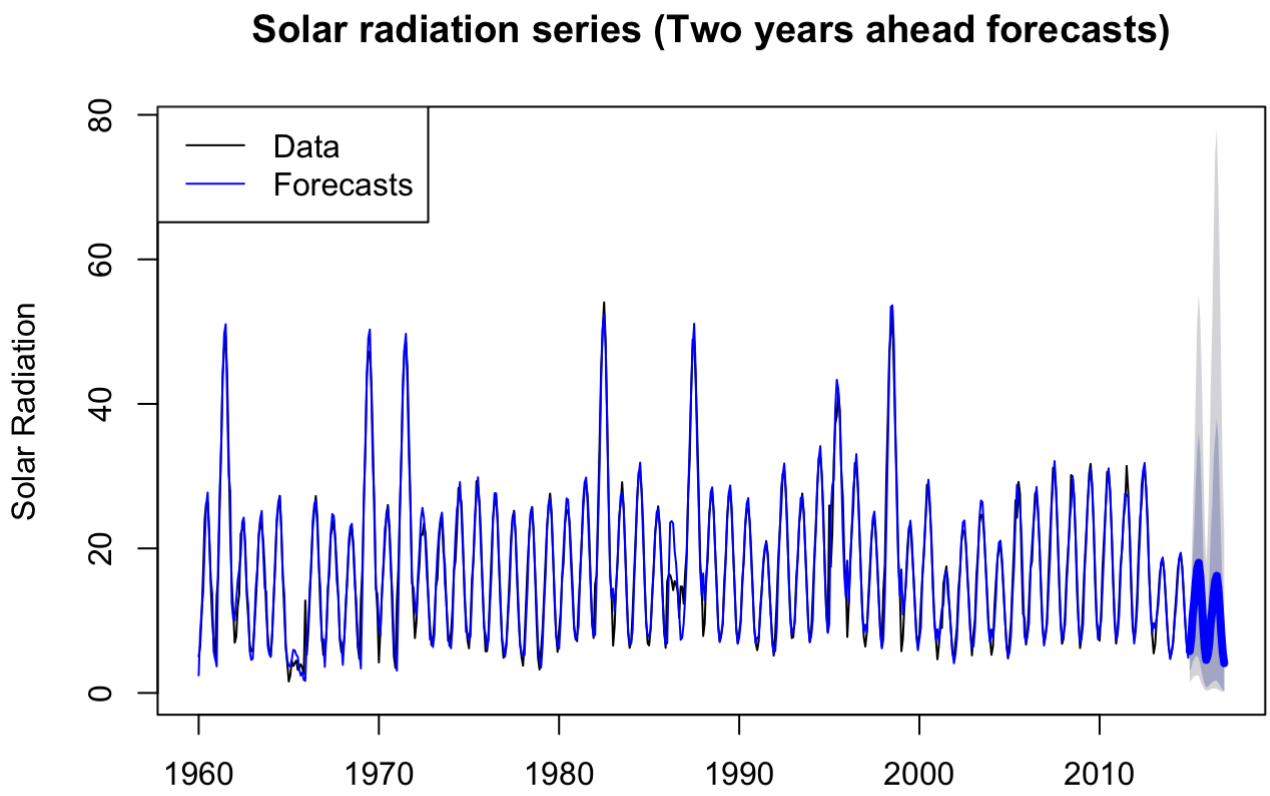
The three models with the lowest MASE are selected: Holt-Winters multiplicative method with multiplicative trend, Holt-Winters multiplicative method with additive trend, and the State-Space (AAA_damped) method. The three models are fitted to compare which one is the most optimal for forecast with two years ahead.

```
fit_mod1 <- hw(solar, seasonal = "multiplicative", exponential = T, h = 2*frequency(s
olar))
fit_mod2 <- hw(solar, seasonal = "multiplicative", h = 2*frequency(solar))
fit_mod3 <- ets(solar,model="AAA", damped=T)
for_fit_mod3 <- forecast.ets(fit_mod3)
plot(for_fit_mod3, fcol = "white", main = "Solar radiation series (Two years ahead fo
recasts)", ylab = "Solar Radiation", ylim = c(-10,55))
lines(fitted(fit_mod1), col = "blue")
lines(fit_mod1$mean, col = "blue", lwd = 4)
lines(fitted(fit_mod2), col = "red")
lines(fit_mod2$mean, col = "red", lwd = 4)
lines(fitted(fit_mod3), col = "green")
lines(for_fit_mod3$mean, col = "green", lwd = 4)
legend("bottomleft", lty = 1, col = c("black", "blue", "red", "green"),
       c("Data", "Holt-Winters' Multiplicative Multiplicative",
         "Holt-Winters' Multiplicative Additive", "State-Space(AAA_damped)"), cex =
0.4)
```




