Forecasting Assignment 2

Namita Chhibba(s3631442)

25 September 2017

Opening Libraries

```
# Libraries

library(dynlm)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(ggplot2)
library(AER)
```

```
## Loading required package: car
```

```
## Loading required package: lmtest
```

```
## Loading required package: sandwich
```

```
## Loading required package: survival
```

```
#install.packages("swirl", repos="http://cran.rstudio.com/", dependencies=TRUE)
library(Hmisc)
```

```
## Loading required package: lattice
```

```
## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units
```

```
library(forecast)
library(dLagM)
```

```
## Loading required package: wavethresh
```

```
## Loading required package: MASS
```

```
## WaveThresh: R wavelet software, release 4.6.8, installed
```

```
## Copyright Guy Nason and others 1993-2016
```

```
## Note: nlevels has been renamed to nlevelsWT
```

```
library(expsmooth)
library(TSA)
```

```
## Loading required package: leaps
```

```
## Loading required package: locfit
```

```
## locfit 1.5-9.1    2013-03-22
```

```
## Loading required package: mgcv
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:forecast':
##
##     getResponse
```

```
## This is mgcv 1.8-17. For overview type 'help("mgcv-package")'.
```

```
## Loading required package: tseries
```

```
##
## Attaching package: 'TSA'
```

```
## The following objects are masked from 'package:stats':
##
##     acf, arima
```

```
## The following object is masked from 'package:utils':
##
##     tar
```

```
library(x12)
```

```
## Loading required package: x13binary
```

```
## x12 is ready to use.
```

```
## Use the package x12GUI for a Graphical User Interface.
```

```
## By default the X13-ARIMA-SEATS binaries provided by the R package x13binary
```

```
## are used but this can be changed with x12path(validpath)
```

```
## ---------------
```

```
## Suggestions and bug-reports can be submitted at: https://github.com/statisti
kat/x12/issues
```

```
library(car)
library(ltsa)
library(readr)
```

```
##
## Attaching package: 'readr'
```

```
## The following object is masked from 'package:TSA':
##
##     spec
```

```
library(bestglm)
library(FitAR)
```

```
##
## Attaching package: 'FitAR'
```

```
## The following object is masked from 'package:forecast':
##
##     BoxCox
```

```
## The following object is masked from 'package:car':
##
##     Boot
```

Question 1

Importing and Reading dataset

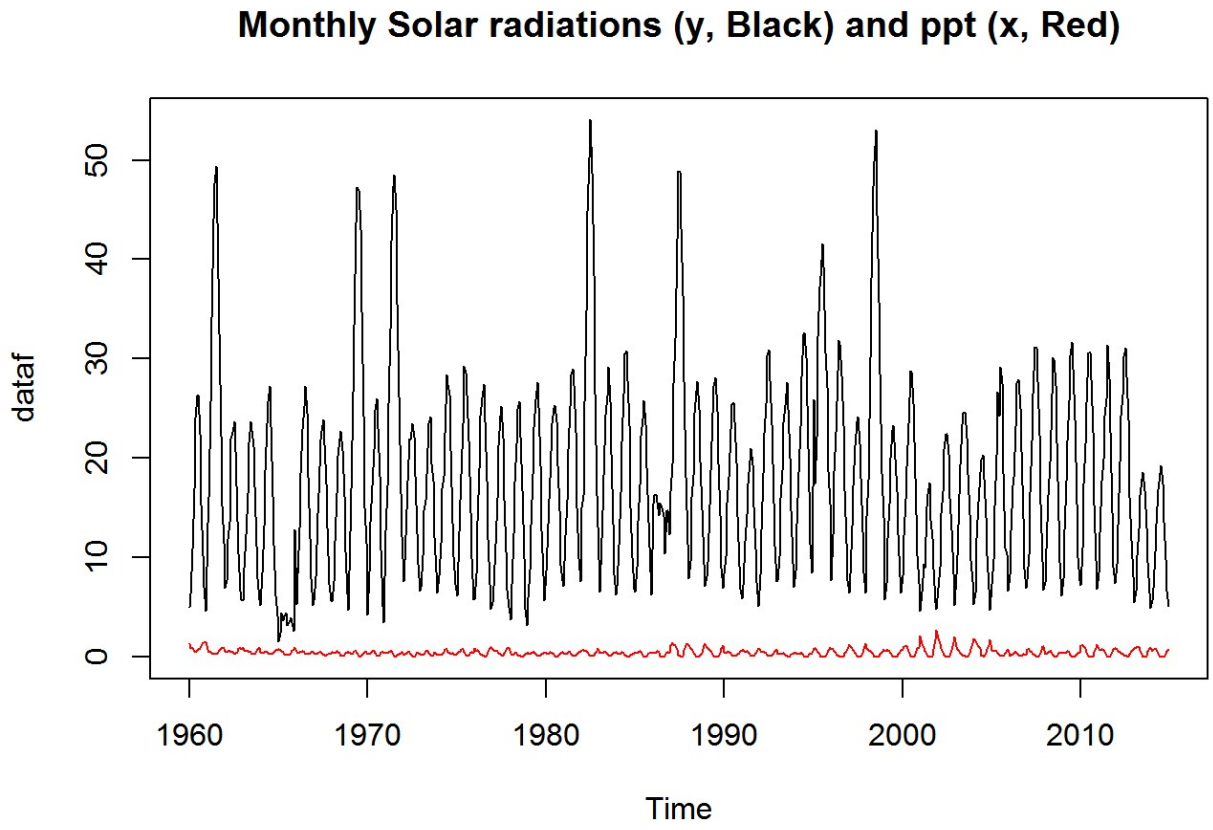
Convert to time series

```
data1 <- read.csv("C:/Users/user/Downloads/data1(5).csv")
#data1
yseries <- ts(data1$solar, start=c(1960,1),frequency = 12)
xseries <- ts(data1$ppt, start=c(1960,1), frequency=12)
dataf=ts(data1[,1:2], start=c(1960,1), frequency=12)
head(data)
```

```
##
## 1 function (..., list = character(), package = NULL, lib.loc = NULL,
## 2     verbose = getOption("verbose"), envir = .GlobalEnv)
## 3 {
## 4     fileExt <- function(x) {
## 5         db <- grepl("\\\\.[^.]+\\\\.(gz|bz2|xz)$", x)
## 6         ans <- sub(".*\\\\.", "", x)
```

Plotting the time series

```
plot(dataf, plot.type="s",col = c("black", "red"),
     main = "Monthly Solar radiations (y, Black) and ppt (x, Red)")
```



Interpretation # There seems to be presence of seasonality but no apparent presence of trend. # Variance of solar radiation is high and that of precipitation is low.

Finding the correlation between precipitation and solar radiations

```
cor(xseries,yseries)
```

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

Interpretation

There is a negative correlation between horizontal solar radiation and precipitation.

Distributed Lag Models

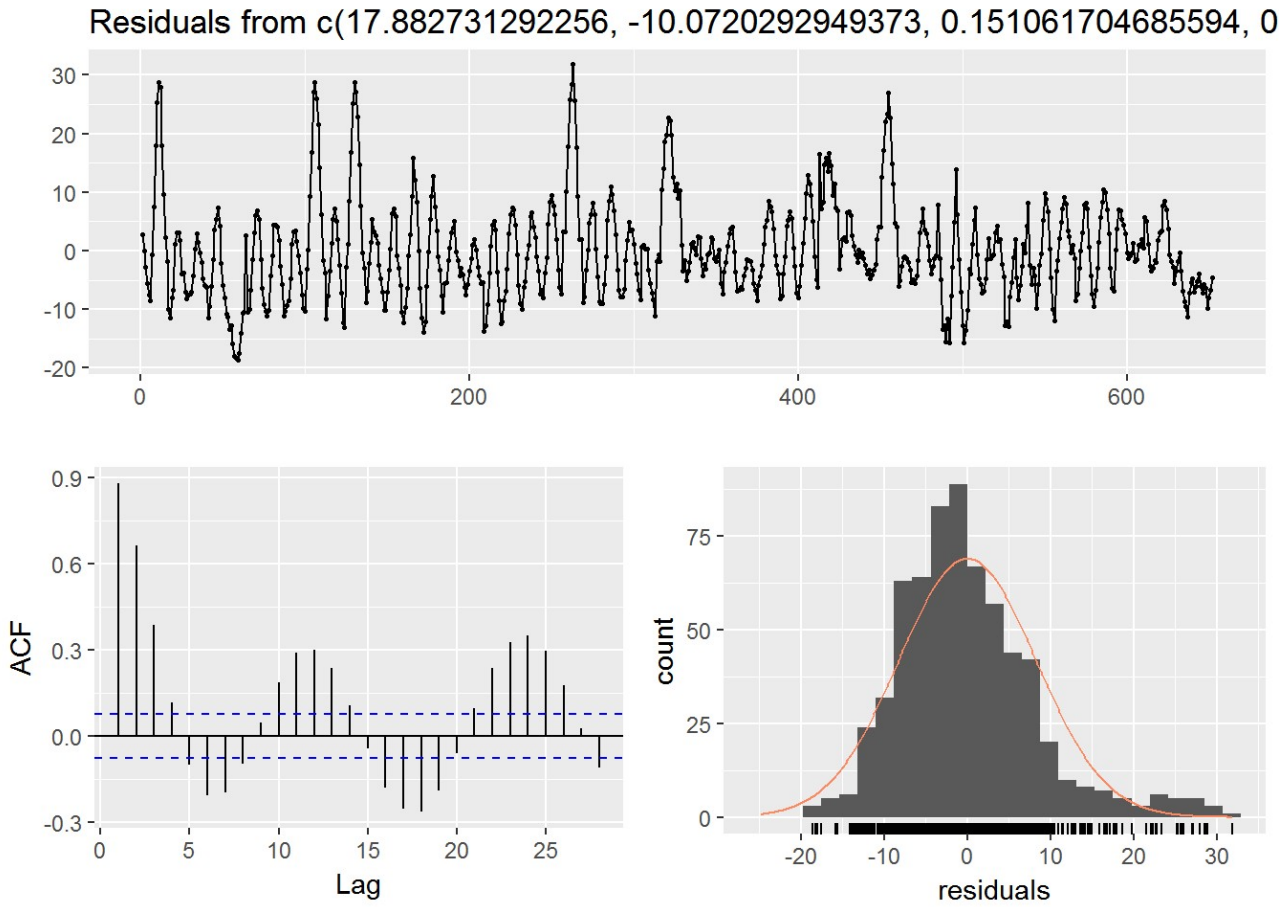
model1_dlm with lag value equal to 8

```
model1_dlm = dlm(x = as.vector(data1$ppt) , y = as.vector(data1$solar) , q = 8, show.summary = TRUE)
```

```
##
## Call:
## lm(formula = y.t ~ ., data = design)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.594  -5.703  -1.197   4.183  31.840
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  17.8827     1.0509  17.016  < 2e-16 ***
## x.t         -10.0720     1.7675  -5.699 1.84e-08 ***
## x.1           0.1511     2.5494   0.059    0.953
## x.2           0.4438     2.5912   0.171    0.864
## x.3           2.0741     2.5977   0.798    0.425
## x.4           1.6863     2.5964   0.649    0.516
## x.5           3.3938     2.5973   1.307    0.192
## x.6           1.3674     2.5825   0.530    0.597
## x.7           3.3675     2.5396   1.326    0.185
## x.8          -2.6471     1.7514  -1.511    0.131
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.327 on 642 degrees of freedom
## Multiple R-squared:  0.2924, Adjusted R-squared:  0.2825
## F-statistic: 29.48 on 9 and 642 DF,  p-value: < 2.2e-16
##
## AIC and BIC values for the model:
##        AIC      BIC
## 1 4625.986 4675.267
```

Apply diagnostic check using checkresiduals() function.

```
checkresiduals(model1_dlm$model, test=FALSE)
```



Interpretation from residuals of model1_dlm ## The time series plot is not random. There is presence of variance though the series fluctuates over the ## zero mean level. The ACF plot shows the seasonality and high serial correlation. And the distribution is ## not completely normal and seems to be right skewed.

BG test is displayed to test the existence of serial correlation upto the displayed order.

```
bgtest(model1_dlm$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model1_dlm$model
## LM test = 521.34, df = 1, p-value < 2.2e-16
```

Interpretation of BG Test and residual check for model1 (dlm)

According to this test and ACF plot we can conclude that serial correlation left in residuals is highly

significant. Also from the time series plot and histogram of residuals, there is an obvious non-random

pattern and very high residual values that violate general assumptions

R squared value is not high but we still need to check for multicollinearity. Overall model is

significant at 5% level of significance with a p value of 2.2e-16. Only first lag value is significant.

To confirm the effect of multicollinearity we will display variance inflation factors (VIFs). I the value

of VIF is greater than 10, we conclude that the effect of multicollinearity is high.

```
VIF.model= vif(model1_dlm$model)
VIF.model
```

```
##      x.t      x.1      x.2      x.3      x.4      x.5      x.6      x.7
## 3.673239 7.647952 7.906673 7.949491 7.923589 7.910292 7.823860 7.568329
##      x.8
## 3.632023
```

```
VIF.model>10
```

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

Interpretation of VIF model

From the VIF values, it can be stated that stimates of the finite DLM coefficients are not suffering

from multicollinearity.

Polynomial lag models to restrict lag weights

```
model_poly = polyDlm(x = as.vector(data1$ppt) , y = as.vector(data1$solar) , q
= 8, k=2, show.beta= TRUE,
                 show.summary = TRUE)
```

```
##
## Call:
## lm(formula = y.t ~ ., data = z)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.818  -5.785  -1.357   4.376  31.750
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 17.43494    1.03121   16.91   <2e-16 ***
## z.t0        -6.75306    0.57384  -11.77   <2e-16 ***
## z.t1         4.30672    0.32482   13.26   <2e-16 ***
## z.t2        -0.45851    0.03978  -11.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.329 on 648 degrees of freedom
## Multiple R-squared:  0.2854, Adjusted R-squared:  0.2821
## F-statistic: 86.25 on 3 and 648 DF,  p-value: < 2.2e-16
##
## Estimates and t-tests for beta coefficients:
##        Estimate Std. Error t value  P(>|t|)
## beta.0  -6.7500      0.574 -11.800 4.28e-29
## beta.1  -2.9000      0.344  -8.440 2.06e-16
## beta.2   0.0263      0.250   0.105 9.16e-01
## beta.3   2.0400      0.259   7.870 1.50e-14
## beta.4   3.1400      0.274  11.500 8.81e-28
## beta.5   3.3200      0.259  12.800 1.31e-33
## beta.6   2.5800      0.250  10.300 3.22e-23
## beta.7   0.9270      0.344   2.700 7.18e-03
## beta.8  -1.6400      0.574  -2.870 4.30e-03
```

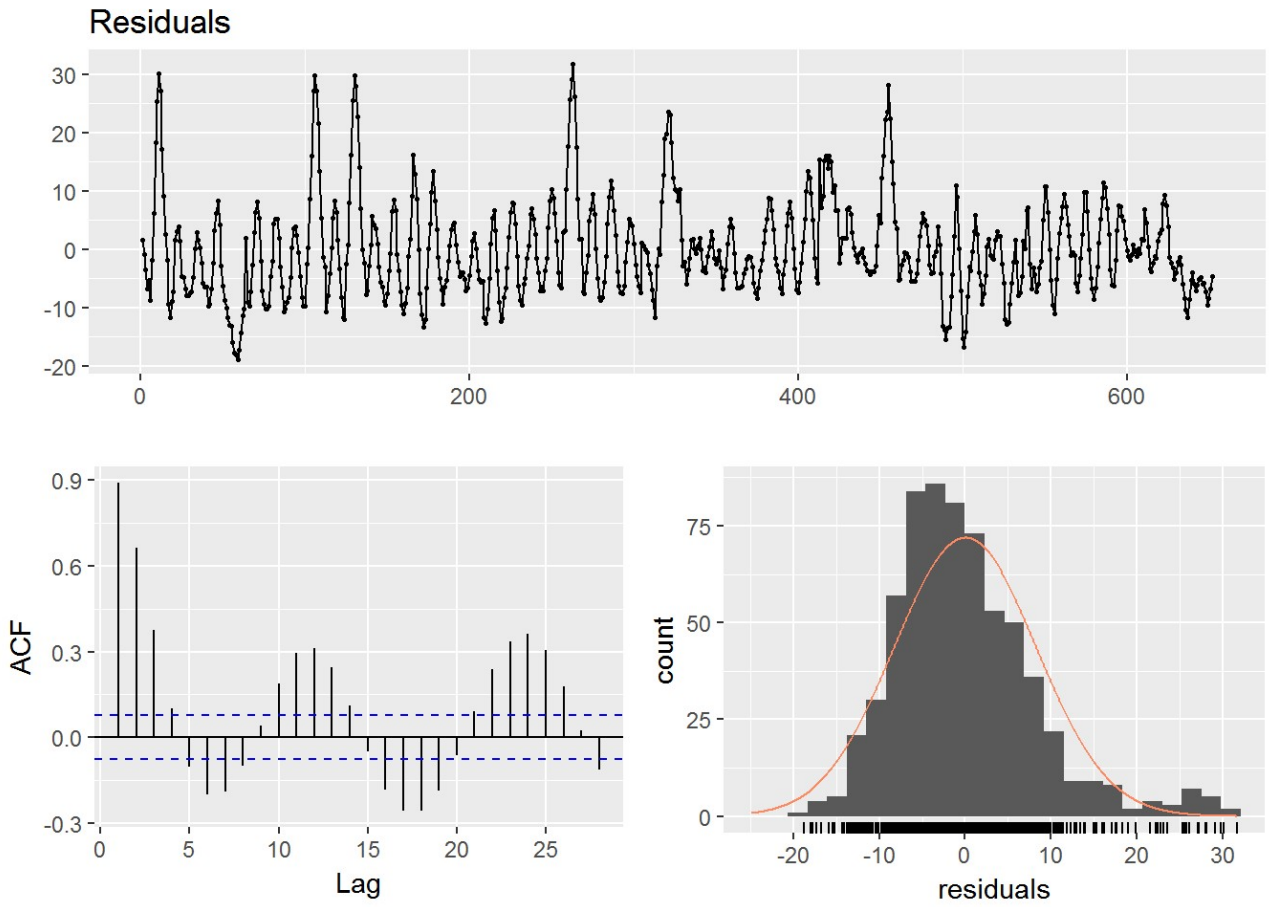
Interpretation

All the distributed lag weights are significant at 5% level of significance. The standard errors of estimators

are much less han their unconstrained counterparts. This implies that we have more precise estimates with

polynomial DLM.

```
checkresiduals(model_poly$model$residuals)
```




```
bgtest(model_poly$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model_poly$model
## LM test = 517.73, df = 1, p-value < 2.2e-16
```

Interpretation

There is no change in the residual plots in comparison to dlm model.

Specification of Finite Lag Length

```
mod.12 = dlm(x= as.vector(data1$ppt), y= as.vector(data1$solar), q=12, show.sum
mary=FALSE) # Best
mod.11 = dlm(x= as.vector(data1$ppt), y= as.vector(data1$solar), q=11, show.sum
mary=FALSE)
mod.10 = dlm(x= as.vector(data1$ppt), y= as.vector(data1$solar), q=10, show.sum
mary=FALSE)
mod.9 = dlm(x= as.vector(data1$ppt), y= as.vector(data1$solar), q=9, show.summa
ry=FALSE)
mod.8 = dlm(x= as.vector(data1$ppt), y= as.vector(data1$solar), q=8, show.summa
ry=FALSE)
mod.7 = dlm(x= as.vector(data1$ppt), y= as.vector(data1$solar), q=7, show.summa
ry=FALSE)
mod.6 = dlm(x= as.vector(data1$ppt), y= as.vector(data1$solar), q=6, show.summa
ry=FALSE)
mod.5 = dlm(x= as.vector(data1$ppt), y= as.vector(data1$solar), q=5, show.summa
ry=FALSE)
mod.4 = dlm(x= as.vector(data1$ppt), y= as.vector(data1$solar), q=4, show.summa
ry=FALSE)
mod.3 = dlm(x= as.vector(data1$ppt), y= as.vector(data1$solar), q=3, show.summa
ry=FALSE)
mod.2 = dlm(x= as.vector(data1$ppt), y= as.vector(data1$solar), q=2, show.summa
ry=FALSE)
mod.1 = dlm(x= as.vector(data1$ppt), y= as.vector(data1$solar), q=1, show.summa
ry=FALSE)
```

Choosing the best model amongst distributed lag models considering AIC, BIC and MASE

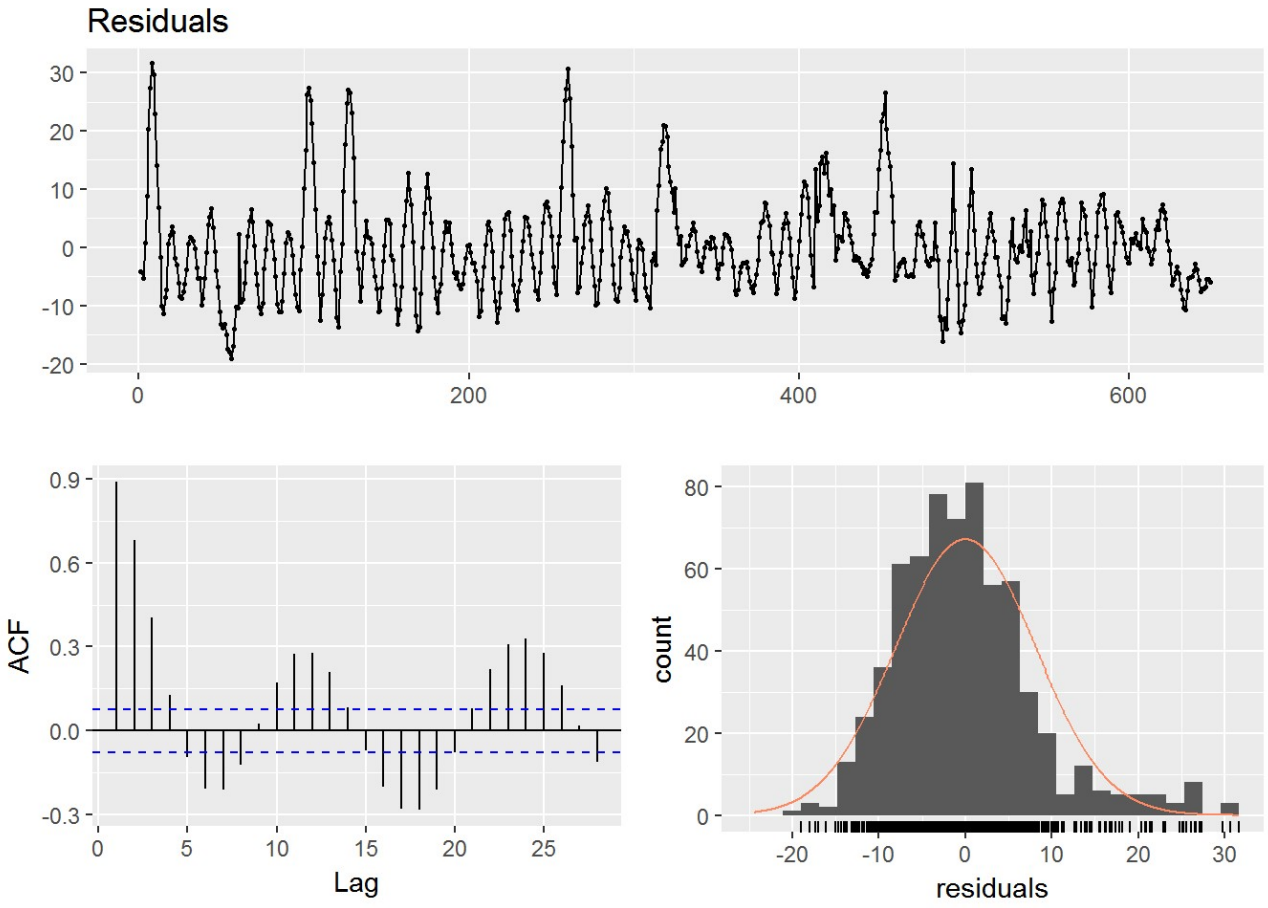
```
models.AIC= AIC(mod.12$model, mod.11$model, mod.10$model, mod.9$model, mod.8$mo
del, mod.7$model, mod.6$model,mod.5$model, mod.4$model,mod.3$model, mod.2$mode
l, mod.1$model)
models.BIC= BIC(mod.12$model, mod.11$model, mod.10$model, mod.9$model, mod.8$mo
del, mod.7$model,mod.6$model, mod.5$model, mod.4$model, mod.3$model, mod.2$mode
l, mod.1$model)


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

```
##              df      AIC
## mod.12$model 15 4578.787
## mod.11$model 14 4590.961
## mod.10$model 13 4602.658
## mod.9$model  12 4615.084
## mod.8$model  11 4625.986
## mod.7$model  10 4632.716
## mod.6$model   9 4637.489
## mod.5$model   8 4644.622
## mod.4$model   7 4663.600
## mod.3$model   6 4688.551
## mod.2$model   5 4712.649
## mod.1$model   4 4728.713
```

```
sort.score(models.BIC, "bic")
```

```
##                df      BIC
## mod.12$model 15 4645.895
## mod.11$model 14 4653.617
## mod.10$model 13 4660.858
## mod.9$model  12 4668.827
## mod.8$model  11 4675.267
## mod.7$model  10 4677.532
## mod.6$model   9 4677.837
## mod.5$model   8 4680.499
## mod.4$model   7 4695.003
## mod.3$model   6 4715.478
## mod.2$model   5 4735.095
## mod.1$model   4 4746.676
```

```
sort.score(models.MASE, "mase")
checkresiduals(mod.11$model$residuals,test=FALSE)
```




```
bgtest(mod.11$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  mod.11$model
## LM test = 524.63, df = 1, p-value < 2.2e-16
```

Interpretation for best Distributed Lag Model

AIC, BIC and MASE measures are lowest for lag length 12 so mod.11 seems to be the best option

of them all. But it's residuals are still significantly correlated, there is variance in time series plot and histogram is not normally distributed. So though mod.11 is best amongst these but still not the suitable for our need.

The Geometric lag model

This model is for infinite geometric DLM. One way to deal with this infinite DLM is to use

Koyck transformation- model, checkresiduals, bgtest

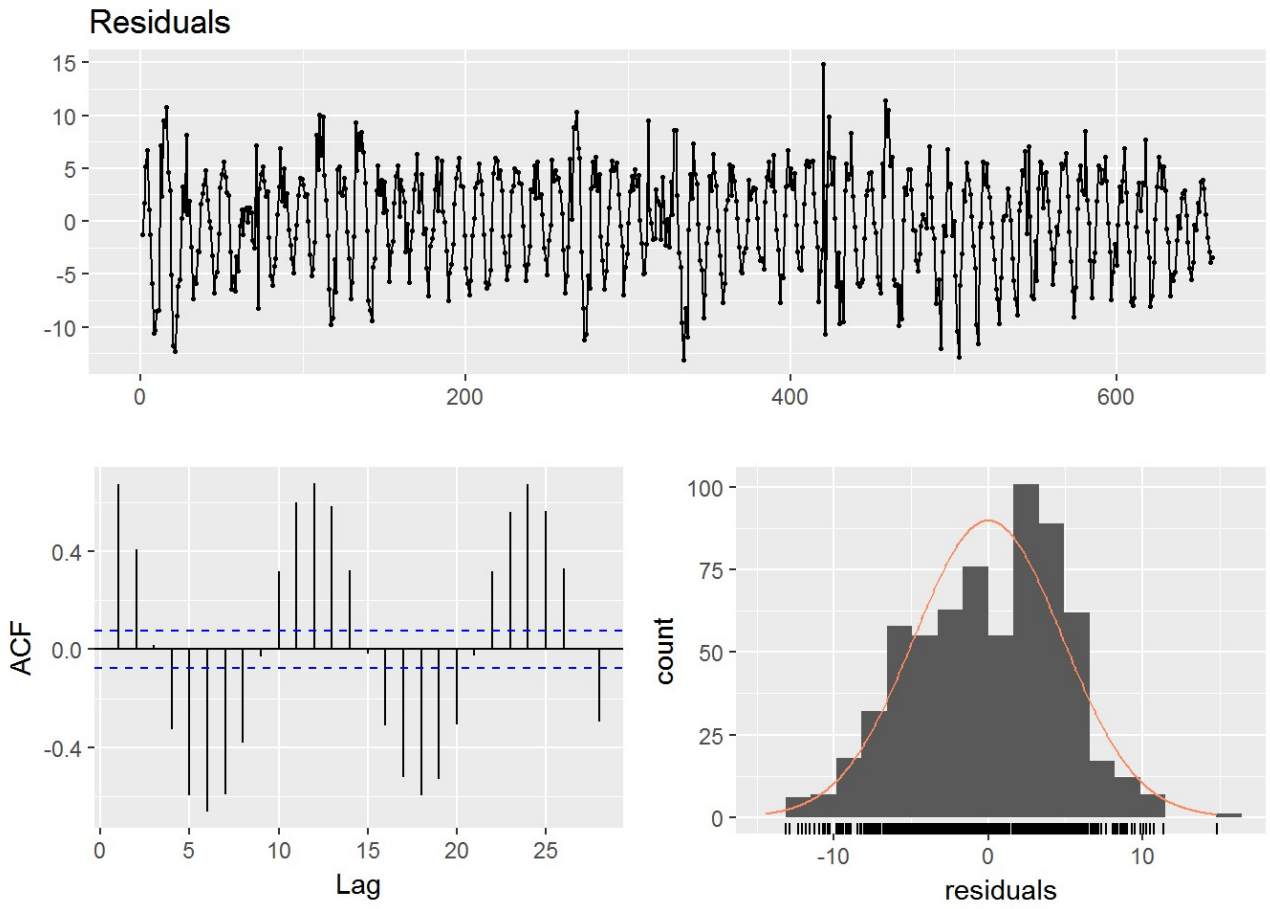
```
model_koyck = koyckDlm(x = as.vector(data1$ppt) , y = as.vector(data1$solar), s
how.summary = TRUE)
```

```
##
## Call:
## ivreg(formula = y.t ~ Y.t_1 + X.t | Y.t_1 + X.t_1)
##
## 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.t_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
##
##                              alpha     beta       phi
## Geometric coefficients:  -154.0203 5.346844 0.9854613
```

```
summary(model_koyck$model, diagnostics=TRUE)
```

```
##
## Call:
## ivreg(formula = y.t ~ Y.t_1 + X.t | Y.t_1 + X.t_1)
##
## 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.t_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
```

```
checkresiduals(model_koyck$model$residuals, test=FALSE)
```




```
bgtest(model_koyck$model)
```

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

Interpretation for Koyck model

The residual plot is not at all in support of using this model. The Time series plot shows seasonality and

is not random. The distribution is not normal and serial correlation factors have rather increased w.r.t.

dlm and polydlm models

From the Weak Instruments line, we conclude that the model at the first stage of least squares

fitting is significant at 5% level of significance. In this model, both ??2 and ??3 are significant

at 5% level. Using the coefficients, we find the estimates of original parameters such that

beta^=-0.0064145 and phi^=0.7268717. So the weights will decline quickly at the rate of 0.72.

Thus, the first lags of appropriations will have more strong impact on solar radiations than latter ones.

In the Koyck DLM, we cannot be sure whether the error term is correlated with the lagged dependent

variable or not. So we test the error term ??t for autocorrelation using the Wu-Hausman test.From the

Wu-Hausman test result in the model output, we reject the null hypothesis that the correlation between explanatory variable and the error term is zero (There is no endogenetiy) at 5% level. So, there is a

explanatory variable and the error term is zero at 5% level.

As none of the above three models are found suitable so we try for Autoregressive Distributed Lag Models.

Autoregressive Distributed Lag Model

Autoregressive DLMs are useful when we cannot find

suitable solutions with neither polynomial nor Kyock DLMs.

```
mod.11 = ardlDlm(x = as.vector(data1$ppt) , y = as.vector(data1$solar) , p =
1 , q = 1 ,
                  show.summary = TRUE)
```

```
##
## Time series regression with "ts" data:
## Start = 2, End = 660
##
## Call:
## dynlm(formula = formula(model.text))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.6739  -2.8807  -0.3641   2.8687  20.1193
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.81174    0.53016   1.531    0.126
## X.t         -6.99904    0.73480  -9.525   <2e-16 ***
## L(X.t, 1)    8.67630    0.71609  12.116   <2e-16 ***
## L(y.t, 1)    0.91001    0.01851  49.161   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.027 on 655 degrees of freedom
## Multiple R-squared:  0.8321, Adjusted R-squared:  0.8314
## F-statistic:  1082 on 3 and 655 DF,  p-value: < 2.2e-16
```

```
mod.22 = ardlDlm(x = as.vector(data1$ppt) , y = as.vector(data1$solar) , p =
2 , q = 2,
                   show.summary = TRUE)
```

```
##
## Time series regression with "ts" data:
## Start = 3, End = 660
##
## Call:
## dynlm(formula = formula(model.text))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.7867  -1.5013  -0.2736   1.2345  18.5318
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.08758    0.39200   5.326 1.39e-07 ***
## X.t         -0.96803    0.59464  -1.628 0.104022
## L(X.t, 1)    0.70618    0.82880   0.852 0.394504
## L(X.t, 2)    2.09832    0.59665   3.517 0.000467 ***
## L(y.t, 1)    1.51119    0.02823  53.539  < 2e-16 ***
## L(y.t, 2)   -0.67673    0.02840 -23.829  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.797 on 652 degrees of freedom
## Multiple R-squared:  0.9192, Adjusted R-squared:  0.9186
## F-statistic:  1484 on 5 and 652 DF,  p-value: < 2.2e-16
```

```
mod.33 = ardlDlm(x = as.vector(data1$ppt) , y = as.vector(data1$solar) , p =
3 , q = 3,
                   show.summary = TRUE) # Best
```

```
##
## Time series regression with "ts" data:
## Start = 4, End = 660
##
## Call:
## dynlm(formula = formula(model.text))
##
## 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
## L(X.t, 1)    1.00698    0.79376   1.269   0.2050
## L(X.t, 2)    1.84292    0.79195   2.327   0.0203 *
## L(X.t, 3)   -0.26711    0.56697  -0.471   0.6377
## L(y.t, 1)    1.26560    0.03696  34.244  < 2e-16 ***
## L(y.t, 2)   -0.13823    0.06139  -2.252   0.0247 *
## L(y.t, 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
```

```
model.AIC = AIC(mod.11$model,mod.22$model,mod.33$model)
model.BIC = BIC(mod.11$model,mod.22$model,mod.33$model)
model.MASE = MASE(mod.11,mod.22,mod.33)

sortScore(model.AIC, "aic")
```

```
##              df      AIC
## mod.33$model  9 3139.409
## mod.22$model  7 3229.051
## mod.11$model  5 3712.311
```

```
## $df
## [1] 9 7 5
##
## $AIC
## [1] 3139.409 3229.051 3712.311
##
## $call
## sortScore.default(x = model.AIC, score = "aic")
##
## attr(,"row.names")
## [1] "mod.33$model" "mod.22$model" "mod.11$model"
## attr(,"class")
## [1] "sortScore" "dLagM"
```

```
sortScore(model.BIC, "bic")
```

```
##              df      BIC
## mod.33$model  9 3179.798
## mod.22$model  7 3260.476
## mod.11$model  5 3734.765
```

```
## $df
## [1] 9 7 5
##
## $BIC
## [1] 3179.798 3260.476 3734.765
##
## $call
## sortScore.default(x = model.BIC, score = "bic")
##
## attr(,"row.names")
## [1] "mod.33$model" "mod.22$model" "mod.11$model"
## attr(,"class")
## [1] "sortScore" "dLagM"
```

```
sortScore(model.MASE, "mase")
```

```
##          n      MASE
## mod.33 657 0.4737144
## mod.22 658 0.4951319
## mod.11 659 0.8392434
```

```
## $n
## [1] 657 658 659
##
## $MASE
## [1] 0.4737144 0.4951319 0.8392434
##
## $call
## sortScore.default(x = model.MASE, score = "mase")
##
## attr(,"row.names")
## [1] "mod.33" "mod.22" "mod.11"
## attr(,"class")
## [1] "sortScore" "dLagM"
```

As per the results ARDL(3,3) is the accurate model as p value greater than 0.05 (=0.1224).

From the time series plot of standardised residuals, we can infer that standardised residuals are

distributed around zero mean level but have got variance. From the histogram, we can say that

standardised residuals are not normally distributed. Also there are outliers. Resulting in large serial

correlations thereby, ACF and PACF of the standardised residuals do not suggest existence of auto correlations

within error terms.

Forecasting for distributed lag models

```
data_x <- read.csv("C:/Users/user/Downloads/data.x.csv")
#data_x
dx<- data.frame(data_x)
dx
```

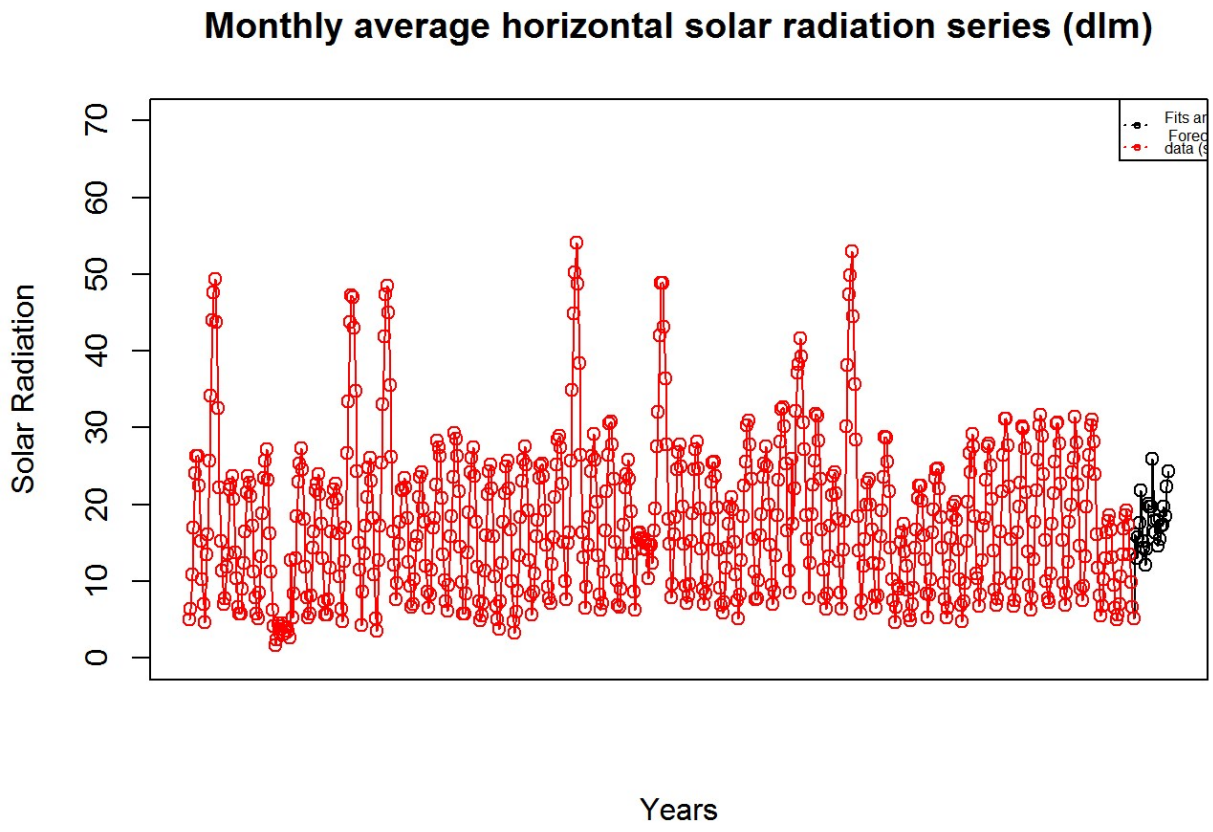
```
##             x
## 1  0.18901000
## 2  0.69726252
## 3  0.59521349
## 4  0.48738853
## 5  0.26167702
## 6  0.80860665
## 7  0.94186202
## 8  0.90563633
## 9  1.05996468
## 10 0.34143878
## 11 0.52580532
## 12 0.60247106
## 13 0.10986063
## 14 0.78146471
## 15 0.69685501
## 16 0.50241391
## 17 0.64938561
## 18 0.74596077
## 19 0.66304712
## 20 0.53377011
## 21 0.61542621
## 22 0.54606508
## 23 0.14267332
## 24 0.01365041
```

```
forecasts.dlm = dlmForecast(model = model1_dlm, x = dx$x, h=24)$forecasts
forecasts.dlm
```

```
##  [1] 16.13450 13.03431 15.68860 17.56535 21.76939 15.25996 14.38213
##  [8] 12.15889 14.19562 20.03604 19.74318 19.66448 25.98795 17.82902
## [15] 16.33176 17.94173 14.64094 15.51710 17.26184 17.34064 19.70004
## [22] 18.51466 22.29337 24.31284
```

Plot for the forecasts

```
y.extended = c(data1$solar , forecasts.dlm)
par(mfrow = c(1, 1))
plot(ts(y.extended),type="o",xaxt="n", ylim= c(0, 70), ylab = "Solar Radiatio
n", xlab = "Years",
     main="Monthly average horizontal solar radiation series (dl
m)")
lines(data1$solar,col="Red",type="o")
legend("topright",lty=10, pch = 1, text.width = 25, text.font = 150, col=c("bla
ck","red"), c("Fits and \n Forecasts",

"data (solar)"), cex=0.5, y.intersp=0.5)
```



#Interpretation from dlm forecasts ## Forecast from dlm does not seems to follow prior sesonality patterns so it might not be the right ##forecast. # Forecast for the PolyDistributed lag model

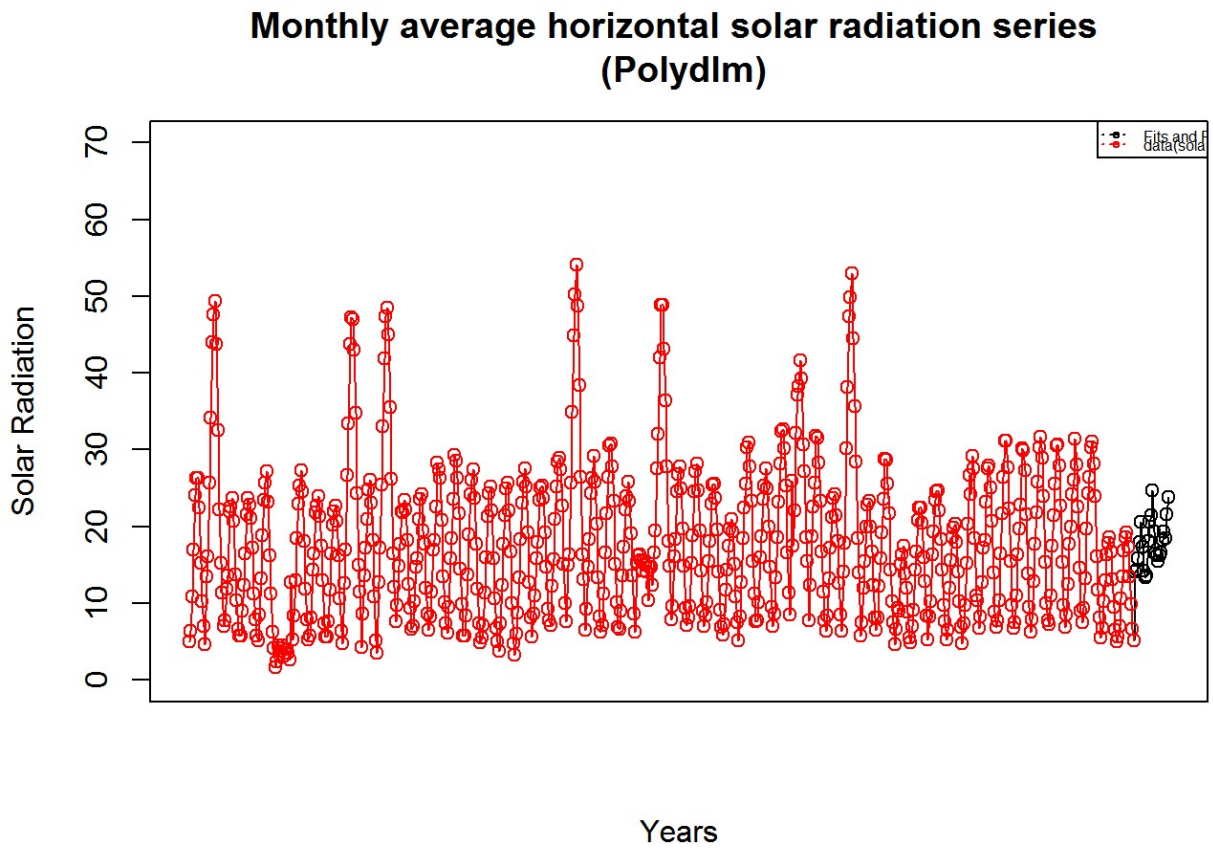
```
forecasts.polydlm = polyDlmForecast(model = model_poly, x = dx$x, h=24)$forecas
ts
forecasts.polydlm
```

```
##  [1] 14.04550 14.39307 15.85818 18.02184 20.62532 17.31576 14.17035
##  [8] 13.32710 13.63119 18.06498 20.55526 21.50601 24.73470 19.39707
## [15] 16.50121 16.27293 15.47814 16.24401 16.58991 18.26608 19.31510
## [22] 18.48013 21.54250 23.77480
```

```
y.extended = c(data1$solar , forecasts.polydlm)
```

Plot for the PolyDistributed Lag Model

```
plot(ts(y.extended),type="o",xaxt="n", ylim= c(0, 70), ylab = "Solar Radiatio
n", xlab = "Years",
     main="Monthly average horizontal solar radiation series \n (Polydlm)")
lines(data1$solar,col="Red",type="o")
legend("topright",lty=10, pch = 1, text.width = 40, text.font = 100, col=c("bla
ck","red"),
       c("Fits and Forecasts (polydlm)","data(solar)"), cex=0.5, y.intersp=0.5)
```



Interpretation from Polydlm forecasts # The forecasts do not match the prior series in seasonality patterns so it might not be the right forecast.

Forecast for Koyckdlm model

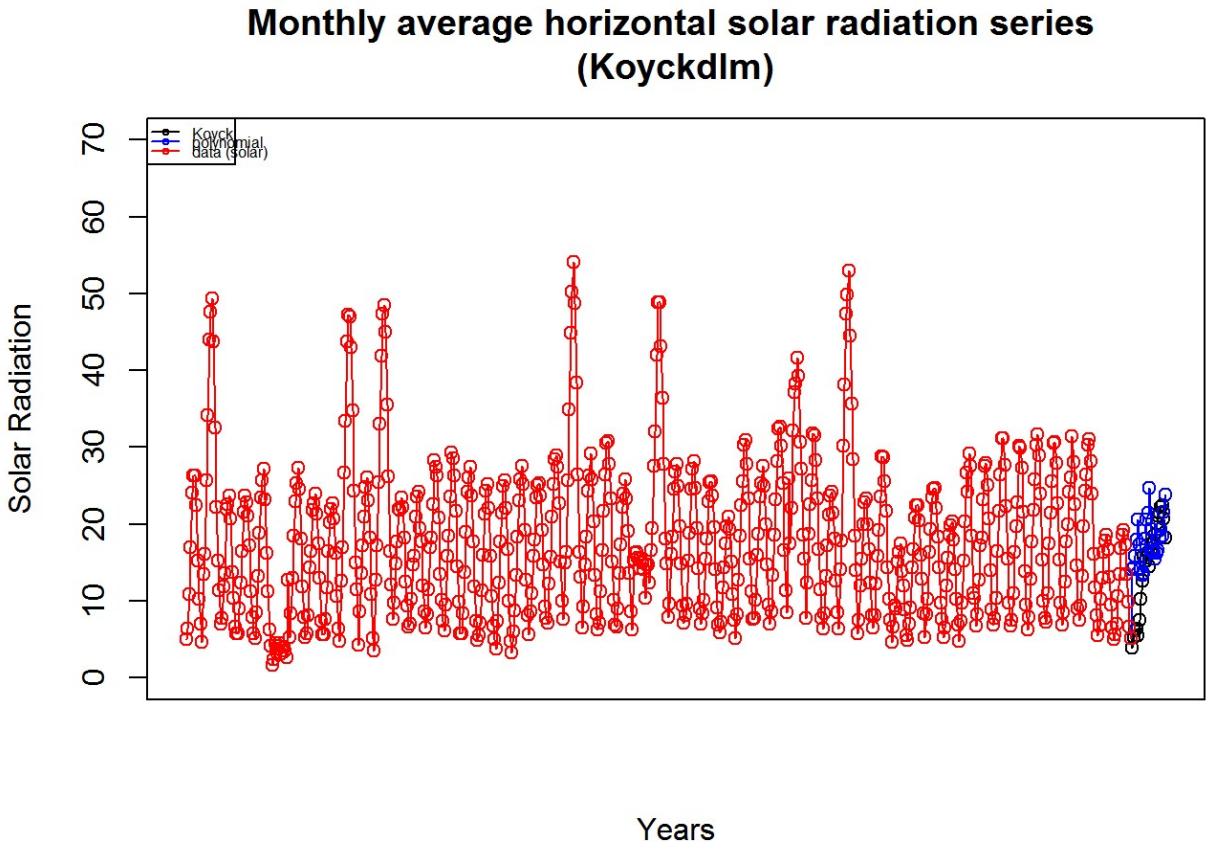
```
forecasts.koyckdlm = koyckDlmForecast(model = model_koyck, x = dx$x, h=24)$fore
casts
forecasts.koyckdlm
```