From the plot, we can see that the State-Space method is the most far off, most evidently around 1967, where the State-Space line is obviously much further from the data line compared to the two Holt-Winters method lines. Between these two methods, we can see that the Holt-Winters multiplicative method with multiplicative trend is closer to the actual data. Thus, it will be used for the final forecast.

```
plot(fit_mod1, fcol = "white", main = "Solar radiation series (Two years ahead foreca
sts)", ylab = "Solar Radiation")
lines(fitted(fit_mod1), col = "blue")
lines(fit_mod1$mean, col = "blue", lwd = 4)
legend("topleft", lty = 1, col = c("black", "blue"), c("Data", "Forecasts"))
```




Finally, we have the 2 years ahead forecast values of the amount of horizontal solar radiation reaching the ground at a particular location over the globe:

```
forc <- fit_mod1$mean
ub <- fit_mod1$upper[,2]
lb <- fit_mod1$lower[,2]
forecasts <- ts.intersect(ts(lb, start = c(2015,1), frequency = 12), ts(forc,start =
c(2015,1), frequency = 12), ts(ub,start = c(2015,1), frequency = 12))
colnames(forecasts) <- c("Lower bound", "Point forecast", "Upper bound")
forecasts
```

```
##          Lower bound Point forecast Upper bound
## Jan 2015  1.52724132       5.797870    10.16838
## Feb 2015  1.75307471       7.513359    15.16638
## Mar 2015  2.08481450      10.758212    23.90197
## Apr 2015  2.34768901      12.694087    31.52663
## May 2015  2.37317391      15.340074    41.22056
## Jun 2015  2.44997758      17.239431    51.48587
## Jul 2015  2.36947876      18.004700    55.15890
## Aug 2015  1.78964717      16.206596    51.42415
## Sep 2015  1.24323107      13.011256    44.09811
## Oct 2015  0.84943058       9.215869    32.60982
## Nov 2015  0.50677539       6.214762    22.82218
## Dec 2015  0.35213049       4.565667    17.82204
## Jan 2016  0.32862477       5.220661    21.93197
## Feb 2016  0.38816407       6.765364    29.43951
## Mar 2016  0.53375998       9.687175    42.12141
## Apr 2016  0.61918627      11.430324    50.70885
## May 2016  0.61449007      13.812889    63.79488
## Jun 2016  0.60592037      15.523155    74.94737
## Jul 2016  0.58512210      16.212237    78.00131
## Aug 2016  0.45722444      14.593143    69.80833
## Sep 2016  0.35580941      11.715916    59.35764
## Oct 2016  0.23275843       8.298381    42.84528
## Nov 2016  0.15692105       5.596049    29.53715
## Dec 2016  0.09786152       4.111130    21.80095
```

# Task 2

# Data description

The first step of task 2 is to load the dataset containing the information about Property Price Index (PPI) and population change.

```
price_data <- read.csv("/Users/macbookair/Desktop/data2.csv")
head(price_data)
```

```
##    Quarter price change
## 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
```