```
##  [1]  3.844783  5.277785  6.144312  6.421717  5.488246  7.492693 10.180493
##  [8] 12.635524 15.880031 15.235522 15.586161 16.341623 14.452190 16.181190
## [15] 17.432657 17.626283 18.602929 20.081748 21.095740 21.403767 22.143917
## [22] 22.502444 20.698885 18.231682
```

```
y.extended2 = c(data1$solar , forecasts.koyckdlm)
```

Plot for Koyckdlm model

```
plot(ts(y.extended2),type="o",xaxt="n", ylim= c(0, 70), ylab = "Solar Radiatio
n", xlab = "Years",
     main="Monthly average horizontal solar radiation series \n (Koyckdlm) ")
lines(y.extended,col="Blue",type="o") # poly
lines(data1$solar,col="Red",type="o")
par(new=TRUE)
legend("topleft",lty=1, pch = 1, text.width = 25, text.font = 100, col=c("blac
k", "blue","red"), c("Koyck", "polynomial",

              "data (solar)"), cex=0.5, y.intersp=0.5)# Forecasts for Autoregre
ssive Distributed lag models
```



Interpretation from above forecast

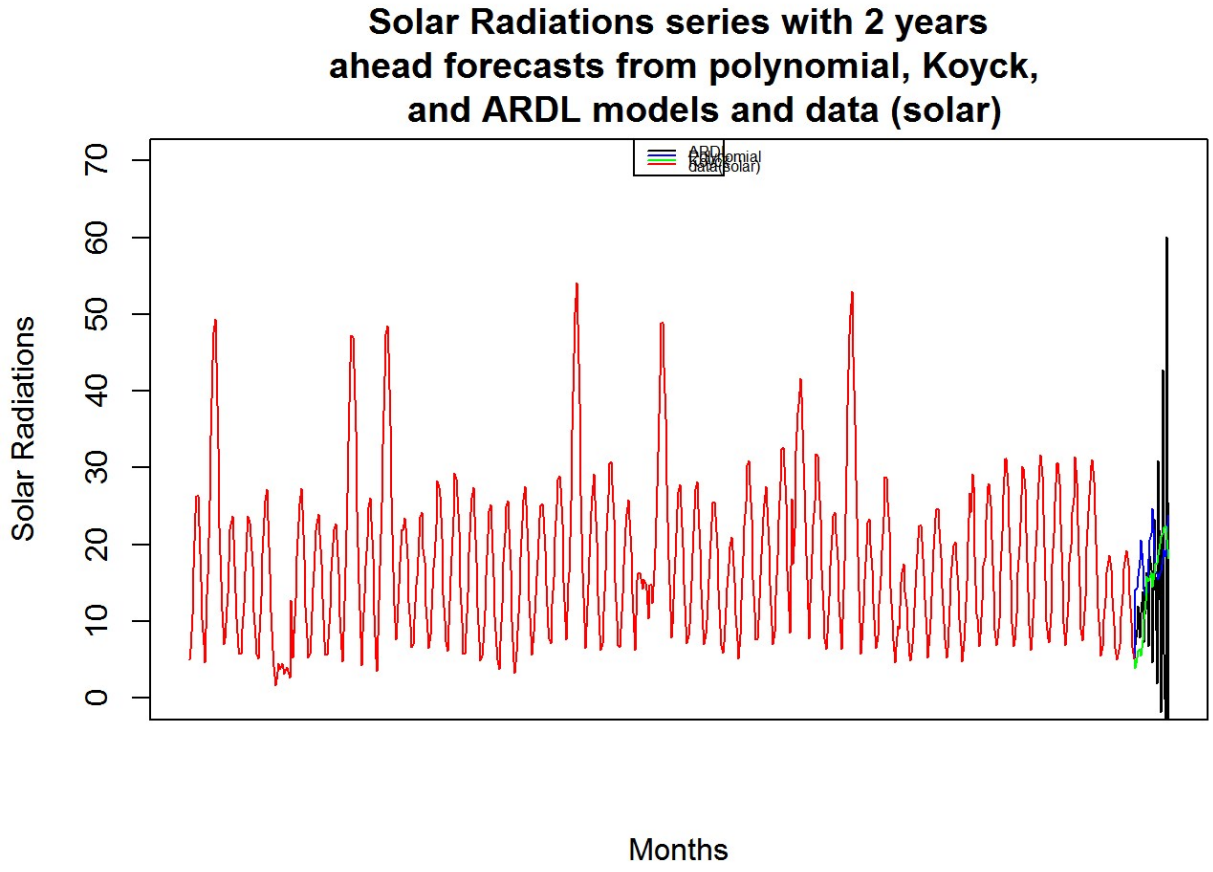
Polydlm model's forecast are better in capturiing prior trend and seasonality in comparison to koyck model

Forecast for Autoregressive Distributed lag model and its plot

```
forecasts.ardldlm = ardlDlmForecast(model = mod.33, x = dx$x, h=24)$forecasts
forecasts.ardldlm
```

```
##  [1]  6.857344  8.845041 12.016105  7.852801 11.098041 13.833209  7.236146
##  [8] 14.523800 16.293669  6.765202 18.585099 16.734391  4.580927 23.319762
## [15] 15.817759  1.895151 30.964447 12.859888 -1.902011 42.745119  5.709499
## [22] -6.126360 60.030120 -9.114401
```

```
#summary(model.33$model)


y.extended3 = c(data1$solar , forecasts.ardldlm)

{
  plot(ts(y.extended3),type="l",xaxt="n", ylim= c(0, 70), ylab = "Solar Radiati
ons", xlab = "Months", main="Solar Radiations series with 2 years
  ahead forecasts from polynomial, Koyck,
     and ARDL models and data (solar)")
 lines(y.extended,col="Blue",type="l")
 lines(y.extended2,col="Green",type="l")
 lines(data1$solar,col="Red",type="l")
  legend("top",lty=1, text.width = 20, col=c("black","blue","green", "red"), c
("ARDL", "Polynomial", "Koyck", "data(solar)"), cex=0.5, y.intersp=0.25)
}
```



Interpretation from ardldlm forecast plot ## The Auto regressive model's forecast is completely out of scope and do not capture the prior trend and seaonality by any means so it's not suitable model.

Intervention Analysis Model

Using only solar radiation series for these models

```
solar.ts <- ts(as.vector(data1$solar), start=c(1960,1), frequency=12)
head(solar.ts)
```

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

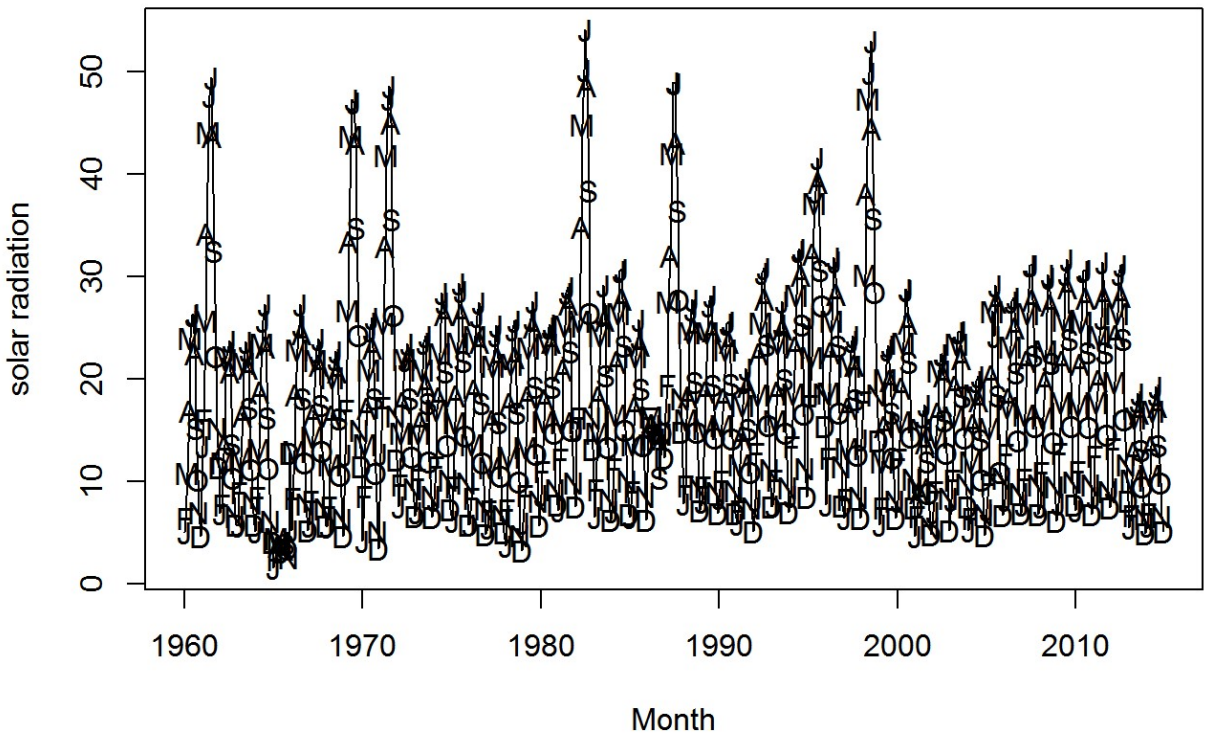
```
class(solar.ts)
```

```
## [1] "ts"
```

```
#solar.ts
par(mfrow = c(1, 1))
```

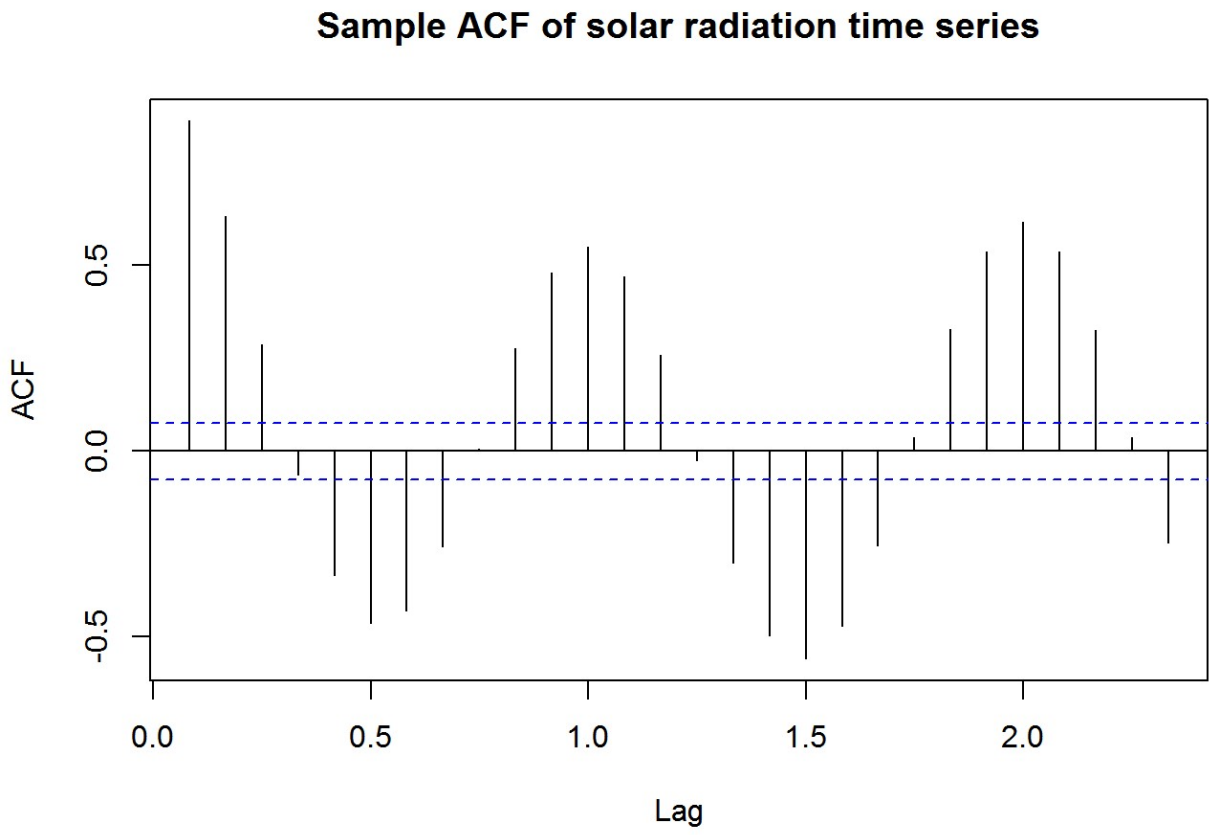
Plotting the solar radiation time series

```
plot(solar.ts,ylab='solar radiation',xlab='Month', type="l")
points( y=solar.ts, x=time(solar.ts), pch=as.vector(season(solar.ts)))
```

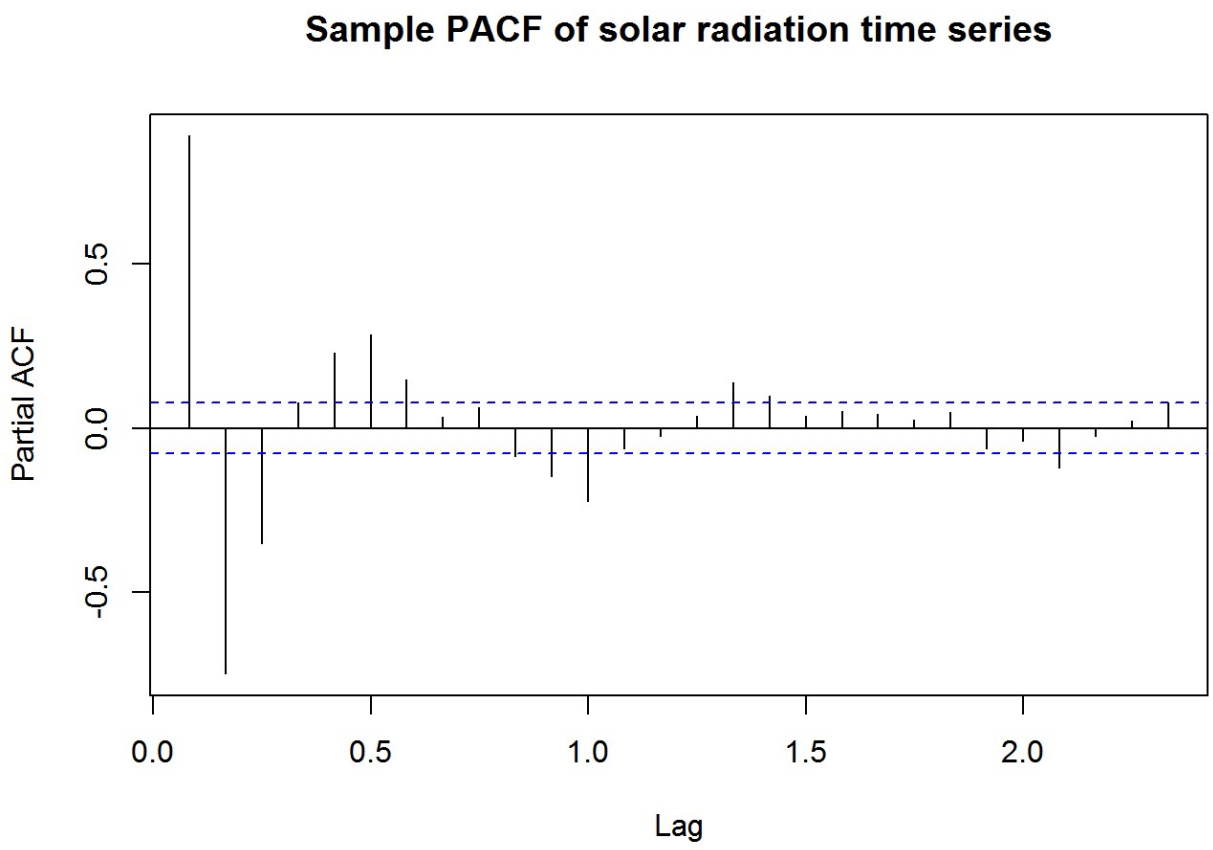


Getting the ACF and PACF plots for solar radiation series

```
acf(solar.ts, main="Sample ACF of solar radiation time series")
```




```
pacf(solar.ts, main="Sample PACF of solar radiation time series")
```



Interpretation from the ACF and PACF plot of solar radiation tiime series

From the ACF and PACF plts, existence of seasonal effect is obvious. There is no obvious presence of ##trend.

```
solar.ts <- ts(as.vector(data1$solar), start=c(1960,1), frequency=12)
#data(solar.ts)
Y.t = solar.ts
T = 64 # The time point when the intervention occurred
P.t = 1*(seq(solar.ts) == T)
P.t.1 = Lag(P.t,+1)
```

In our implementation, we will include a trend and a seasonal component in the model first.

To fit the model, we will use dynlm function of dynlm package.

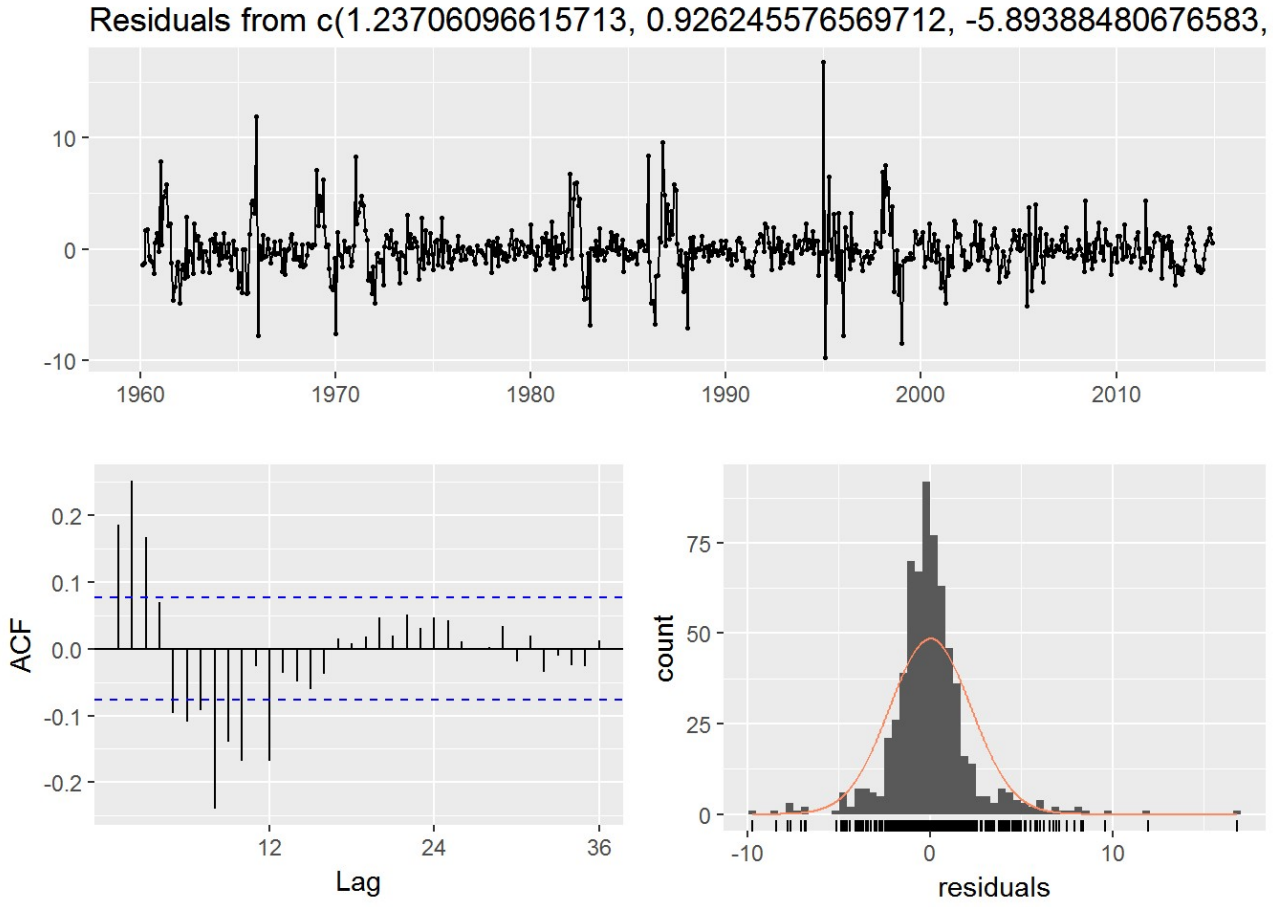
```
mod0 = dynlm(Y.t ~ L(Y.t , k = 1 ) + P.t.1 + P.t+ season(Y.t)+ trend(Y.t))
summary(mod0)
```

```
##
## Time series regression with "ts" data:
## Start = 1960(2), End = 2014(12)
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, k = 1) + P.t.1 + P.t + season(Y.t) +
##     trend(Y.t))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.7276 -0.9960 -0.1367  0.8145 16.8460
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.298064   0.362009   3.586 0.000362 ***
## L(Y.t, k = 1)   0.926229   0.014221  65.132  < 2e-16 ***
## P.t.1          -5.943614   2.313029  -2.570 0.010405 *
## P.t            -5.659923   2.307174  -2.453 0.014424 *
## season(Y.t)Feb  1.929558   0.436262   4.423 1.14e-05 ***
## season(Y.t)Mar  4.905700   0.438554  11.186  < 2e-16 ***
## season(Y.t)Apr  3.944733   0.455639   8.658  < 2e-16 ***
## season(Y.t)May  5.334905   0.474349  11.247  < 2e-16 ***
## season(Y.t)Jun  3.475954   0.502019   6.924 1.07e-11 ***
## season(Y.t)Jul  1.754952   0.523507   3.352 0.000848 ***
## season(Y.t)Aug -2.028815   0.531067  -3.820 0.000146 ***
## season(Y.t)Sep -4.586801   0.508613  -9.018  < 2e-16 ***
## season(Y.t)Oct -5.730880   0.474542 -12.077  < 2e-16 ***
## season(Y.t)Nov -5.052372   0.447823 -11.282  < 2e-16 ***
## season(Y.t)Dec -2.818024   0.437342  -6.444 2.29e-10 ***
## trend(Y.t)     -0.002207   0.005612  -0.393 0.694227
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.277 on 643 degrees of freedom
## Multiple R-squared:  0.9473, Adjusted R-squared:  0.9461
## F-statistic: 771.2 on 15 and 643 DF,  p-value: < 2.2e-16
```

Since trend coefficient is not significant therefore, we can leave the trend and just carry on with the

seasonality by modeli1-checking residuals and doing bg test

```
modeli1 = dynlm(Y.t ~ L(Y.t , k = 1 ) +P.t.1 + P.t+ season(Y.t))
#summary(modeli1)
checkresiduals(modeli1, test=FALSE)
```




```
bgtest(modeli1)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  modeli1
## LM test = 25.994, df = 1, p-value = 3.425e-07
```

Interpretation from dynlm model considering only seasonality

High Serial Correlation for residuals.

Our model and all coefficients are significant at 5% level of significance. Although most of the

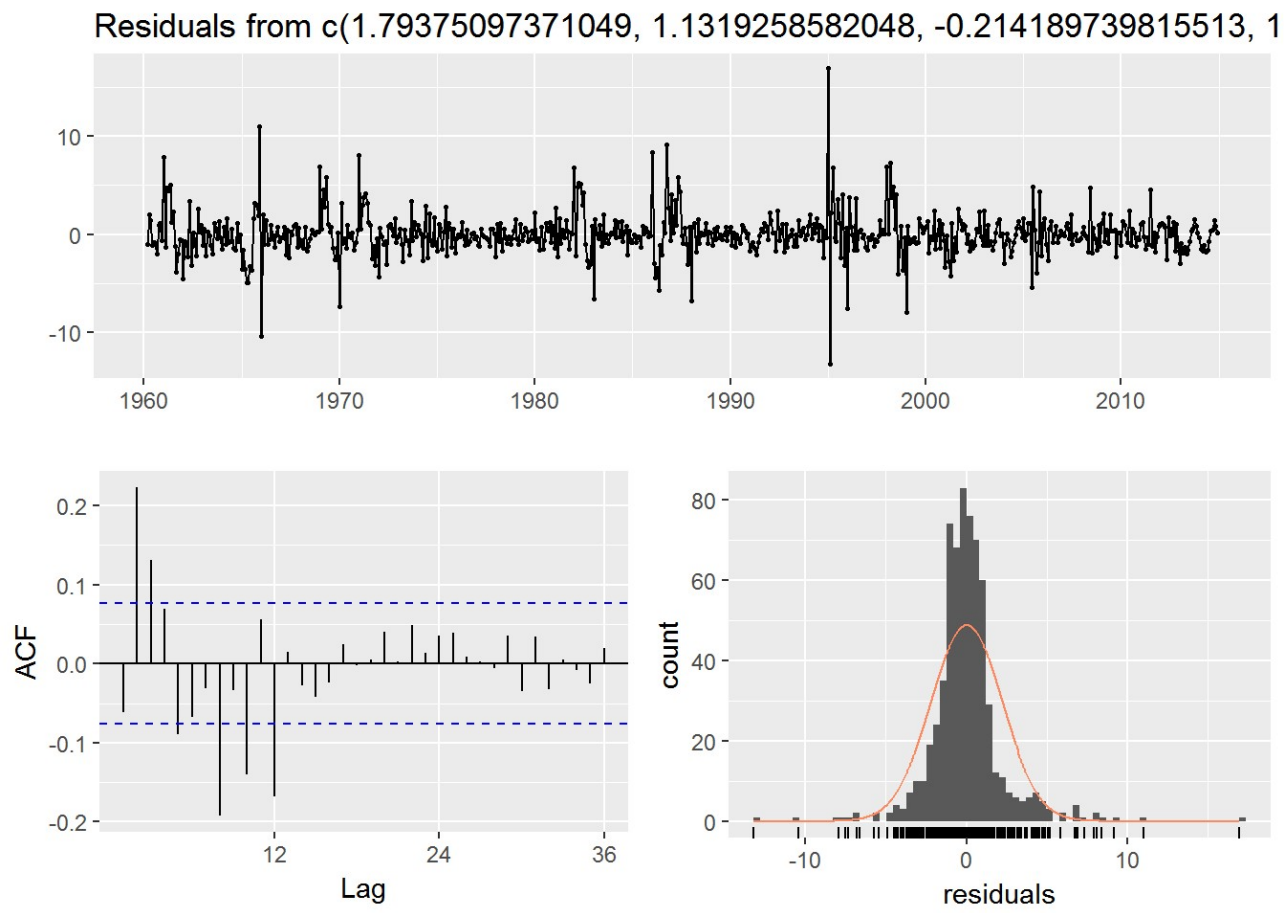
diagnostics are suitable, there is still some serial correlation left in the residuals. To overcome this

issue, we add Yt2 to the model as another predictor and check its residuals

```
modeli2 = dynlm(Y.t ~ L(Y.t , k = 1 ) + L(Y.t , k = 2 ) + season(Y.t)) # Best
summary(modeli2)
```

```
##
## Time series regression with "ts" data:
## Start = 1960(3), End = 2014(12)
##
## Call:
## dynlm(formula = Y.t ~ L(Y.t, k = 1) + L(Y.t, k = 2) + season(Y.t))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.2068  -0.9908  -0.0859   0.8165  16.9505
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.79375    0.33960   5.282 1.75e-07 ***
## L(Y.t, k = 1)   1.13193    0.03848  29.415  < 2e-16 ***
## L(Y.t, k = 2)  -0.21419    0.03848  -5.566 3.82e-08 ***
## season(Y.t)Feb  1.32049    0.44646   2.958 0.003214 **
## season(Y.t)Mar  3.88796    0.46783   8.311 5.64e-16 ***
## season(Y.t)Apr  2.27273    0.52275   4.348 1.60e-05 ***
## season(Y.t)May  4.00335    0.50732   7.891 1.29e-14 ***
## season(Y.t)Jun  2.05979    0.54704   3.765 0.000182 ***
## season(Y.t)Jul  0.81712    0.53504   1.527 0.127200
## season(Y.t)Aug -2.54463    0.52655  -4.833 1.69e-06 ***
## season(Y.t)Sep -4.30206    0.50509  -8.517  < 2e-16 ***
## season(Y.t)Oct -4.98893    0.48970 -10.188  < 2e-16 ***
## season(Y.t)Nov -4.19952    0.46948  -8.945  < 2e-16 ***
## season(Y.t)Dec -2.24708    0.44348  -5.067 5.29e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.243 on 644 degrees of freedom
## Multiple R-squared:  0.9487, Adjusted R-squared:  0.9477
## F-statistic: 916.2 on 13 and 644 DF,  p-value: < 2.2e-16
```

```
checkresiduals(modeli2, test=FALSE)
```



Comparing AICs and BICs for both the above models (lag 1 and 2)

```
AIC(modeli1)
```

```
## [1] 2970.496
```

```
AIC(modeli2)
```

```
## [1] 2946.243
```

```
BIC(modeli1)
```

```
## [1] 3042.347
```

```
BIC(modeli2)
```

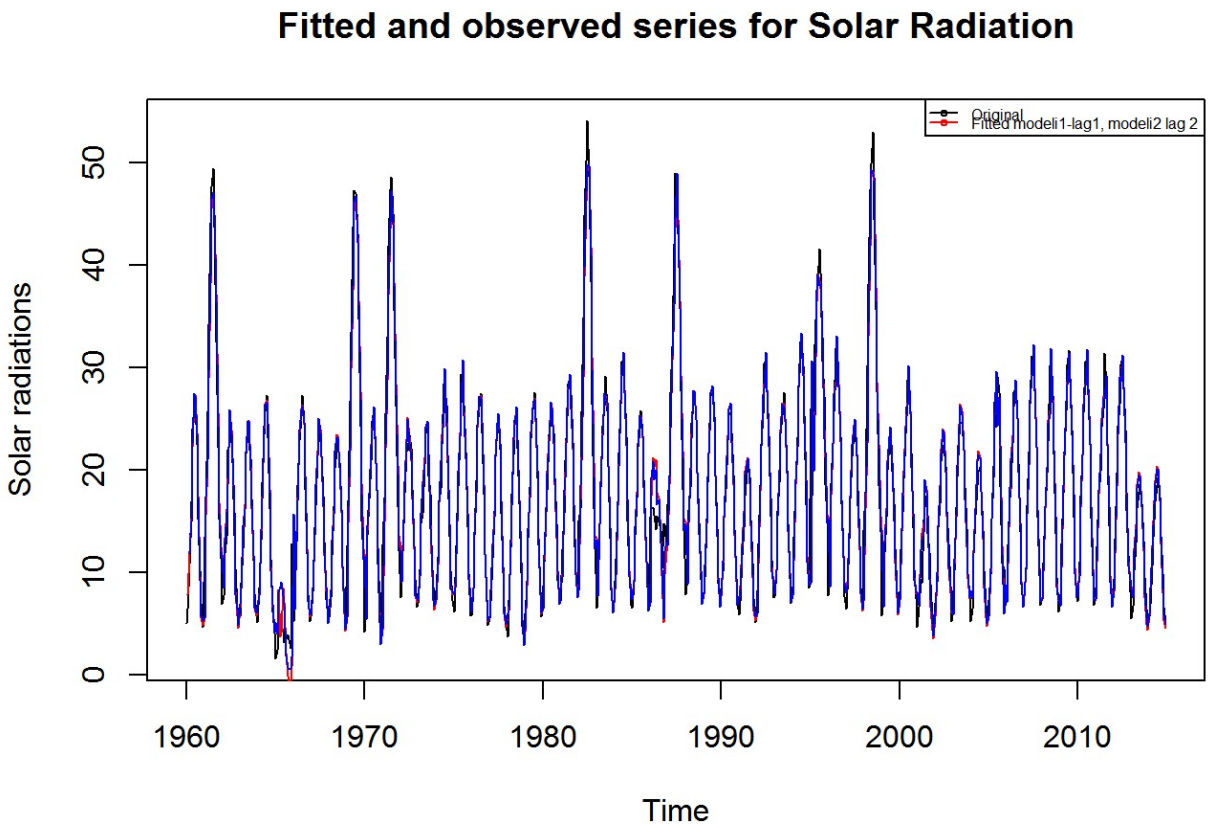
```
## [1] 3013.581
```

Interpretation for Intervention Analysis modeli2

As seen from modeli2 ACF plot there is decrease in serial correlation and also the AIC and BIC values are lower

in comparison to that of model1. Therefore we will go with model2 and plot for it.

```
plot(solar.ts, ylab='Solar radiations',type="l", main="Fitted and observed seri
es for Solar Radiation")
lines(modeli1$fitted.values,col="red")
lines(modeli2$fitted.values,col="blue")
legend("topright",lty=1, pch = 1, col=c("black","red", "blue"), c("Original","F
itted modeli1-lag1, modeli2 lag 2"),cex=0.5, y.intersp=0.5)
```



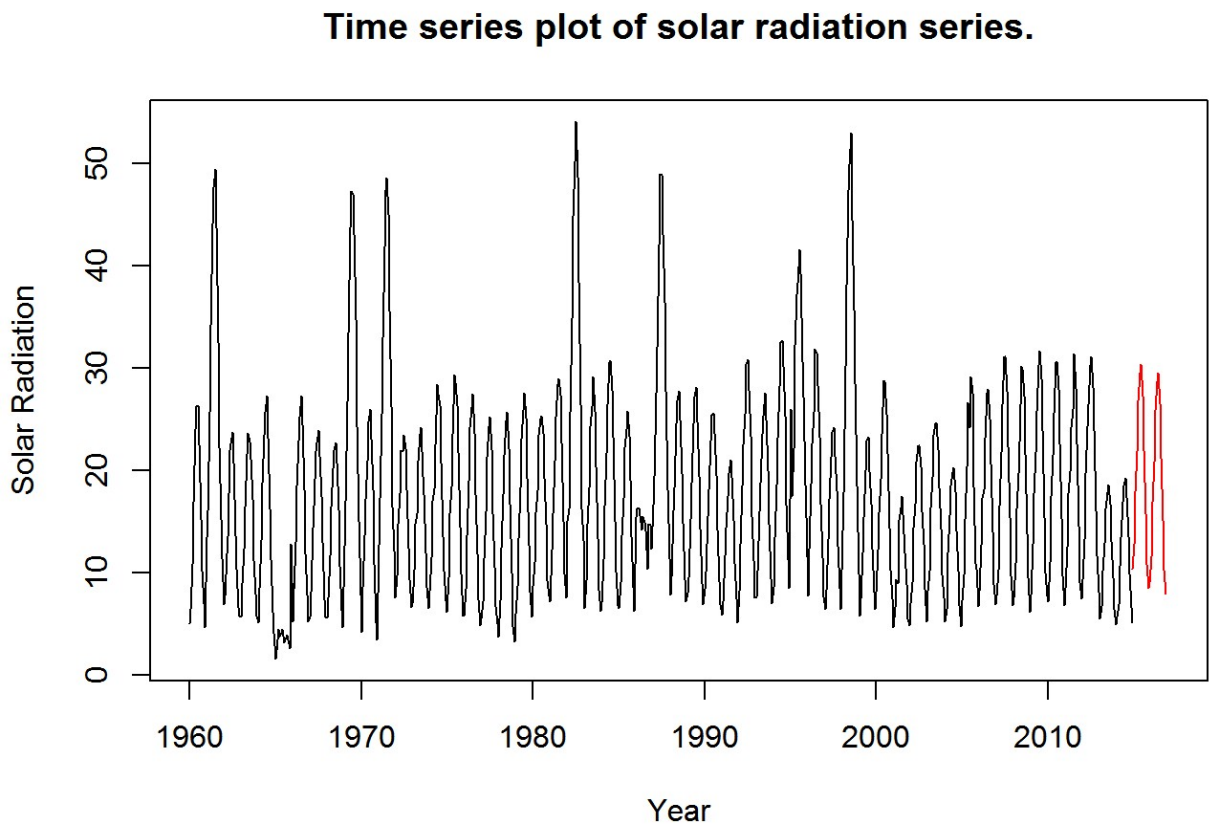
Interpretation ## Observed and fitted values are plotted above. This plot indicates a good agreement between modeli2 and the ## original series.

Forecasting and plotting for intervention analysis models

```
q = 24
n = nrow(modeli2$model)
solar.frc = array(NA , (n + q))
solar.frc[1:n] = Y.t[1:length(Y.t)]


for (i in 1:q){
  months = array(0,11)
  months[(i-1)%%12] = 1
  data.new = c(1,solar.frc[n-1+i], P.t.1[n] , P.t[n] ,months)
  solar.frc[n+i] = as.vector(modeli1$coefficients) %*% data.new
}
solar.frc[-(1:n)]
```

```
##  [1] 10.383726 12.785409 17.985872 21.840635 26.801014 29.537230 30.350404
##  [8] 27.319636 21.954279 15.840465 10.855995  8.473398  9.085508 11.582940
## [15] 16.872091 20.809000 25.845467 28.652159 29.530610 26.560306 21.250953
## [22] 15.189012 10.252590  7.914497
```

```
par(mfrow=c(1,1))

plot(Y.t,xlim=c(1960,2017),ylab='Solar Radiation',xlab='Year',main = "Time seri
es plot of solar radiation series.")
lines(ts(solar.frc[(n+1):(n+q)],start=c(2014,12),frequency = 12),col="red")
```



Interpretation from intervention analysis forecast plot ## The forecasts seem to capture the prior trend ans seasonality effectively. Hence can be considered as acceptable model.

Simple Exponential Smoothening Methods

```
data("solar.ts")
fit1 <- ses(solar.ts, alpha=0.1, initial="simple", h=24) # Set alpha to a smal
l value
summary(fit1)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = solar.ts, h = 24, initial = "simple", alpha = 0.1)
##
##   Smoothing parameters:
##     alpha = 0.1
##
##   Initial states:
##     l = 5.0517
##
##   sigma:  9.4649
## Error measures:
##                       ME     RMSE      MAE       MPE     MAPE     MASE
## Training set 0.1082408 9.464872 7.716889 -35.66493 63.73288 1.267744
##                    ACF1
## Training set 0.8756093
##
## Forecasts:
##          Point Forecast        Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2015       12.19562  0.065897818 24.32534 -6.355189 30.74643
## Feb 2015       12.19562  0.005400079 24.38584 -6.447712 30.83895
## Mar 2015       12.19562 -0.054798900 24.44604 -6.539779 30.93102
## Apr 2015       12.19562 -0.114703502 24.50594 -6.631395 31.02263
## May 2015       12.19562 -0.174318005 24.56556 -6.722567 31.11381
## Jun 2015       12.19562 -0.233646581 24.62488 -6.813302 31.20454
## Jul 2015       12.19562 -0.292693306 24.68393 -6.903607 31.29484
## Aug 2015       12.19562 -0.351462160 24.74270 -6.993486 31.38472
## Sep 2015       12.19562 -0.409957029 24.80120 -7.082946 31.47418
## Oct 2015       12.19562 -0.468181710 24.85942 -7.171993 31.56323
## Nov 2015       12.19562 -0.526139911 24.91738 -7.260632 31.65187
## Dec 2015       12.19562 -0.583835260 24.97507 -7.348870 31.74011
## Jan 2016       12.19562 -0.641271300 25.03251 -7.436711 31.82795
## Feb 2016       12.19562 -0.698451496 25.08969 -7.524160 31.91540
## Mar 2016       12.19562 -0.755379237 25.14662 -7.611224 32.00246
## Apr 2016       12.19562 -0.812057838 25.20330 -7.697906 32.08914
## May 2016       12.19562 -0.868490541 25.25973 -7.784212 32.17545
## Jun 2016       12.19562 -0.924680519 25.31592 -7.870148 32.26139
## Jul 2016       12.19562 -0.980630879 25.37187 -7.955716 32.34695
## Aug 2016       12.19562 -1.036344658 25.42758 -8.040923 32.43216
## Sep 2016       12.19562 -1.091824834 25.48306 -8.125773 32.51701
## Oct 2016       12.19562 -1.147074320 25.53831 -8.210270 32.60151
## Nov 2016       12.19562 -1.202095969 25.59333 -8.294418 32.68566
## Dec 2016       12.19562 -1.256892579 25.64813 -8.378222 32.76946
```

```
fit2 <- ses(solar.ts, alpha=0.8, initial="simple", h=24) # Set alpha to a large value
summary(fit2)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = solar.ts, h = 24, initial = "simple", alpha = 0.8)
##
##   Smoothing parameters:
##     alpha = 0.8
##
##   Initial states:
##     l = 5.0517
##
##   sigma:  5.3164
## Error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.001073712 5.316378 4.574236 -8.172062 32.43497 0.7514636
##                    ACF1
## Training set 0.7536451
##
## Forecasts:
##          Point Forecast      Lo 80    Hi 80      Lo 95    Hi 95
## Jan 2015       5.618649  -1.194563 12.43186  -4.801260 16.03856
## Feb 2015       5.618649  -3.106520 14.34382  -7.725346 18.96264
## Mar 2015       5.618649  -4.669076 15.90637 -10.115069 21.35237
## Apr 2015       5.618649  -6.023773 17.26107 -12.186900 23.42420
## May 2015       5.618649  -7.236494 18.47379 -14.041596 25.27889
## Jun 2015       5.618649  -8.344281 19.58158 -15.735810 26.97311
## Jul 2015       5.618649  -9.370418 20.60772 -17.305151 28.54245
## Aug 2015       5.618649 -10.330672 21.56797 -18.773733 30.01103
## Sep 2015       5.618649 -11.236308 22.47361 -20.158783 31.39608
## Oct 2015       5.618649 -12.095703 23.33300 -21.473115 32.71041
## Nov 2015       5.618649 -12.915292 24.15259 -22.726568 33.96387
## Dec 2015       5.618649 -13.700142 24.93744 -23.926893 35.16419
## Jan 2016       5.618649 -14.454328 25.69163 -25.080320 36.31762
## Feb 2016       5.618649 -15.181186 26.41848 -26.191953 37.42925
## Mar 2016       5.618649 -15.883487 27.12078 -27.266030 38.50333
## Apr 2016       5.618649 -16.563563 27.80086 -28.306117 39.54342
## May 2016       5.618649 -17.223401 28.46070 -29.315252 40.55255
## Jun 2016       5.618649 -17.864706 29.10200 -30.296044 41.53334
## Jul 2016       5.618649 -18.488957 29.72626 -31.250753 42.48805
## Aug 2016       5.618649 -19.097447 30.33475 -32.181358 43.41866
## Sep 2016       5.618649 -19.691312 30.92861 -33.089596 44.32689
## Oct 2016       5.618649 -20.271558 31.50886 -33.977006 45.21430
## Nov 2016       5.618649 -20.839082 32.07638 -34.844959 46.08226
## Dec 2016       5.618649 -21.394686 32.63198 -35.694681 46.93198
```

```
fit3 <- ses(solar.ts, h=24) # Let the software estimate alpha
#Best
summary(fit3)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = solar.ts, h = 24)
##
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l = 5.0923
##
##   sigma:  4.5691
##
##      AIC     AICc      BIC
## 6296.371 6296.407 6309.847
##
## Error measures:
##                       ME     RMSE      MAE       MPE     MAPE     MASE
## Training set 8.500441e-05 4.569082 3.876444 -5.214174 27.30157 0.636829
##                   ACF1
## Training set 0.6678339
##
## Forecasts:
##          Point Forecast       Lo 80    Hi 80      Lo 95    Hi 95
## Jan 2015       5.148434  -0.7070806 11.00395  -3.806803 14.10367
## Feb 2015       5.148434  -3.1321000 13.42897  -7.515550 17.81242
## Mar 2015       5.148434  -4.9929385 15.28981 -10.361457 20.65832
## Apr 2015       5.148434  -6.5617166 16.85858 -12.760696 23.05756
## May 2015       5.148434  -7.9438469 18.24071 -14.874482 25.17135
## Jun 2015       5.148434  -9.1933933 19.49026 -16.785498 27.08237
## Jul 2015       5.148434 -10.3424731 20.63934 -18.542864 28.83973
## Aug 2015       5.148434 -11.4120126 21.70888 -20.178583 30.47545
## Sep 2015       5.148434 -12.4165478 22.71342 -21.714887 32.01175
## Oct 2015       5.148434 -13.3666619 23.66353 -23.167961 33.46483
## Nov 2015       5.148434 -14.2703447 24.56721 -24.550025 34.84689
## Dec 2015       5.148434 -15.1338035 25.43067 -25.870571 36.16744
## Jan 2016       5.148434 -15.9619745 26.25884 -27.137150 37.43402
## Feb 2016       5.148434 -16.7588602 27.05573 -28.355881 38.65275
## Mar 2016       5.148434 -17.5277590 27.82463 -29.531810 39.82868
## Apr 2016       5.148434 -18.2714277 28.56830 -30.669153 40.96602
## May 2016       5.148434 -18.9921980 29.28907 -31.771476 42.06834
## Jun 2016       5.148434 -19.6920633 29.98893 -32.841828 43.13870
## Jul 2016       5.148434 -20.3727434 30.66961 -33.882838 44.17971
## Aug 2016       5.148434 -21.0357346 31.33260 -34.896796 45.19366
```

```
## Sep 2016       5.148434 -21.6823483 31.97922 -35.885706 46.18257
## Oct 2016       5.148434 -22.3137412 32.61061 -36.851338 47.14821
## Nov 2016       5.148434 -22.9309403 33.22781 -37.795263 48.09213
## Dec 2016       5.148434 -23.5348617 33.83173 -38.718881 49.01575
```

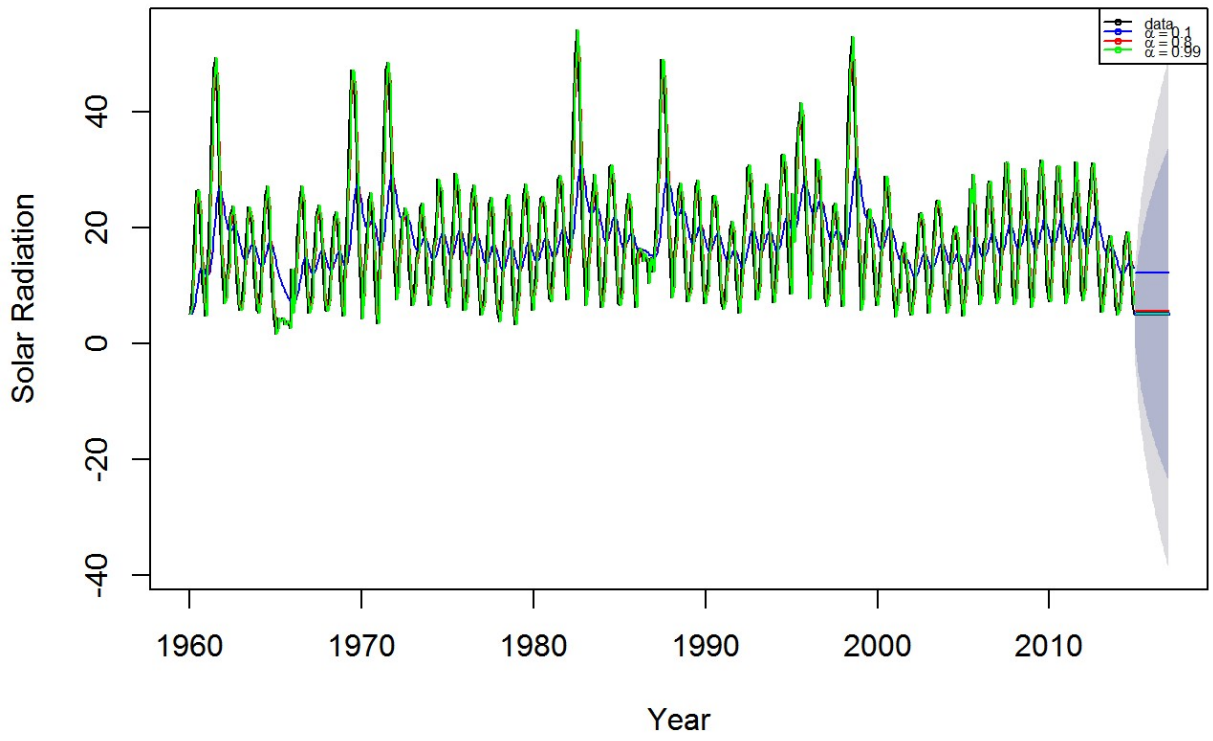
Interpretation

The output of ses() function mainly shows the measures of model fit and the forecasts.

For the error measures model with alpha chosen by the software (=0.9999) gives better results

than the one with alpha=0.1 or 0.8