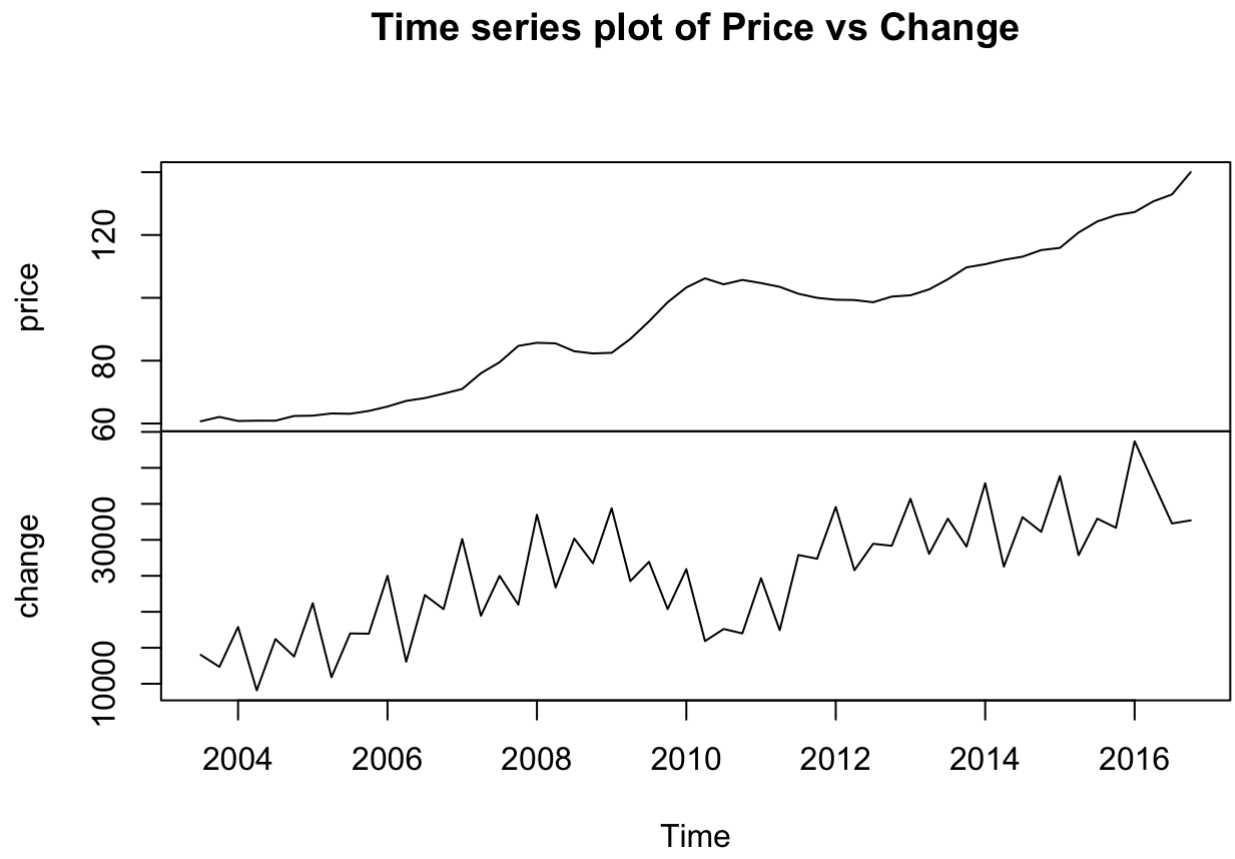
After that, time series of the property price data and the population change data are created.

```
prop_change <- ts(price_data[,2:3], start = c(2003,3), frequency = 4)
prop <- ts(price_data$price, start = c(2003,3), frequency = 4)
change <- ts(price_data$change, start = c(2003,3), frequency = 4)
head(prop_change)
```

```
##         price change
## 2003 Q3  60.7  14017
## 2003 Q4  62.1  12350
## 2004 Q1  60.8  17894
## 2004 Q2  60.9   9079
## 2004 Q3  60.9  16210
## 2004 Q4  62.4  13788
```

# Data exploration, visualisation, and investigation

```
plot(prop_change, main="Time series plot of Price vs Change")
```



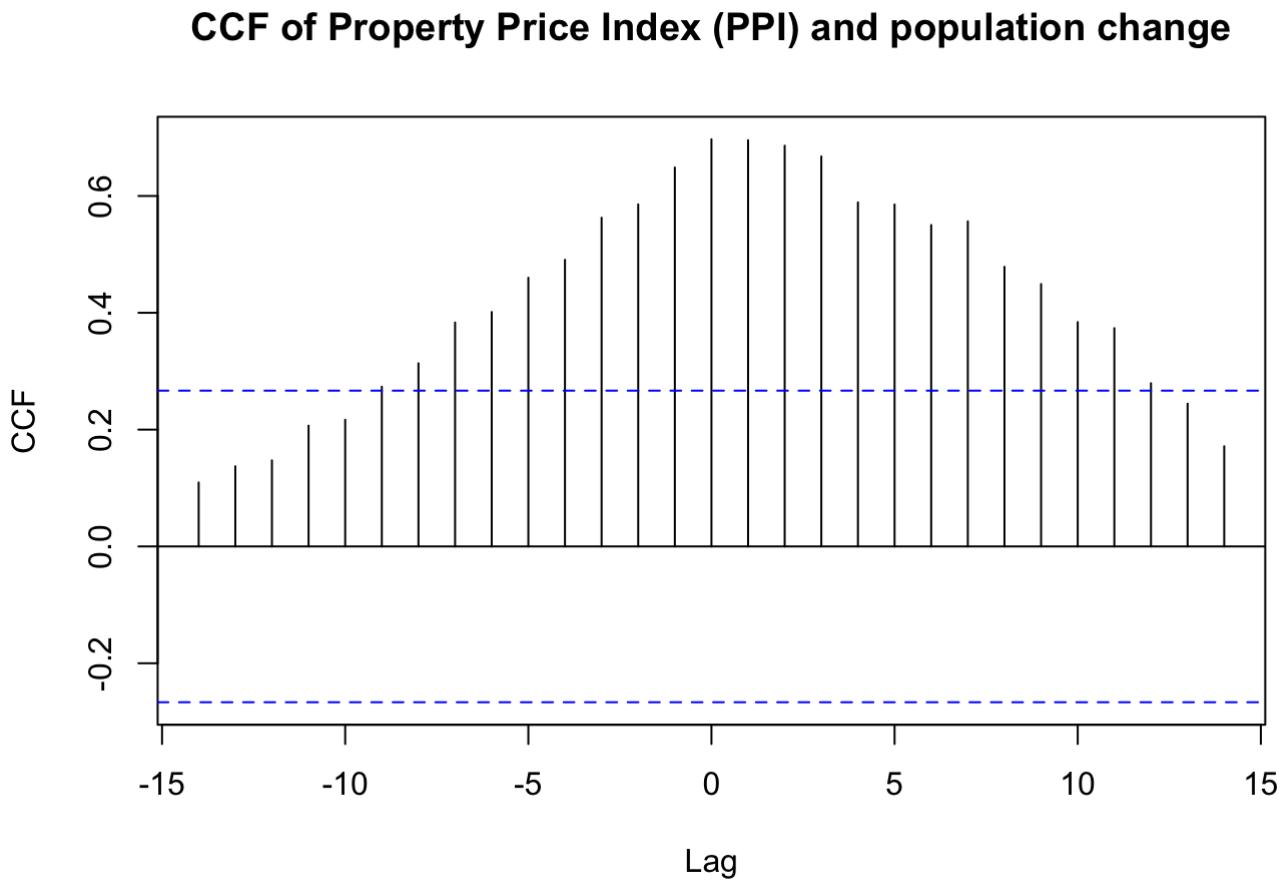
From the plot, we can see that both series have an increasing trend. The visualisation implies that there is some correlation between the property price index and the population change series. In order to investigate this further, the correlation coefficient is calculated.

```
cor(prop,change)
```

```
## [1] 0.6970439
```

The correlation coefficient of 0.6970439 suggests that there is a positive correlation between the two series. Next, a CCF plot of the two series is constructed.

```
ccf(as.vector(prop), as.vector(change), ylab = "CCF", main = "CCF of Property Price I
ndex (PPI) and population change")
```




From the CCF plot, we can see that there is a high chance of cross correlation between Property Price Index (PPI) and population change. From the time series plot, the correlation coefficient, and the CCF plot, it is very likely that correlation exists between the two series. However, to be certain, we need to conduct adf tests on both series to check their stationarity.

```
adf.test(prop)
```