```
plot(fit3, plot.conf=FALSE, ylab="Solar Radiation", xlab="Year", main="", type
="l")
lines(fitted(fit1), col="blue", type="l")
lines(fitted(fit2), col="red", type="l")
lines(fitted(fit3), col="green", type="l")
lines(fit1$mean, col="blue", type="l")
lines(fit2$mean, col="red", type="l")
lines(fit3$mean, col="green", type="l")
legend("topright",lty=1, col=c(1,"blue","red","green"),  c("data", expression(a
lpha == 0.1), expression(alpha == 0.8), expression(alpha == 0.99)),pch=1, cex=
0.5, y.intersp=0.5)
```



Interpretation from simple exponential smoothening models and forecasts ## For the lower value of alpha the curve becomes more smooth. If value of alpha is increased then the curves ## follow the fluctuations of the series. ## The confidence interval is too large to predict the accurate forecasts and neither trend nor seasonality is captured.

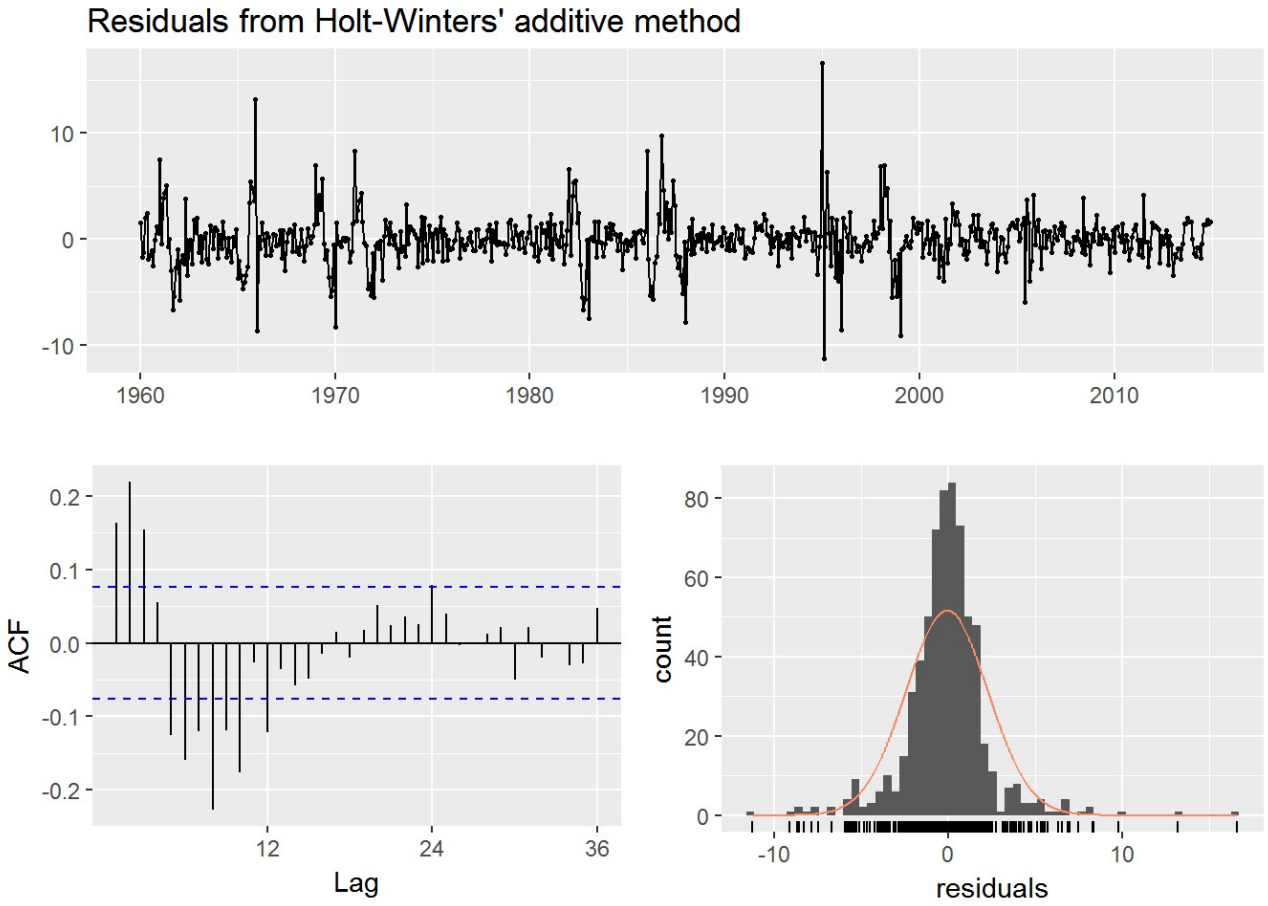
Holt Winter's methods of trend and seasonality

```
fit_1 <- hw(solar.ts,seasonal="additive", h=2*frequency(solar.ts)) #Additive Tr
end, Additive Seasonality
summary(fit_1) # Holt-Winters' additive method (AA)
```

```
##
## Forecast method: Holt-Winters' additive method
##
## Model Information:
## Holt-Winters' additive method
##
## Call:
##  hw(y = solar.ts, h = 2 * frequency(solar.ts), seasonal = "additive")
##
##   Smoothing parameters:
##     alpha = 0.9968
##     beta  = 0.0079
##     gamma = 0.0027
##
##   Initial states:
##     l = 12.813
##     b = 0.4276
##     s=-10.6349 -7.3748 -2.6593 2.7233 7.775 11.0058
##            9.8199 6.1144 1.8544 -1.8065 -7.0856 -9.7316
##
##   sigma:  2.3699
##
##      AIC     AICc      BIC
## 5457.817 5458.770 5534.185
##
## Error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.08375221 2.369864 1.547273 -1.615444 12.99165 0.2541887
##                  ACF1
## Training set 0.163735
##
## Forecasts:
##          Point Forecast      Lo 80     Hi 80       Lo 95    Hi 95
## Jan 2015       5.899303   2.862201  8.936406   1.2544557 10.54415
## Feb 2015       8.536199   4.230998 12.841399   1.9519623 15.12043
## Mar 2015      13.828280   8.537485 19.119075   5.7367070 21.91985
## Apr 2015      17.502130  11.370374 23.633886   8.1244191 26.87984
## May 2015      21.822830  14.941431 28.704228  11.2986391 32.34702
## Jun 2015      25.314433  17.747444 32.881421  13.7417227 36.88714
## Jul 2015      26.552786  18.348117 34.757454  14.0048274 39.10074
## Aug 2015      23.394989  14.590069 32.199910   9.9290252 36.86095
## Sep 2015      18.270816   8.895819 27.645813   3.9329954 32.60864
## Oct 2015      12.811417   2.891257 22.731577  -2.3601587 27.98299
## Nov 2015       8.147760  -2.296607 18.592126  -7.8255203 24.12104
## Dec 2015       5.037795  -5.912884 15.988474 -11.7098227 21.78541
## Jan 2016       5.789632  -5.654269 17.233532 -11.7123037 23.29157
## Feb 2016       8.426527  -3.494615 20.347668  -9.8052858 26.65834
## Mar 2016      13.718608   1.332116 26.105100  -5.2248969 32.66211
```

```
## Apr 2016       17.392458   4.551168 30.233748  -2.2466003 37.03152
## May 2016       21.713158   8.426495 34.999820   1.3929612 42.03335
## Jun 2016       25.204761  11.481193 38.928329   4.2163742 46.19315
## Jul 2016       26.443114  12.290280 40.595947   4.7982231 48.08800
## Aug 2016       23.285317   8.710146 37.860488   0.9945162 45.57612
## Sep 2016       18.161144   3.169938 33.152351  -4.7659279 41.08822
## Oct 2016       12.701745  -2.699742 28.103232 -10.8527973 36.25629
## Nov 2016        8.038088  -7.768410 23.844585 -16.1358645 32.21204
## Dec 2016        4.928123 -11.278545 21.134792 -19.8578375 29.71408
```

```
checkresiduals(fit_1)
```

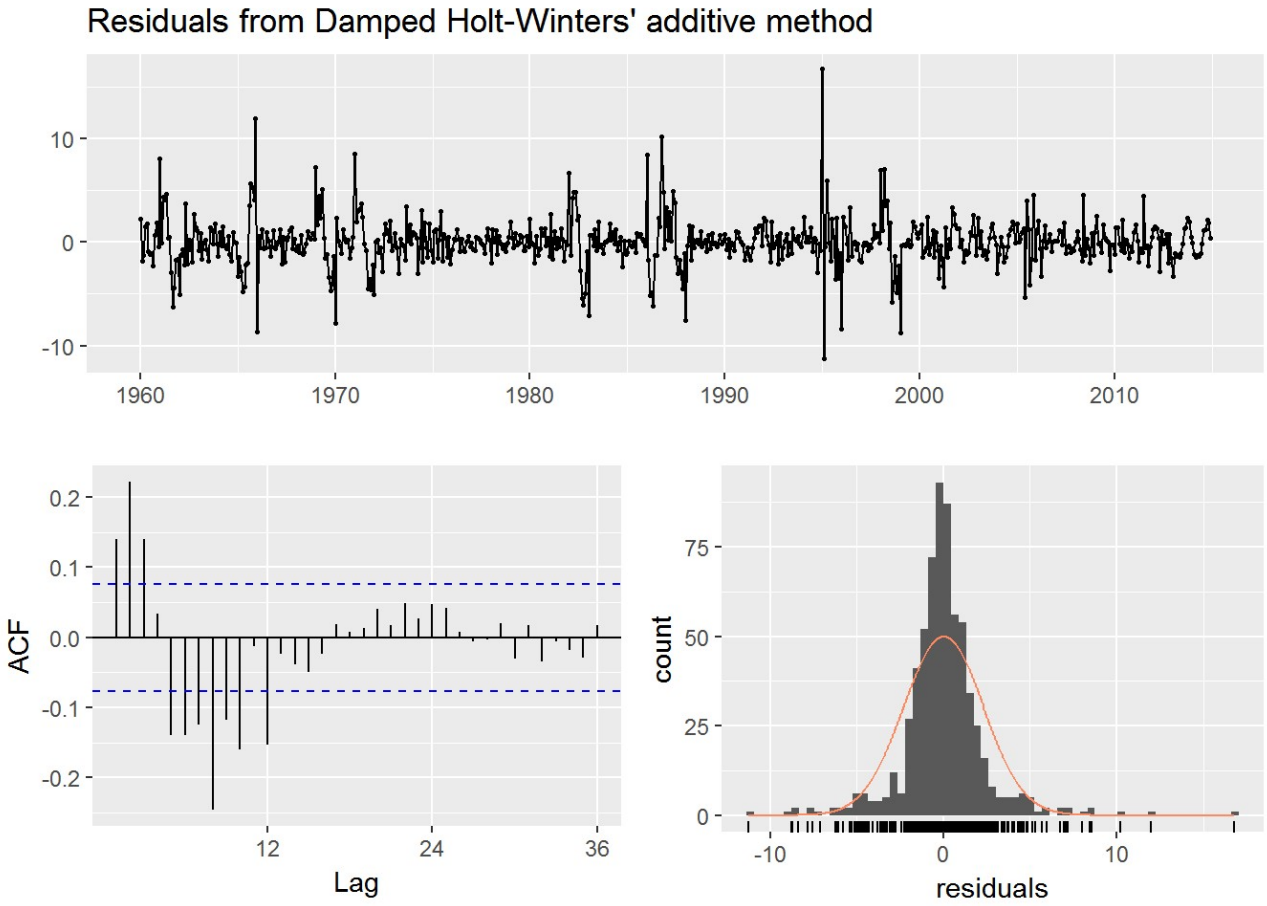



```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' additive method
## Q* = 193.75, df = 8, p-value < 2.2e-16
##
## Model df: 16.   Total lags used: 24
```

```
# Additive Damped Trend and Additive Seasonality
fit_2 <- hw(solar.ts,seasonal="additive",damped = TRUE, h=2*frequency(solar.t
s))
summary(fit2) # Damped Holt-Winters' additive method (AdA)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = solar.ts, h = 24, initial = "simple", alpha = 0.8)
##
##   Smoothing parameters:
##     alpha = 0.8
##
##   Initial states:
##     l = 5.0517
##
##   sigma:  5.3164
## Error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.001073712 5.316378 4.574236 -8.172062 32.43497 0.7514636
##                    ACF1
## Training set 0.7536451
##
## Forecasts:
##          Point Forecast      Lo 80    Hi 80      Lo 95    Hi 95
## Jan 2015       5.618649  -1.194563 12.43186  -4.801260 16.03856
## Feb 2015       5.618649  -3.106520 14.34382  -7.725346 18.96264
## Mar 2015       5.618649  -4.669076 15.90637 -10.115069 21.35237
## Apr 2015       5.618649  -6.023773 17.26107 -12.186900 23.42420
## May 2015       5.618649  -7.236494 18.47379 -14.041596 25.27889
## Jun 2015       5.618649  -8.344281 19.58158 -15.735810 26.97311
## Jul 2015       5.618649  -9.370418 20.60772 -17.305151 28.54245
## Aug 2015       5.618649 -10.330672 21.56797 -18.773733 30.01103
## Sep 2015       5.618649 -11.236308 22.47361 -20.158783 31.39608
## Oct 2015       5.618649 -12.095703 23.33300 -21.473115 32.71041
## Nov 2015       5.618649 -12.915292 24.15259 -22.726568 33.96387
## Dec 2015       5.618649 -13.700142 24.93744 -23.926893 35.16419
## Jan 2016       5.618649 -14.454328 25.69163 -25.080320 36.31762
## Feb 2016       5.618649 -15.181186 26.41848 -26.191953 37.42925
## Mar 2016       5.618649 -15.883487 27.12078 -27.266030 38.50333
## Apr 2016       5.618649 -16.563563 27.80086 -28.306117 39.54342
## May 2016       5.618649 -17.223401 28.46070 -29.315252 40.55255
## Jun 2016       5.618649 -17.864706 29.10200 -30.296044 41.53334
## Jul 2016       5.618649 -18.488957 29.72626 -31.250753 42.48805
## Aug 2016       5.618649 -19.097447 30.33475 -32.181358 43.41866
## Sep 2016       5.618649 -19.691312 30.92861 -33.089596 44.32689
## Oct 2016       5.618649 -20.271558 31.50886 -33.977006 45.21430
## Nov 2016       5.618649 -20.839082 32.07638 -34.844959 46.08226
## Dec 2016       5.618649 -21.394686 32.63198 -35.694681 46.93198
```

```
checkresiduals(fit_2)
```



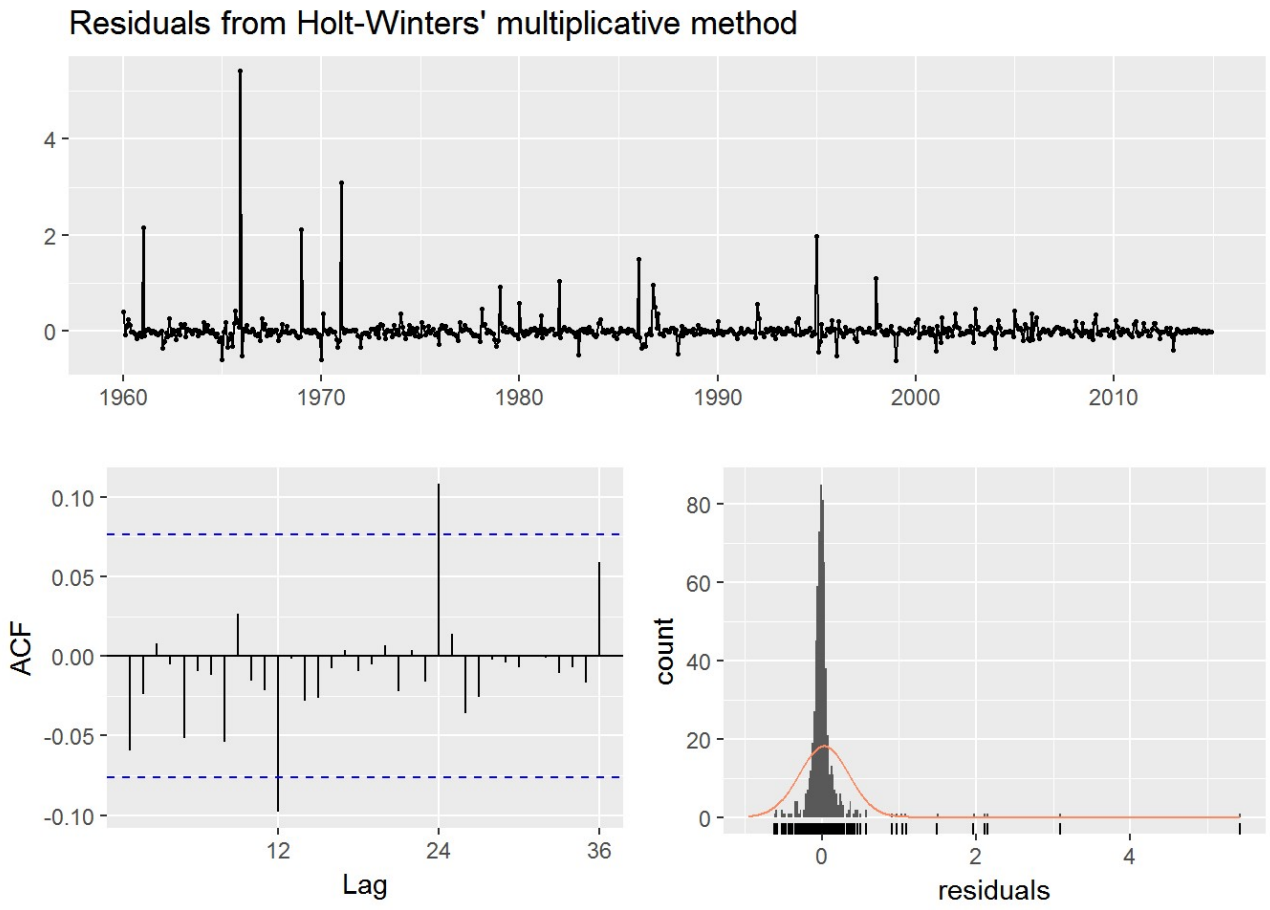

```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt-Winters' additive method
## Q* = 187.39, df = 7, p-value < 2.2e-16
##
## Model df: 17.   Total lags used: 24
```

```
# Additive Trend and Multiplicative Seasonality
fit_3 <- hw(solar.ts,seasonal="multiplicative", h=2*frequency(solar.ts))
summary(fit_3) # Holt-Winters' multiplicative method (AM)
```

```
##
## Forecast method: Holt-Winters' multiplicative method
##
## Model Information:
## Holt-Winters' multiplicative method
##
## Call:
##  hw(y = solar.ts, h = 2 * frequency(solar.ts), seasonal = "multiplicative")
##
##   Smoothing parameters:
##     alpha = 0.9181
##     beta  = 1e-04
##     gamma = 0.0155
##
##   Initial states:
##     l = 9.0986
##     b = 0.0427
##     s=0.4397 0.5864 0.8389 1.1545 1.4509 1.62
##            1.5856 1.4029 1.0993 0.8686 0.5587 0.3944
##
##   sigma:  0.3238
##
##      AIC     AICc      BIC
## 6420.503 6421.456 6496.871
##
## Error measures:
##                       ME     RMSE      MAE      MPE     MAPE      MASE
## Training set -0.1060967 2.062279 1.255284 -2.17078 10.01439 0.2062203
##                      ACF1
## Training set -0.07132262
##
## Forecasts:
##          Point Forecast       Lo 80     Hi 80       Lo 95     Hi 95
## Jan 2015       5.608518   3.2813132  7.935723   2.0493654  9.167671
## Feb 2015       6.942511   2.9455771 10.939445   0.8297282 13.055293
## Mar 2015      10.231515   2.9702619 17.492769  -0.8736134 21.336644
## Apr 2015      12.735791   2.1575755 23.314007  -3.4421936 28.913776
## May 2015      16.245849   0.9089146 31.582784  -7.2099681 39.701667
## Jun 2015      18.528840  -0.9834884 38.041169 -11.3126913 48.370372
## Jul 2015      19.215693  -3.0649048 41.496291 -14.8595407 53.290927
## Aug 2015      17.268548  -4.5683751 39.105472 -16.1281441 50.665241
## Sep 2015      13.834786  -5.1066311 32.776204 -15.1336119 42.803185
## Oct 2015       9.939120  -4.7099408 24.588180 -12.4646849 32.342924
## Nov 2015       6.823089  -3.9531865 17.599364  -9.6578022 23.303980
## Dec 2015       5.368708  -3.6833678 14.420784  -8.4752472 19.212663
## Jan 2016       5.847459  -4.6648771 16.359795 -10.2297716 21.924689
## Feb 2016       7.237310  -6.5718178 21.046437 -13.8819283 28.356547
## Mar 2016      10.664548 -10.8839742 32.213071 -22.2910729 43.620169
```

```
## Apr 2016      13.273050 -15.0733145 41.619415 -30.0789736 56.625074
## May 2016      16.928946 -21.2202792 55.078171 -41.4152591 75.273151
## Jun 2016      19.305400 -26.5334907 65.144291 -50.7991337 89.409934
## Jul 2016      20.018432 -30.0003163 70.037181 -56.4786424 96.515507
## Aug 2016      17.987618 -29.2553832 65.230620 -54.2643175 90.239554
## Sep 2016      14.409021 -25.3310708 54.149113 -46.3682047 75.186247
## Oct 2016      10.350337 -19.5996717 40.300345 -35.4542484 56.154922
## Nov 2016       7.104483 -14.4472403 28.656207 -25.8560336 40.065000
## Dec 2016       5.589417 -12.1736547 23.352489 -21.5768568 32.755691
```

```
checkresiduals(fit_3)
```



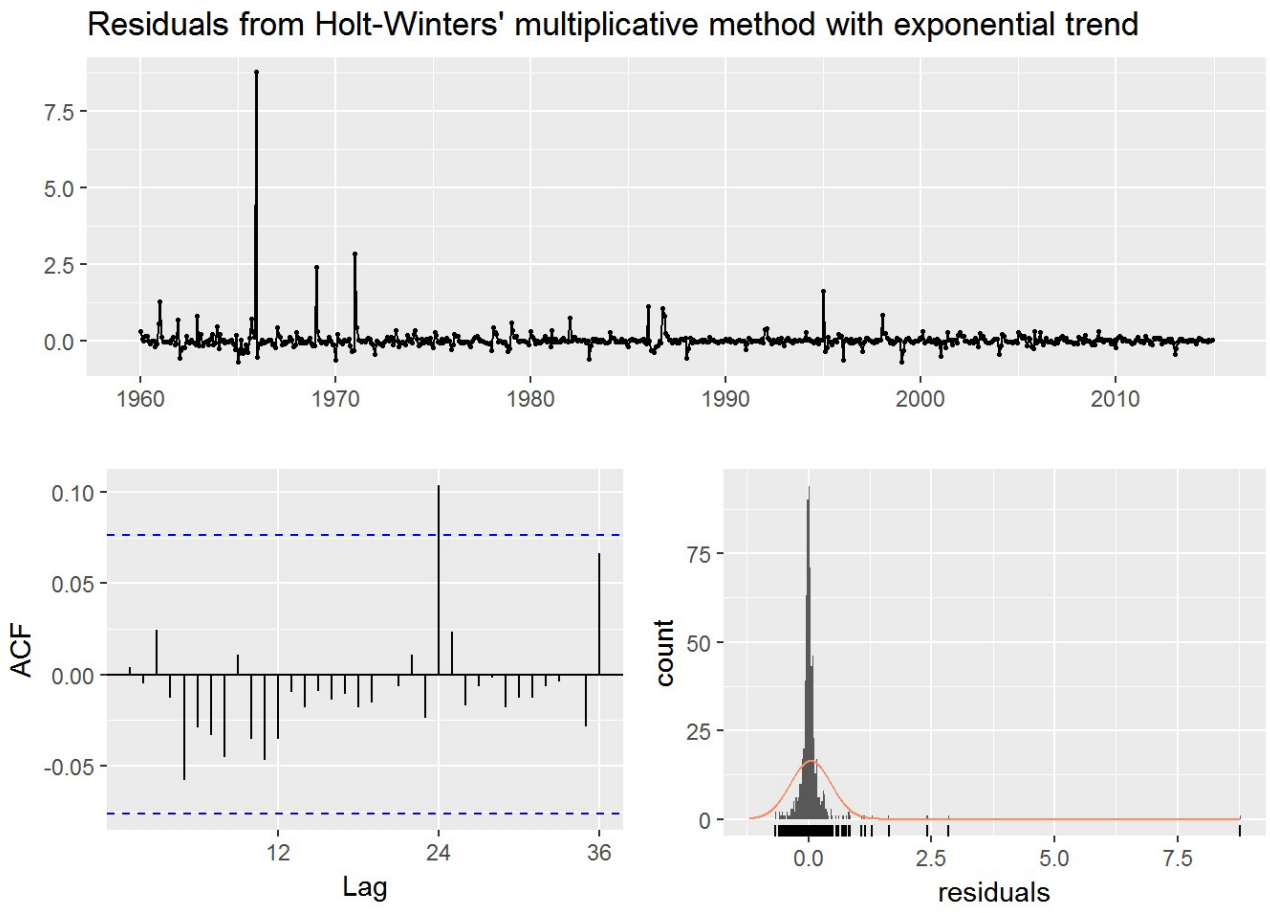

```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' multiplicative method
## Q* = 23.735, df = 8, p-value = 0.002539
##
## Model df: 16.   Total lags used: 24
```