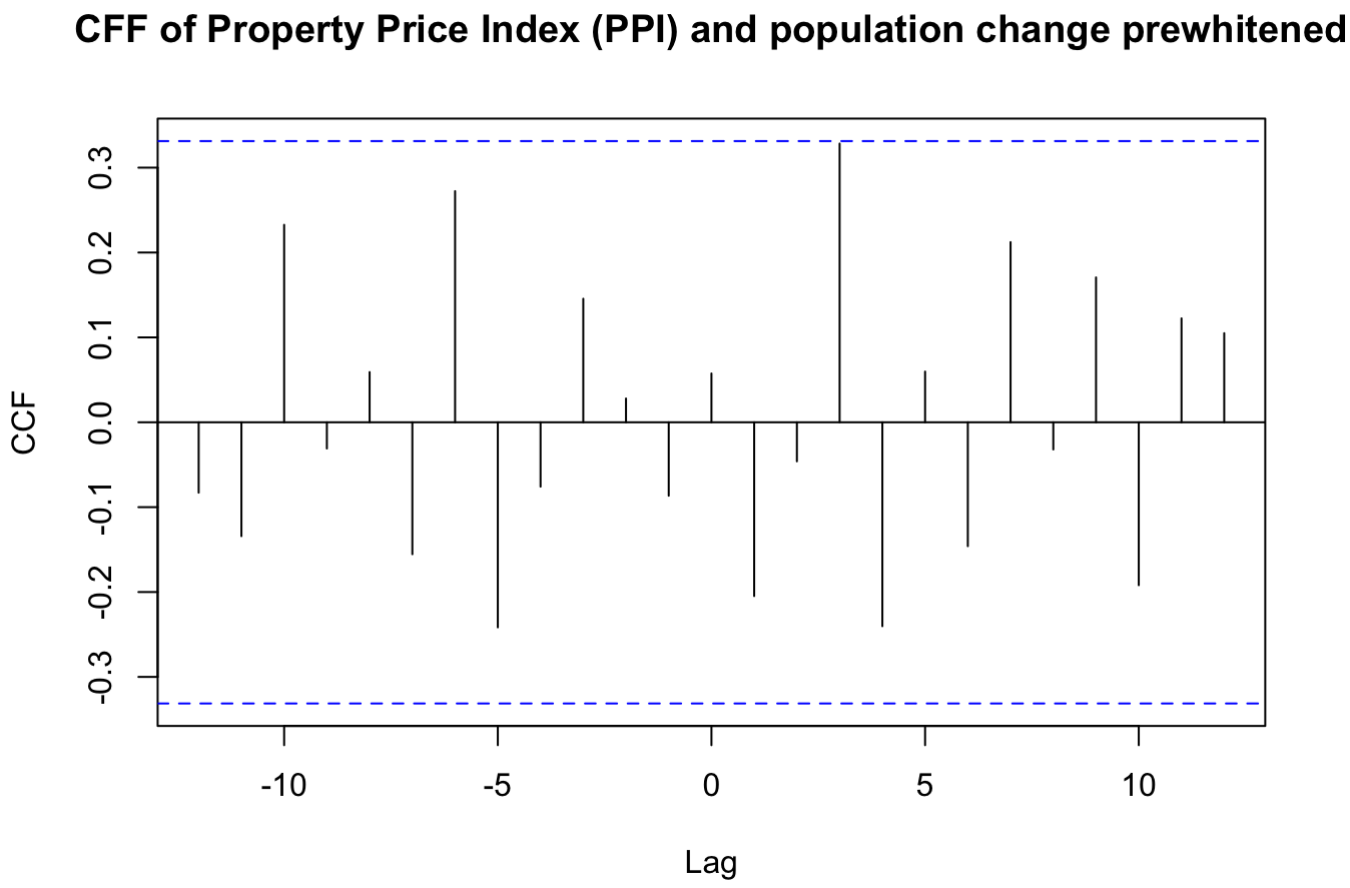
```
##
##  Augmented Dickey-Fuller Test
##
## data:  prop
## Dickey-Fuller = -1.3264, Lag order = 3, p-value = 0.8458
## alternative hypothesis: stationary
```

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

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

Both ADF tests produced p-value greater than 0.05, thus, both series are nonstationary. This increases the chance that the signs of correlation between the two series that were shown in previous tests might be spurious. In order to be definitively certain, we will apply prewhitening and plot a CCF of the two series after making sure they are stationary.

```
diff <- ts.intersect(diff(diff(prop,4)), diff(diff(change,4)))
prewhiten(as.vector(diff[,1]), as.vector(diff[,2]), ylab='CCF', main = "CFF of Proper
ty Price Index (PPI) and population change prewhitened")
```



From the prewhitened CCF plot, we can see that there is no significant correlation between Property Price Index (PPI) and population change. Thus, it is concluded that the correlation between the two series is spurious.

# Conclusion

Task 1: All the models that were tested in this task came with their own set of problems. The final three models chosen were Holt-Winters multiplicative method with multiplicative trend, Holt-Winters multiplicative method with additive trend, and the State-Space (AAA_damped) method based on their MASE values. However, all of them suffer from normality issues which will affect any statistical analysis that assume normality. It was found that the Holt-Winters multiplicative method with multiplicative trend was the most successful model, thus, it was utilized for the final forecast. However, the results have very wide bounds, thus, it can be seen as unreliable.

Task 2: The Property Price Index (PPI) series and the Population Change series seemed like they were very likely to be cross-correlated through the first few tests including a time series plot, calculating their correlation coefficient, and a CCF plot. However, after testing their nonstationarity, it was found that there is a high chance that these tests can be misleading due to the series' nonstationarity. Thus, after implementing the prewhitening method, it was found that the correlation between the two series was indeed spurious.