```
# Multiplicative Trend and Multiplicative Seasonality
fit_4 <- hw(solar.ts,seasonal="multiplicative",exponential = TRUE, h=2*frequenc
y(solar.ts))
summary(fit_4) # Holt-Winters' multiplicative method with exponential trend (M
M)
```

```
##
## Forecast method: Holt-Winters' multiplicative method with exponential trend
##
## Model Information:
## Holt-Winters' multiplicative method with exponential trend
##
## Call:
##  hw(y = solar.ts, h = 2 * frequency(solar.ts), seasonal = "multiplicativ
e",
##
##  Call:
##      exponential = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.6907
##     beta  = 1e-04
##     gamma = 0.0685
##
##   Initial states:
##     l = 10.5634
##     b = 0.9838
##     s=0.259 0.6185 0.856 1.0368 1.5259 1.832
##            1.6442 1.4562 1.1104 0.8209 0.4742 0.366
##
##   sigma:  0.4177
##
##      AIC     AICc      BIC
## 6733.816 6734.769 6810.184
##
## Error measures:
##                      ME     RMSE      MAE        MPE    MAPE      MASE
## Training set 0.2010306 2.222834 1.403495 -0.8975766 11.4617 0.2305686
##                   ACF1
## Training set 0.132215
##
## Forecasts:
##          Point Forecast     Lo 80     Hi 80      Lo 95    Hi 95
## Jan 2015       5.821598 2.7546663  8.951708 1.22143848 10.64502
## Feb 2015       7.683304 2.9626202 12.839435 1.33432044 16.83335
## Mar 2015      11.028436 3.5505662 20.206029 1.35296588 26.93775
## Apr 2015      12.893378 3.6467849 24.137764 1.46985834 35.62301
## May 2015      15.318059 3.6832637 30.150129 1.31155523 45.52258
## Jun 2015      16.972008 3.5258993 34.743421 1.33310841 55.35626
## Jul 2015      17.507690 3.3251284 36.779461 1.04520552 60.90281
## Aug 2015      15.624948 2.6415436 34.729629 0.86804110 58.99137
## Sep 2015      12.509490 1.8918628 28.552455 0.54116726 50.25118
## Oct 2015       8.804528 1.1582386 20.382141 0.30153145 37.79444
## Nov 2015       5.965868 0.7444946 14.250153 0.22018413 26.99639
```

```
## Dec 2015       4.378617 0.4877842 10.630240 0.14585795 21.40814
## Jan 2016       5.014058 0.4953414 12.257639 0.14656591 25.11738
## Feb 2016       6.617518 0.5906621 16.600568 0.16480183 34.16288
## Mar 2016       9.498632 0.7498657 23.712632 0.20503424 57.00495
## Apr 2016      11.104880 0.8037919 27.786378 0.22732415 65.70746
## May 2016      13.193222 0.8623503 33.221668 0.22682622 79.48224
## Jun 2016      14.617744 0.9263471 37.304895 0.23542906 84.21618
## Jul 2016      15.079120 0.8354508 38.885053 0.18173500 87.57102
## Aug 2016      13.457541 0.6950527 32.971662 0.16557280 81.73750
## Sep 2016      10.774242 0.5081952 26.690201 0.13258664 63.90468
## Oct 2016       7.583212 0.3213547 18.836130 0.07171536 46.97492
## Nov 2016       5.138315 0.2005578 12.569953 0.04466555 31.85355
## Dec 2016       3.771239 0.1367125  9.169251 0.02912552 22.38691
```

```
checkresiduals(fit_4)
```



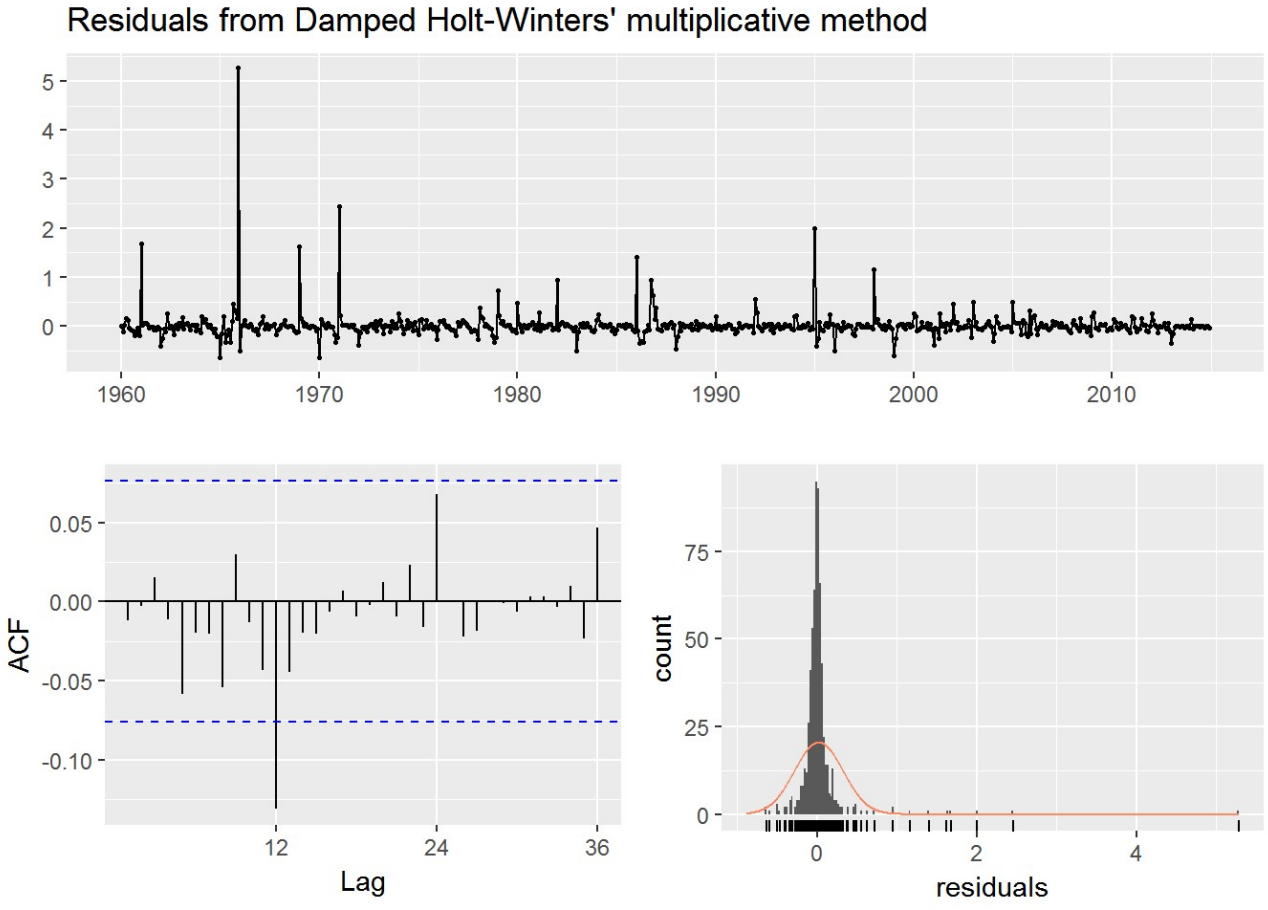
```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' multiplicative method with exponential t
rend
## Q* = 17.483, df = 8, p-value = 0.02545
##
## Model df: 16.   Total lags used: 24
```

```
# Multiplicative Trend and Additive damped trend
fit_5 <- hw(solar.ts,seasonal="multiplicative", damped=TRUE, h=2*frequency(sola
r.ts))
summary(fit_5) # Damped Holt-Winters' multiplicative method (AdM)
```

```
##
## Forecast method: Damped Holt-Winters' multiplicative method
##
## Model Information:
## Damped Holt-Winters' multiplicative method
##
## Call:
##  hw(y = solar.ts, h = 2 * frequency(solar.ts), seasonal = "multiplicativ
e",
##
##  Call:
##      damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.7951
##     beta  = 4e-04
##     gamma = 1e-04
##     phi   = 0.8805
##
##   Initial states:
##     l = 9.9766
##     b = 1.3498
##     s=0.4466 0.5613 0.8259 1.1521 1.4418 1.6021
##            1.549 1.393 1.1118 0.8795 0.5884 0.4484
##
##   sigma:  0.3011
##
##      AIC     AICc      BIC
## 6327.138 6328.205 6407.999
##
## Error measures:
##                        ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.03547783 2.039583 1.240267 -2.200423 10.02395 0.2037532
##                     ACF1
## Training set 0.05899635
##
## Forecasts:
##          Point Forecast        Lo 80     Hi 80       Lo 95     Hi 95
## Jan 2015       5.210042   3.19992817  7.220156   2.1358381  8.284246
## Feb 2015       6.829713   3.40403682 10.255390   1.5905933 12.068833
## Mar 2015      10.207852   4.08059108 16.335112   0.8370152 19.578688
## Apr 2015      12.899256   4.00597381 21.792538  -0.7018454 26.500357
## May 2015      16.163790   3.67436373 28.653216  -2.9371391 35.264719
## Jun 2015      17.974152   2.66137944 33.286925  -5.4447128 41.393017
## Jul 2015      18.588747   1.33021801 35.847275  -7.8058952 44.983388
## Aug 2015      16.729045  -0.05085655 33.508946  -8.9335997 42.391689
## Sep 2015      13.367488  -1.02047080 27.755447  -8.6369962 35.371972
## Oct 2015       9.582696  -1.42585953 20.591251  -7.2534366 26.418828
```

```
## Nov 2015       6.511184  -1.43743113 14.459799  -5.6451738 18.667542
## Dec 2015       5.182438  -1.51602944 11.880905  -5.0619837 15.426859
## Jan 2016       5.209234  -1.89803200 12.316499  -5.6603911 16.078858
## Feb 2016       6.828787  -2.98013876 16.637712  -8.1726702 21.830244
## Mar 2016      10.206643  -5.19389071 25.607176 -13.3464407 33.759726
## Apr 2016      12.897924  -7.50542324 33.301271 -18.3063028 44.102150
## May 2016      16.162336 -10.59690033 42.921573 -24.7623846 57.087057
## Jun 2016      17.972747 -13.12429372 49.069788 -29.5860728 65.531567
## Jul 2016      18.587486 -14.97709822 52.152070 -32.7451158 69.920088
## Aug 2016      16.728063 -14.76001179 48.216137 -31.4287916 64.884917
## Sep 2016      13.366811 -12.83345715 39.567079 -26.7030412 53.436663
## Oct 2016       9.582278  -9.95698803 29.121543 -20.3004505 39.465006
## Nov 2016       6.510941  -7.28874491 20.310626 -14.5938572 27.615738
## Dec 2016       5.182272  -6.22524971 16.589794 -12.2640271 22.628572
```

```
# Lowest MASE and acceptable residual plot
checkresiduals(fit_5)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from Damped Holt-Winters' multiplicative method
## Q* = 24.591, df = 7, p-value = 0.0008964
##
## Model df: 17.   Total lags used: 24
```

Interpretation from holt winter's methods

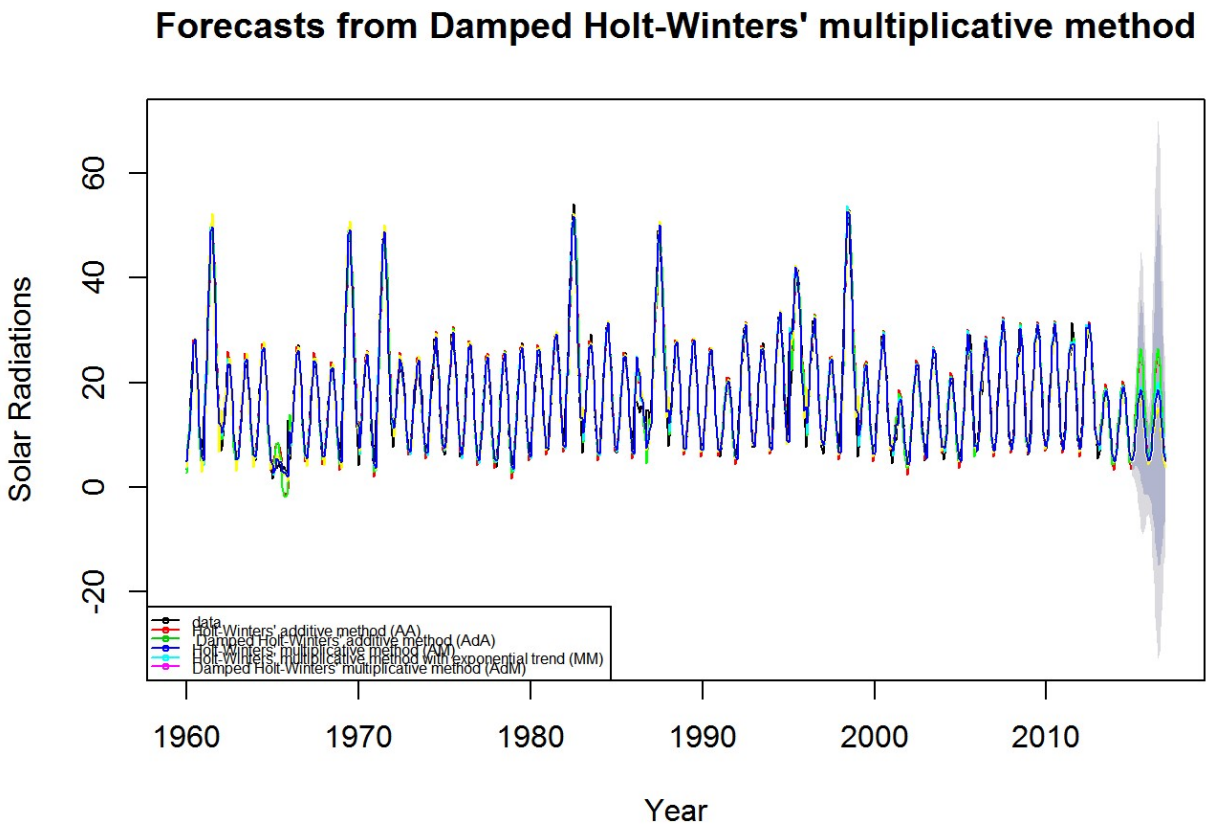
Holt- Winter's multiplicative method (fit_5) gives the smallest MASE value within the set of considered

models. And also the serial correlation in residuals and the variance in time series plot has tremendously

decreased so it seems to be a good model.

The following code chunk displays the fitted models, observed series and forecasts for 2 years ahead.

```
plot(fit_5, ylab="Solar Radiations",
     plot.conf=FALSE, type="l", fcol="white", xlab="Year")
lines(fitted(fit_1), col="red", lty=1)
lines(fitted(fit_2), col="green", lty=1)
lines(fitted(fit_3), col="cyan", lty=1)
lines(fitted(fit_4), col="yellow", lty=1)
lines(fitted(fit_5), col="blue", lty=1)
lines(fit_1$mean, type="l", col="red")
lines(fit_2$mean, type="l", col="green")
lines(fit_3$mean, type="l", col="cyan")
lines(fit_4$mean, type="l", col="yellow")
lines(fit_5$mean, type="l", col="blue")
legend("bottomleft",lty=1, pch=1, col=1:6,
       c("data","Holt-Winters' additive method (AA)", " Damped Holt-Winters' ad
ditive method (AdA)",
         "Holt-Winters' multiplicative method (AM)", "Holt-Winters' multiplicat
ive method with exponential trend (MM)",
         "Damped Holt-Winters' multiplicative method (AdM)"), cex=0.5, y.inters
p=0.5)
```



Interpretation

fit_3, fit_4, fit_5 are equally good models for forecasting. So we can go with the one with lowest MASE i.e.

fit_5

State Space Models

Let's fit Damped Holt-Winters' multiplicative method (AdM) with additive and multiplicative errors using

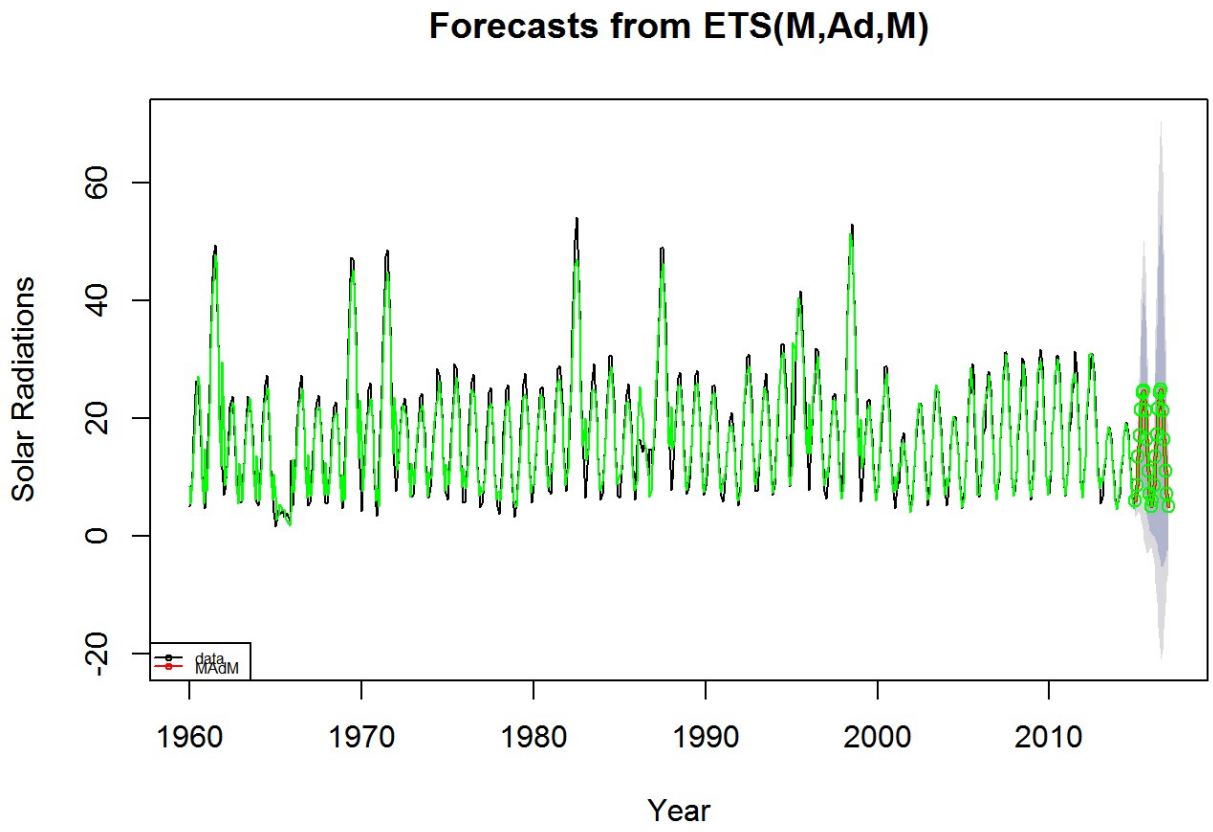
state space approach.

```
# AAdM Forbidden Model
fit.MAdM = ets(solar.ts, model = "MAM", damped = TRUE)
summary(fit.MAdM)
```

```
## ETS(M,Ad,M)
##
## Call:
##  ets(y = solar.ts, model = "MAM", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.7842
##     beta  = 1e-04
##     gamma = 0.0661
##     phi   = 0.9613
##
##   Initial states:
##     l = 10.4979
##     b = 0.7605
##     s=0.6918 0.3215 0.6002 1.001 1.3928 1.4728
##            1.4421 1.4614 1.2139 0.9745 0.6685 0.7595
##
##   sigma:  0.2294
##
##      AIC     AICc      BIC
## 5974.796 5975.863 6055.656
##
## Training set error measures:
##                      ME  RMSE      MAE       MPE     MAPE      MASE
## Training set 0.2739231 3.004 1.989601 -4.834858 17.12599 0.3268551
##                    ACF1
## Training set 0.2643485
```

Forecasts for MAdM model

```
frc.MAdM = forecast(fit.MAdM, h =24)
plot(frc.MAdM, ylab="Solar Radiations",plot.conf=FALSE, type="l", fcol="red", x
lab="Year")

lines(fitted(fit.MAdM), col="green", lty=1)
lines(frc.MAdM$mean,col="green", type="o")
legend("bottomleft",lty=1, pch=1, col=1:3, c("data","MAdM "), cex=0.5, y.inters
p=0.5)
```



Interpretation ## We get a good fit and forecasts from with the Damped Holt-Winters' multiplicative method with ## Multiplicative errors state space model (MAdM)

Auto Search

To trigger auto-search, we set the argument model to "ZZZ". We can choose the penalized measure to use for

auto selection using the argument ic=c ("aicc","aic","bic").

```
fit.auto.solar = ets(solar.ts, model="ZZZ",ic="bic") # Use BIC
fit.auto.solar$method # Holt Winters Additive damped trend with additive error
s is selected
```

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

```
fit.auto.solar = ets(solar.ts, model="ZZZ",ic="aic") # Use AIC
fit.auto.solar$method # Holt Winters Additive damped trend with additive error
s is selected
```

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

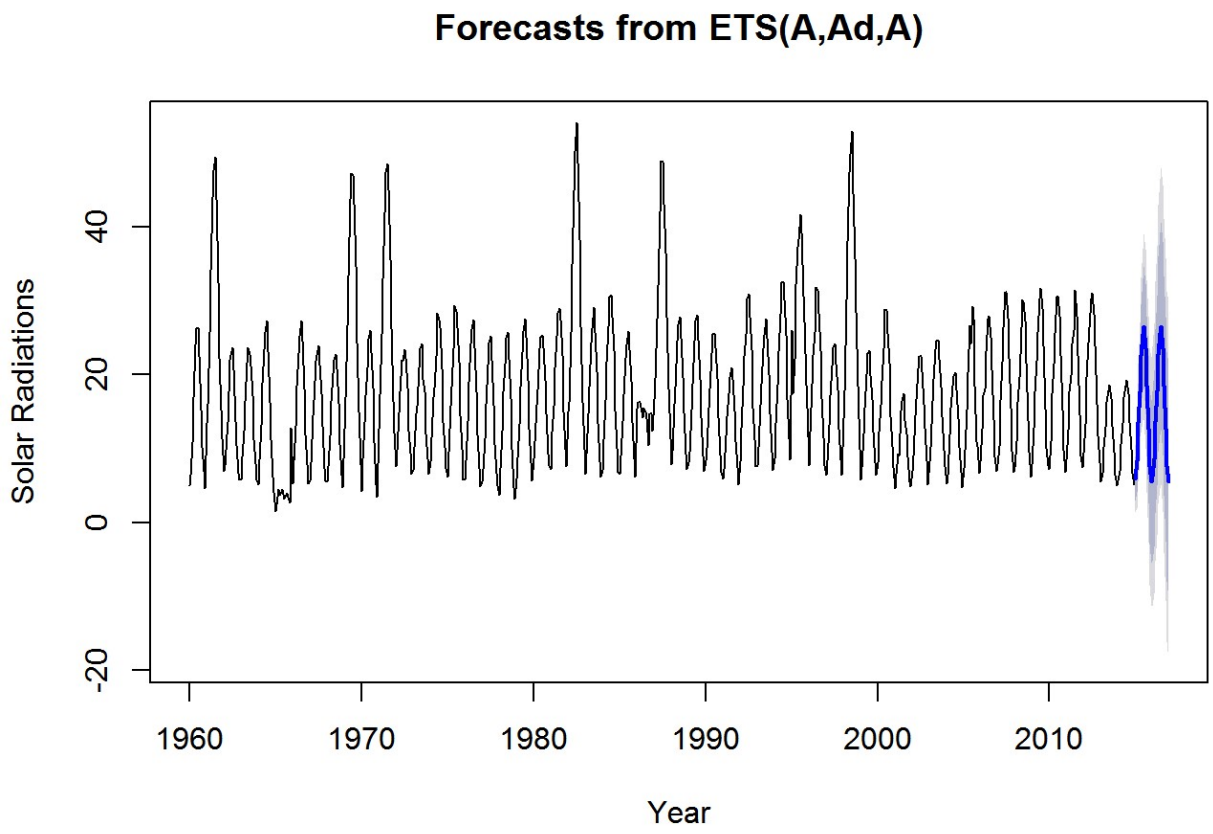
```
fit.auto.solar = ets(solar.ts,model="ZZZ",ic="aicc") # Use AICc
fit.auto.solar$method # Holt Winters Additive damped trend with additive error
s is selected
```

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

```
fit.auto.solar = ets(solar.ts,model="ZZZ") # Use the default
fit.auto.solar$method # Holt-Winters Additive damped trend with additive error
s is selected
```

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

```
plot(forecast(fit.auto.solar), ylab="Solar Radiations",plot.conf=FALSE, type
="l",
     xlab="Year")
```




```
#AAdA Best
```

Therefore, Based on auto-search (A,Ad,A) model is the best fit.

For time series that exhibit seasonal patterns, the local trend model can be augmented by seasonal effects.

Often the structure of the seasonal pattern changes over time in response to changes in

environment. For example, solar radiation used to peak in summers, but in some locations it now peaks in

winters too due to the global warming/ ozone depletion. Thus, the formulae used to represent the seasonal

effects should allow for the possibility of changing seasonal patterns. For this we consider Local ##Additive

Seasonal Models.

Local Additive Seasonal Models

```
fit.AAA = ets(solar.ts, model="AAA")
summary(fit.AAA) # Equally good
```

```
## ETS(A,Ad,A)
##
## Call:
##  ets(y = solar.ts, model = "AAA")
##
##   Smoothing parameters:
##     alpha = 0.9998
##     beta  = 0.0305
##     gamma = 1e-04
##     phi   = 0.8002
##
##   Initial states:
##     l = 11.3091
##     b = 1.1812
##     s=-10.2162 -8.1852 -3.0863 2.7434 7.8222 10.7833
##            9.7852 6.9704 2.0583 -2.0649 -7.1175 -9.4927
##
##   sigma:  2.3047
##
##      AIC     AICc      BIC
## 5423.009 5424.076 5503.869
##
## Training set error measures:
##                         ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.0003458727 2.304693 1.479661 -1.293116 12.22459 0.2430814
##                   ACF1
## Training set 0.139996
```

```
fit.AAA.damped = ets(solar.ts, model="AAA",damped = TRUE)
summary(fit.AAA.damped) # Equally good
```

```
## ETS(A,Ad,A)
##
## Call:
##  ets(y = solar.ts, model = "AAA", damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.9998
##     beta  = 0.0305
##     gamma = 1e-04
##     phi   = 0.8002
##
##   Initial states:
##     l = 11.3091
##     b = 1.1812
##     s=-10.2162 -8.1852 -3.0863 2.7434 7.8222 10.7833
##            9.7852 6.9704 2.0583 -2.0649 -7.1175 -9.4927
##
##   sigma:  2.3047
##
##      AIC     AICc      BIC
## 5423.009 5424.076 5503.869
##
## Training set error measures:
##                         ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.0003458727 2.304693 1.479661 -1.293116 12.22459 0.2430814
##                   ACF1
## Training set 0.139996
```

Interpretation

Both the model give the same result in terms of predictive performance. And therefore, are equally suitable.

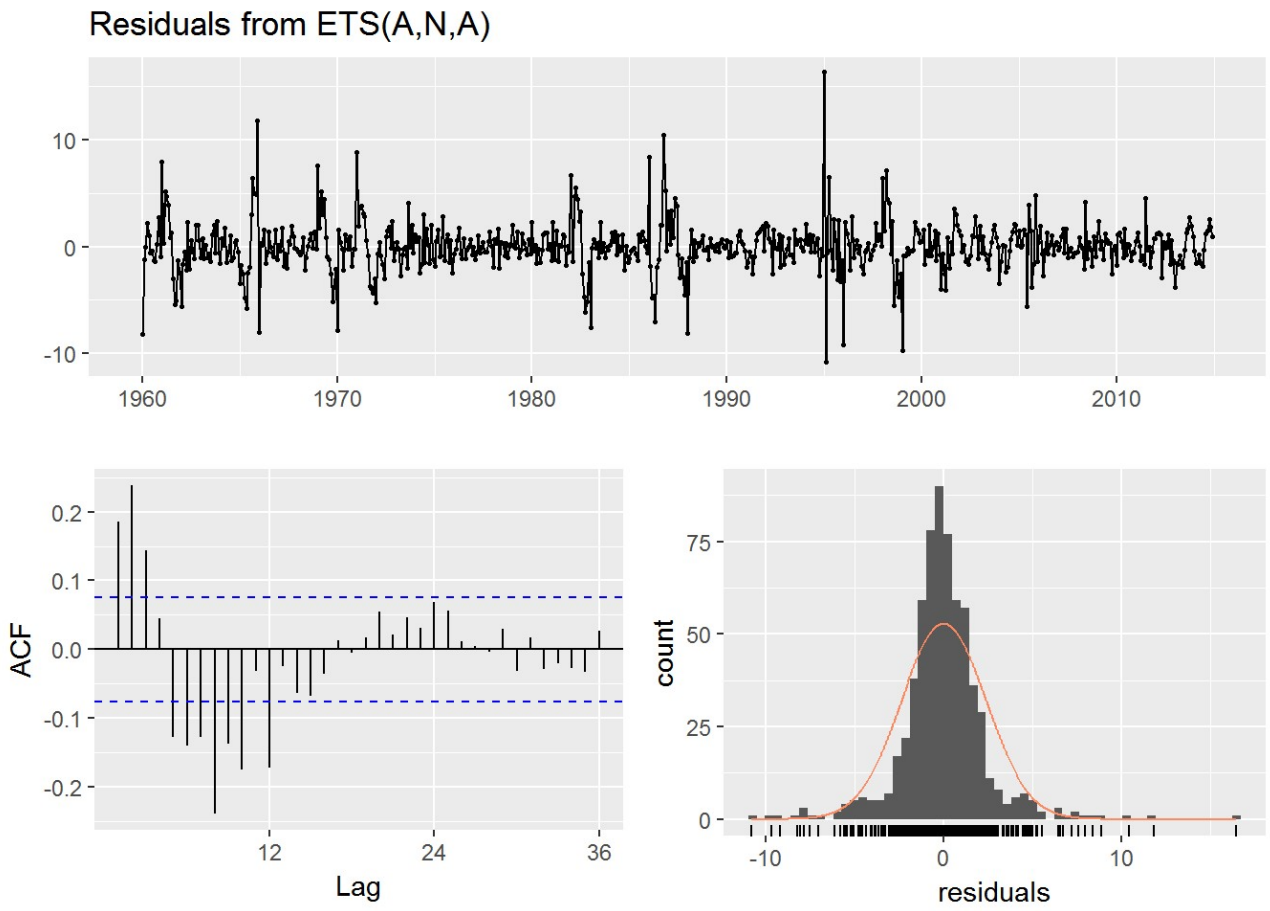
The following displays the forecast and fits from four candidate models.

Seasonal Models Based Only on Seasonal Levels

```
fit.ANA = ets(solar.ts, model="ANA")
summary(fit.ANA) # Lowest MASE, AICc, BIC, AIC
```

```
## ETS(A,N,A)
##
## Call:
##  ets(y = solar.ts, model = "ANA")
##
##   Smoothing parameters:
##     alpha = 0.9752
##     gamma = 0.0248
##
##   Initial states:
##     l = 22.4659
##     s=-10.2239 -8.9188 -3.0532 2.3506 8.2098 10.8414
##            10.4077 7.5173 1.4119 -2.4583 -6.8829 -9.2016
##
##   sigma:  2.3788
##
##      AIC     AICc      BIC
## 5458.759 5459.505 5526.143
##
## Training set error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.01014996 2.378754 1.557999 -1.880712 13.08487 0.2559508
##                   ACF1
## Training set 0.185752
```

```
checkresiduals(fit.ANA)
```

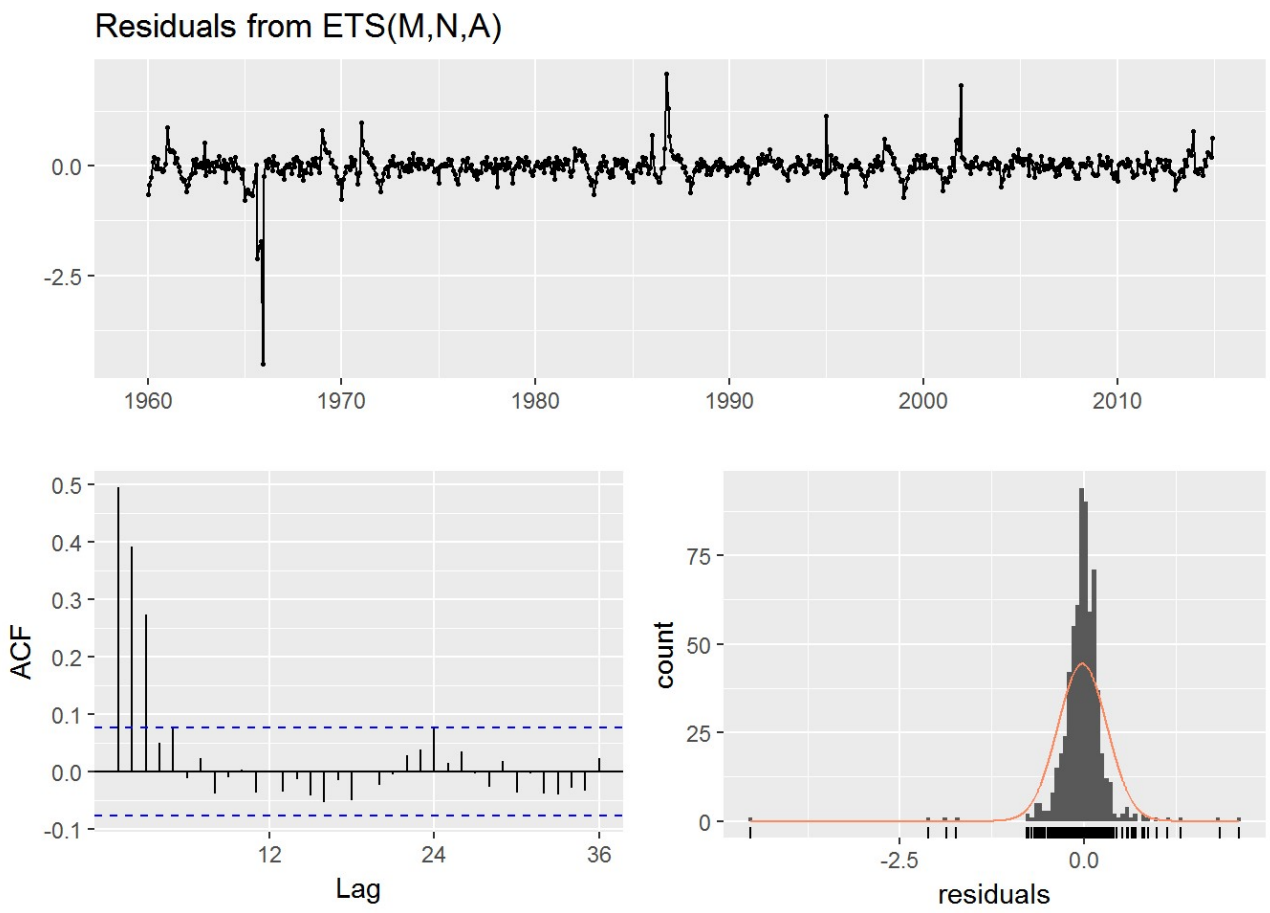



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,A)
## Q* = 217.68, df = 10, p-value < 2.2e-16
##
## Model df: 14.   Total lags used: 24
```

```
fit.MNA = ets(solar.ts, model= "MNA")
summary(fit.MNA)
```

```
## ETS(M,N,A)
##
## Call:
##  ets(y = solar.ts, model = "MNA")
##
##   Smoothing parameters:
##     alpha = 0.4777
##     gamma = 1e-04
##
##   Initial states:
##     l = 21.5697
##     s=-10.1753 -7.1745 -4.0165 0.0827 7.1147 7.8517
##            12.2277 6.0807 2.1198 -0.5072 -6.0681 -7.5357
##
##   sigma:  0.334
##
##      AIC     AICc      BIC
## 6496.630 6497.376 6564.014
##
## Training set error measures:
##                       ME   RMSE      MAE       MPE     MAPE      MASE
## Training set -0.02316152 3.6531 2.621824 -6.455377 20.94911 0.4307179
##                    ACF1
## Training set 0.4615006
```

```
checkresiduals(fit.MNA)
```

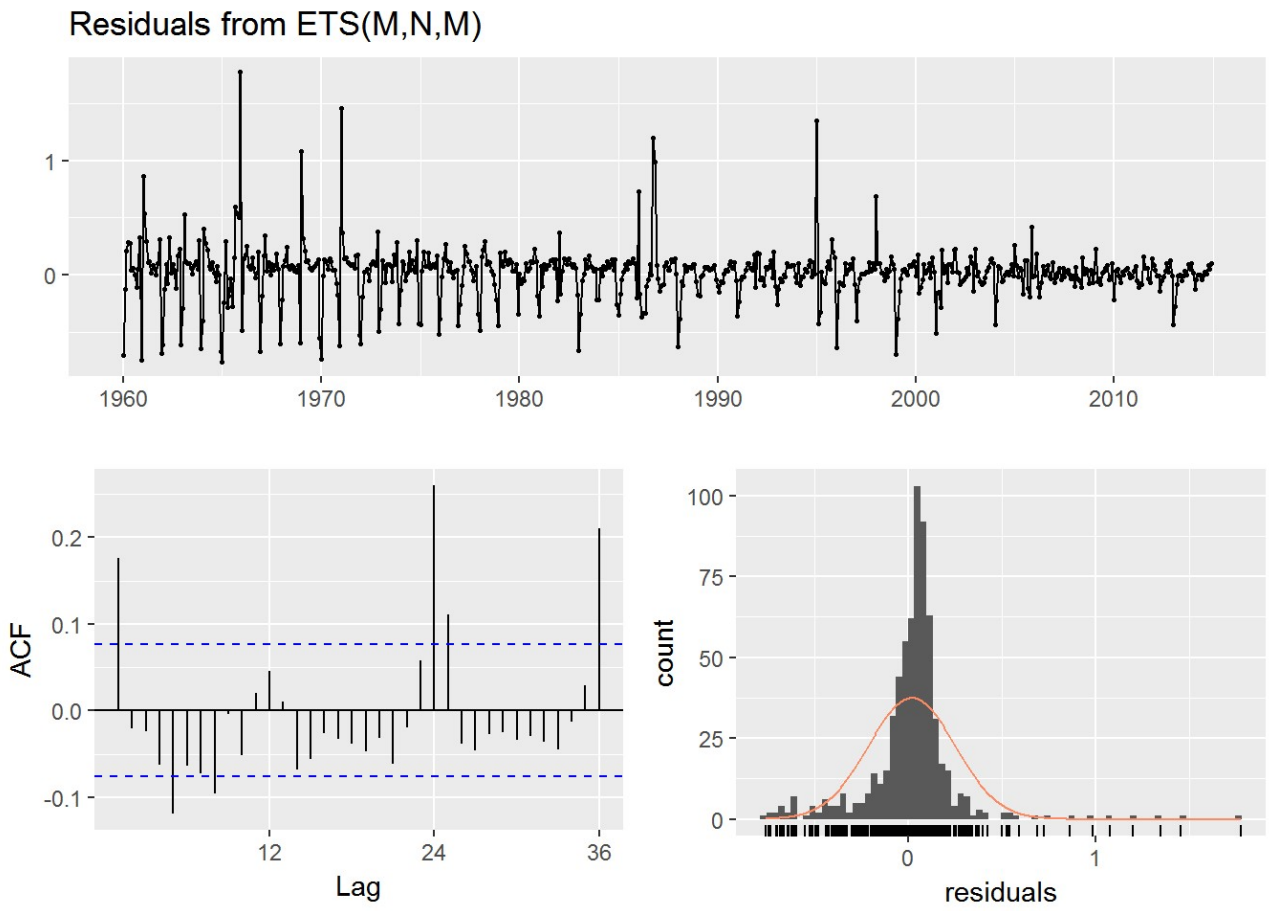



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,A)
## Q* = 335.01, df = 10, p-value < 2.2e-16
##
## Model df: 14.   Total lags used: 24
```

```
# ANM is forbidden model

fit.MNM = ets(solar.ts, model= "MNM")
summary(fit.MNM)
```

```
## ETS(M,N,M)
##
## Call:
##  ets(y = solar.ts, model = "MNM")
##
##   Smoothing parameters:
##     alpha = 0.7065
##     gamma = 0.0804
##
##   Initial states:
##     l = 21.5335
##     s=0.8906 0.3179 0.6025 0.9817 1.2849 1.4813
##            1.5419 1.375 1.1558 0.9043 0.6746 0.7896
##
##   sigma:  0.2323
##
##      AIC     AICc      BIC
## 5988.832 5989.577 6056.215
##
## Training set error measures:
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.2126627 3.195065 2.043712 -5.568916 17.95583 0.3357446
##                   ACF1
## Training set 0.2896291
```

```
checkresiduals(fit.MNM)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,M)
## Q* = 110.43, df = 10, p-value < 2.2e-16
##
## Model df: 14.   Total lags used: 24
```

Interpretation

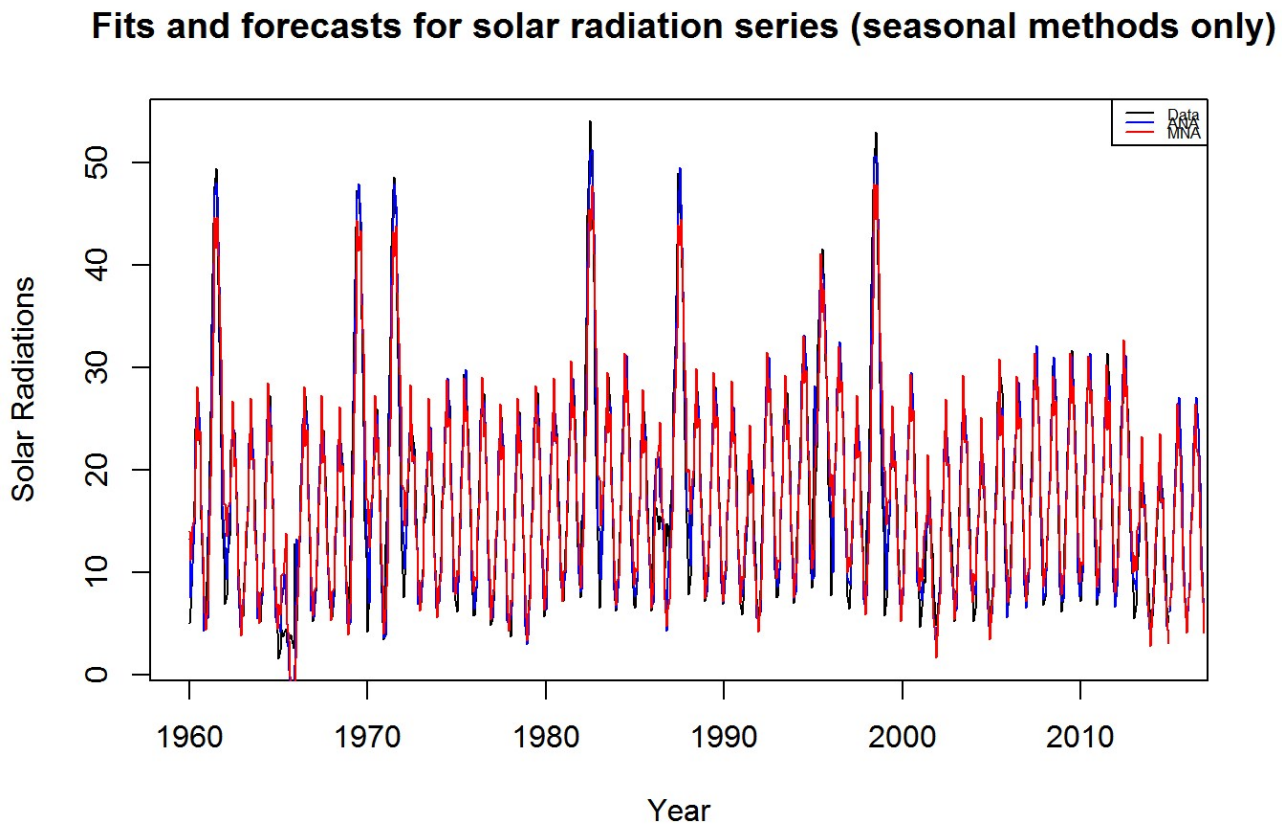
According to MASE, we get the best fit from ANA model.According to AIC, AICc and BIC values, the best fit

comes from the ANA model as well. However, according to the error diagnostics, MNA models is the best one

in terms of capturing the serial correlation structure in the series. But since the the goodness-of-fit

measures are quite different so we will go for the ANA model.

```
     plot(solar.ts, type="l", ylab="Solar Radiations", xlab="Year", main = " Fi
ts and forecasts for solar radiation series (seasonal methods only)")
lines(fitted(fit.ANA), col="blue",type="l")
lines(fitted(fit.MNA), col="red",type="l")
lines(forecast(fit.ANA)$mean, col="blue",type="l")
lines(forecast(fit.MNA)$mean, col="red",type="l")
legend("topright", lty=1, col=c("black","blue","red"),
       c("Data","ANA", "MNA"), cex=0.5, y.intersp=0.5)
```

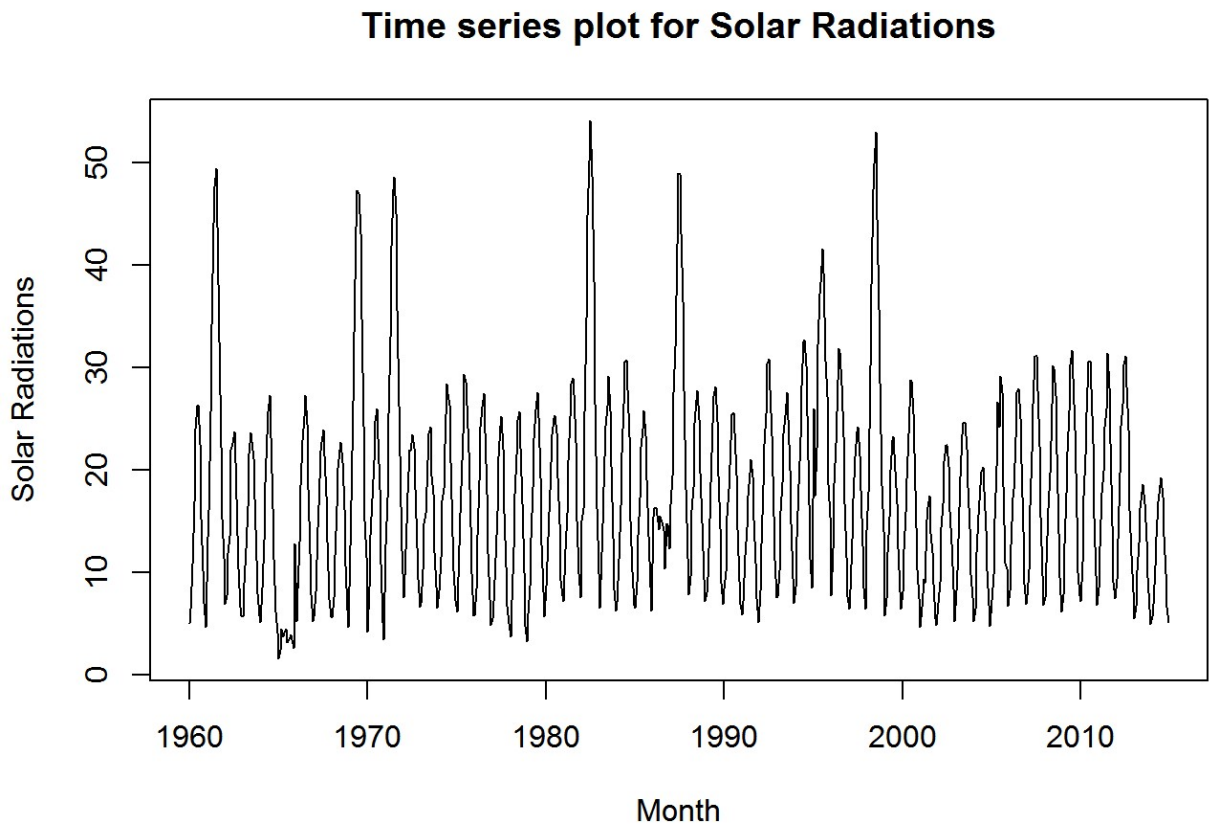


We get a better fit and forecasts from additive seasonality model with the additive errors

state space model.

Non Linear and Heteroscedastic State Space Models.

```
plot(solar.ts,ylab = "Solar Radiations", xlab="Month", main="Time series plot f
or Solar Radiations" )
```




```
fit.solar.AAA = ets(solar.ts, model="AAA")
summary(fit.solar.AAA) # Lowest MASE, AIC, AICc, BIC
```

```
## ETS(A,Ad,A)
##
## Call:
##  ets(y = solar.ts, model = "AAA")
##
##   Smoothing parameters:
##     alpha = 0.9998
##     beta  = 0.0305
##     gamma = 1e-04
##     phi   = 0.8002
##
##   Initial states:
##     l = 11.3091
##     b = 1.1812
##     s=-10.2162 -8.1852 -3.0863 2.7434 7.8222 10.7833
##            9.7852 6.9704 2.0583 -2.0649 -7.1175 -9.4927
##
##   sigma:  2.3047
##
##      AIC     AICc      BIC
## 5423.009 5424.076 5503.869
##
## Training set error measures:
##                         ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.0003458727 2.304693 1.479661 -1.293116 12.22459 0.2430814
##                   ACF1
## Training set 0.139996
```

```
fit.solar.MAA = ets(solar.ts, model="MAA")
summary(fit.solar.MAA)
```

```
## ETS(M,Ad,A)
##
## Call:
##  ets(y = solar.ts, model = "MAA")
##
##   Smoothing parameters:
##     alpha = 0.478
##     beta  = 8e-04
##     gamma = 1e-04
##     phi   = 0.8495
##
##   Initial states:
##     l = 10.7367
##     b = 2.9076
##     s=-10.3436 -7.8261 -3.4126 0.1089 7.7705 10.7246
##            9.8295 7.1223 2.5865 -2.0162 -6.9922 -7.5514
##
##   sigma:  0.335
##
##      AIC     AICc      BIC
## 6492.852 6493.919 6573.712
##
## Training set error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.04002889 3.335056 2.289546 -5.087836 19.54459 0.3761306
##                   ACF1
## Training set 0.6061329
```

```
fit.solar.MAM = ets(solar.ts, model="MAM")
summary(fit.solar.MAM)
```

```
## ETS(M,Ad,M)
##
## Call:
##  ets(y = solar.ts, model = "MAM")
##
##   Smoothing parameters:
##     alpha = 0.7842
##     beta  = 1e-04
##     gamma = 0.0661
##     phi   = 0.9613
##
##   Initial states:
##     l = 10.4979
##     b = 0.7605
##     s=0.6918 0.3215 0.6002 1.001 1.3928 1.4728
##            1.4421 1.4614 1.2139 0.9745 0.6685 0.7595
##
##   sigma:  0.2294
##
##      AIC     AICc      BIC
## 5974.796 5975.863 6055.656
##
## Training set error measures:
##                      ME  RMSE      MAE       MPE     MAPE      MASE
## Training set 0.2739231 3.004 1.989601 -4.834858 17.12599 0.3268551
##                    ACF1
## Training set 0.2643485
```

```
fit.solar.MMM = ets(solar.ts, model="MMM")
summary(fit.solar.MMM)
```

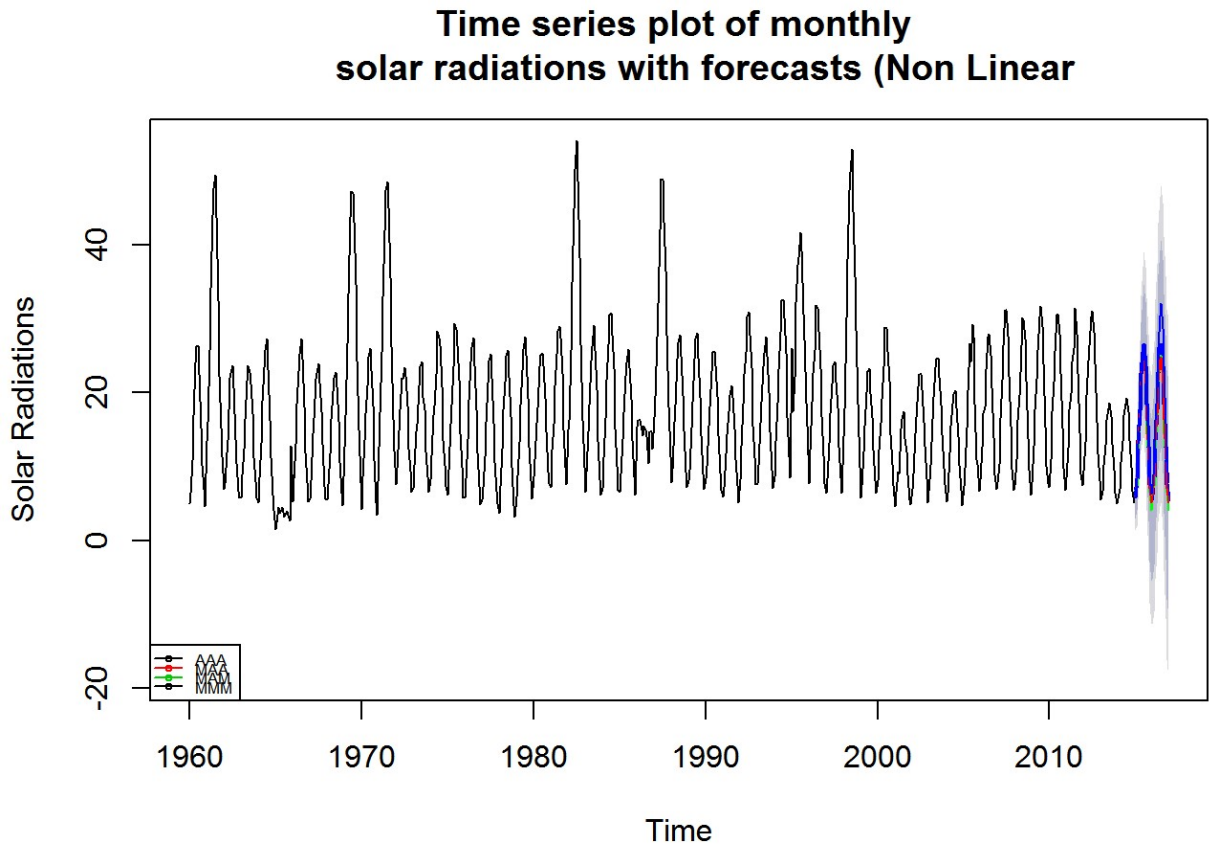
```
## ETS(M,M,M)
##
## Call:
##  ets(y = solar.ts, model = "MMM")
##
##   Smoothing parameters:
##     alpha = 0.7228
##     beta  = 8e-04
##     gamma = 0.0867
##
##   Initial states:
##     l = 10.93
##     b = 1.0255
##     s=0.9057 0.3029 0.5813 1.2518 1.3012 1.4324
##            1.583 1.4286 1.0283 0.8702 0.5773 0.7373
##
##   sigma:  0.2269
##
##      AIC     AICc      BIC
## 6001.785 6002.738 6078.153
##
## Training set error measures:
##                       ME     RMSE      MAE       MPE    MAPE      MASE
## Training set -0.4255136 3.176437 1.867805 -8.598519 17.3914 0.3068462
##                    ACF1
## Training set 0.1676206
```

```
# "AMA" Forbidden Combination
# "AAM" Forbidden Combination
# "AMM" Forbidden Combination
# "MMA" Forbidden Combination
```

Interpretation

So we get the smaller MASE, BIC, AICc and AIC from AAA model. The forecasts from all models are displayed ## in the following plot. They successfully reflect the seasonal behaviour of the series.

```
plot(forecast(fit.solar.AAA),ylab= "Solar Radiations", xlab="Time", main="Time
series plot of monthly
     solar radiations with forecasts (Non Linear")
lines(forecast(fit.solar.MAA)$mean,col="green", type="l")
lines(forecast(fit.solar.MAM)$mean,col="red", type="l")
lines(forecast(fit.solar.MMM)$mean,col="blue", type="l")
legend("bottomleft",lty=1, pch=1, col=1:3, c("AAA","MAA", "MAM", "MMM"), cex=0.
5, y.intersp=0.5)
```



Interpretation ## Forecasts capture the prior trend and seasonality effectively.

Questiion 2

Reading Data and converting to time series and plotting the series

```
data2 <- read.csv("C:/Users/user/Downloads/data2.csv")
data2
```

```
##     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
## 7  Mar-2005  62.5  21195
## 8  Jun-2005  63.2  10904
## 9  Sep-2005  63.1  16995
## 10 Dec-2005  64.0  16962
## 11 Mar-2006  65.4  25004
## 12 Jun-2006  67.2  13059
## 13 Sep-2006  68.1  22327
## 14 Dec-2006  69.5  20372
## 15 Mar-2007  71.0  30109
## 16 Jun-2007  76.0  19448
## 17 Sep-2007  79.5  24993
## 18 Dec-2007  84.7  20988
## 19 Mar-2008  85.7  33497
## 20 Jun-2008  85.5  23375
## 21 Sep-2008  83.0  30174
## 22 Dec-2008  82.3  26736
## 23 Mar-2009  82.5  34387
## 24 Jun-2009  86.9  24262
## 25 Sep-2009  92.5  26940
## 26 Dec-2009  98.6  20375
## 27 Mar-2010 103.3  25923
## 28 Jun-2010 106.2  15929
## 29 Sep-2010 104.3  17609
## 30 Dec-2010 105.7  17001
## 31 Mar-2011 104.7  24667
## 32 Jun-2011 103.5  17439
## 33 Sep-2011 101.3  27892
## 34 Dec-2011 100.0  27378
## 35 Mar-2012  99.4  34569
## 36 Jun-2012  99.3  25773
## 37 Sep-2012  98.6  29453
## 38 Dec-2012 100.4  29174
## 39 Mar-2013 100.8  35704
## 40 Jun-2013 102.7  28048
## 41 Sep-2013 105.9  32942
## 42 Dec-2013 109.7  29061
## 43 Mar-2014 110.7  37877
## 44 Jun-2014 112.1  26282
## 45 Sep-2014 113.1  33154
## 46 Dec-2014 115.2  31115
## 47 Mar-2015 115.9  38857
```

```
## 48 Jun-2015 120.8  27872
## 49 Sep-2015 124.3  32948
## 50 Dec-2015 126.3  31683
## 51 Mar-2016 127.3  43701
## 52 Jun-2016 130.7  37949
## 53 Sep-2016 132.9  32275
## 54 Dec-2016 140.0  32703
```

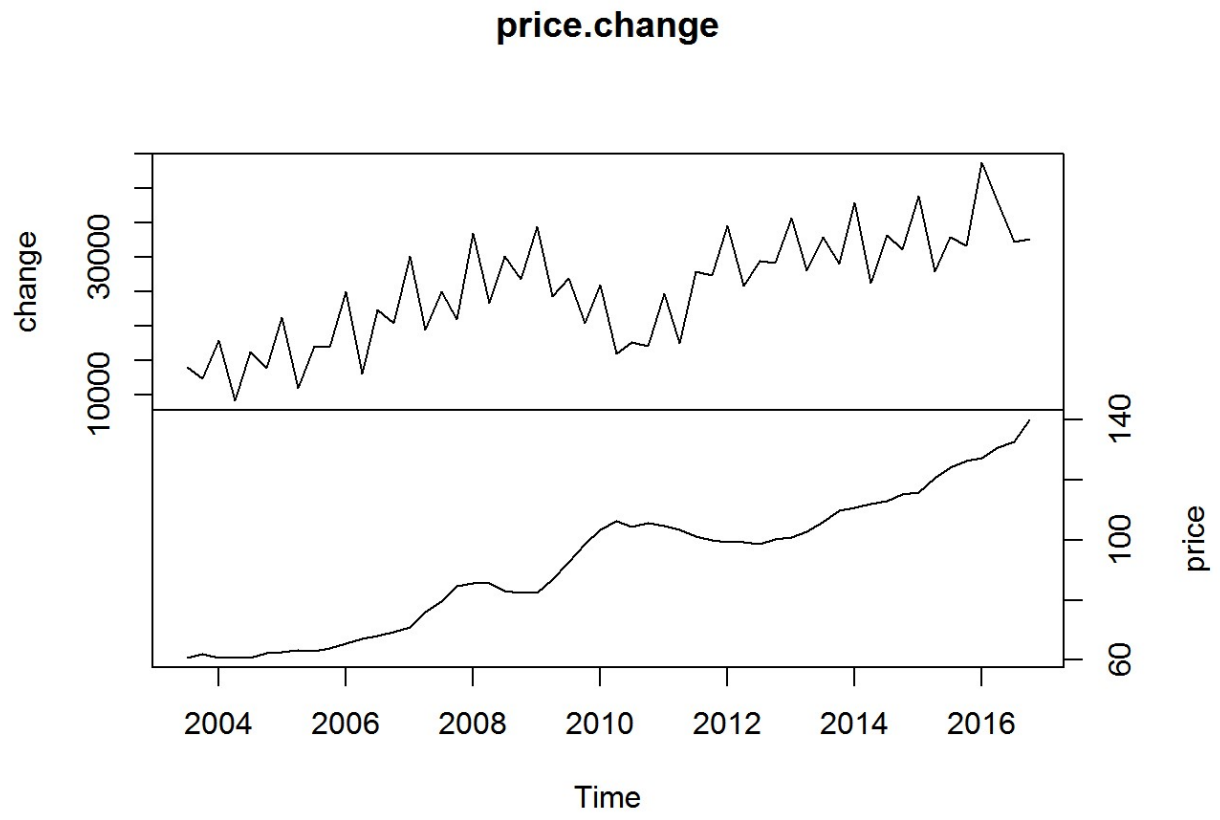
```
price <- ts(data2$price, start=c(2003,3),frequency = 4)
price
```

```
##       Qtr1  Qtr2  Qtr3  Qtr4
## 2003              60.7  62.1
## 2004  60.8  60.9  60.9  62.4
## 2005  62.5  63.2  63.1  64.0
## 2006  65.4  67.2  68.1  69.5
## 2007  71.0  76.0  79.5  84.7
## 2008  85.7  85.5  83.0  82.3
## 2009  82.5  86.9  92.5  98.6
## 2010 103.3 106.2 104.3 105.7
## 2011 104.7 103.5 101.3 100.0
## 2012  99.4  99.3  98.6 100.4
## 2013 100.8 102.7 105.9 109.7
## 2014 110.7 112.1 113.1 115.2
## 2015 115.9 120.8 124.3 126.3
## 2016 127.3 130.7 132.9 140.0
```

```
change <- ts(data2$change, start= c(2003,3), frequency=4)
change
```

```
##       Qtr1  Qtr2  Qtr3  Qtr4
## 2003             14017 12350
## 2004 17894  9079 16210 13788
## 2005 21195 10904 16995 16962
## 2006 25004 13059 22327 20372
## 2007 30109 19448 24993 20988
## 2008 33497 23375 30174 26736
## 2009 34387 24262 26940 20375
## 2010 25923 15929 17609 17001
## 2011 24667 17439 27892 27378
## 2012 34569 25773 29453 29174
## 2013 35704 28048 32942 29061
## 2014 37877 26282 33154 31115
## 2015 38857 27872 32948 31683
## 2016 43701 37949 32275 32703
```

```
price.change=ts.intersect(change,price)
plot(price.change,yax.flip=T)
```

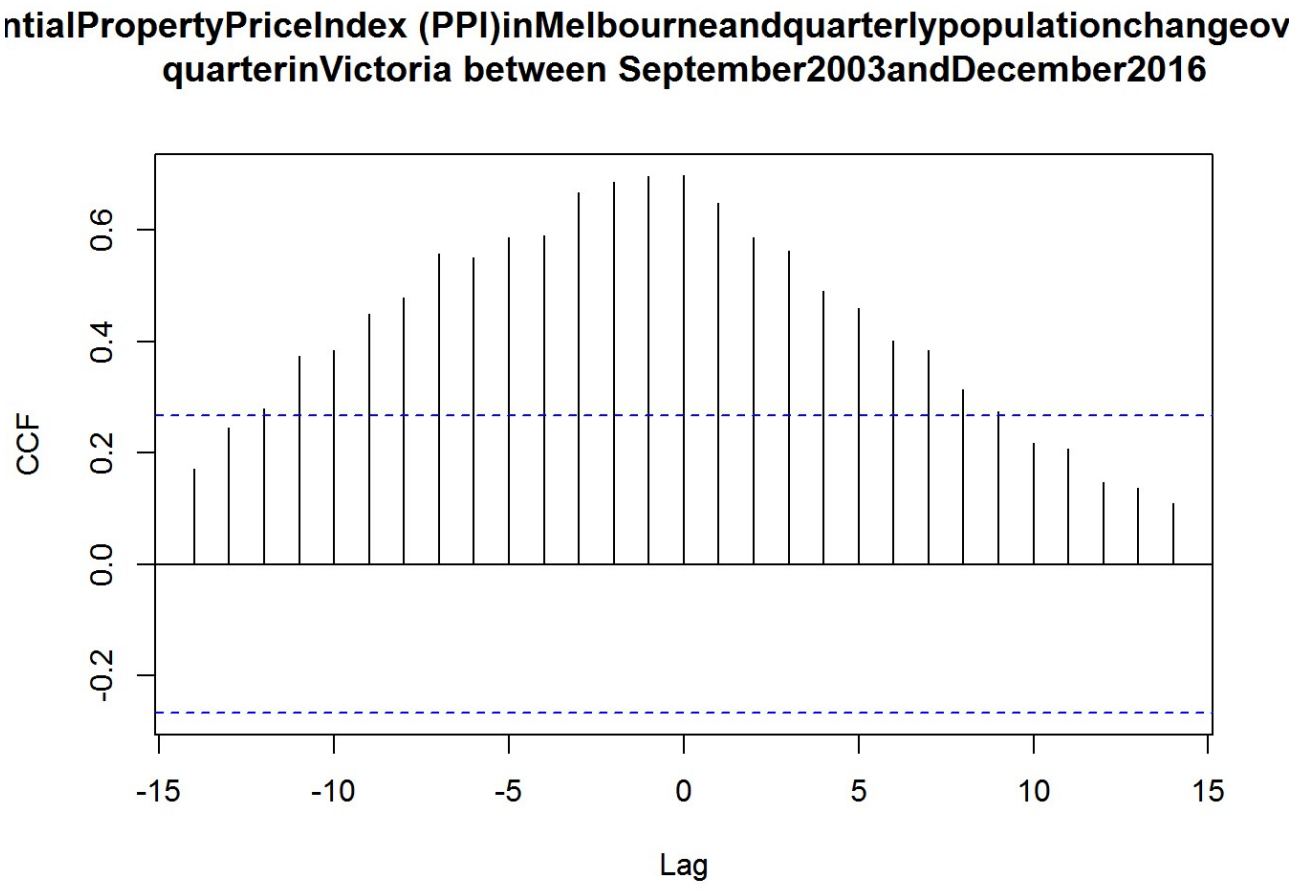


Graphically it seems that there is positive correlation between Property Price Indices and Population
change.

```
ccf(price, change,0, plot=F)
```

```
##
## Autocorrelations of series 'X', by lag
##
##     0
## 0.697
```

```
# The value of sample cross-correlation between these series is 0.697 (quite la
rge) at lag zero and the
# sample CCF is displayed as follows:

ccf(as.vector(price.change[,1]), as.vector(price.change[,2]),ylab='CCF', main
= "Sample CCF between quarterly
Residential Property    Price   Index (PPI) in  Melbourne   and quarterly   pop
ulation  change  over    previous
quarter in  Victoria between September  2003    and December    2016")
```



Nearly all of the cross-correlations are significantly different from zero according to the 1.96/n rule.

There can be genuine correlation but to the nonstationarity in the price series and in the change series

can also be the cause of the spurious correlations found between the two series (if it exists).

To see the efect of nonstationaroty, the sample CCF plot of the first differences of both series is

displayed as follows:

```
ccf(as.vector(diff(ts(price.change[,1]))), as.vector(diff(ts(price.change[,
2]))),ylab='CCF', main =
        "Sample CCF between the first differenced quarterly Property Price Inde
x and quarterly population
 change in Melbourne")
```

When we display the stationary versions of Price and Population change series, the number of singificant lags

in the sample CCF plot is considerably decreased. However, this is not enough to conclude that there is no

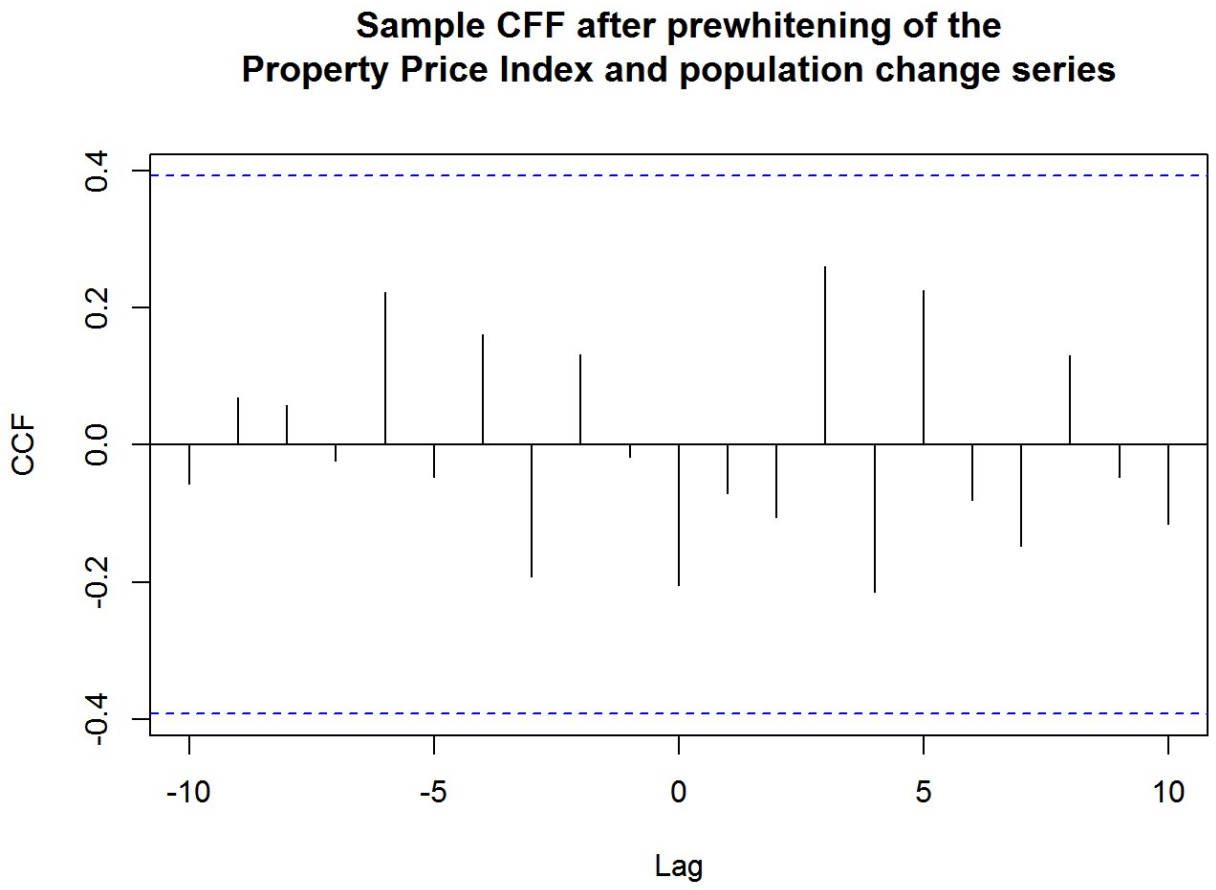
spurious correlation.

With strongly autocorrelated data it is difficult to assess the dependence between the two processes.

A useful approach to disentangle the linear association between X and Y series from their autocorrelation by

the process called prewhitening.

```
library(TSA)

pc.dif=ts.intersect(diff(diff(price,12)),diff(diff(change,12)))
# Unload dplyr
prewhiten(as.vector(pc.dif[,1]),as.vector(pc.dif[,2]), ylab='CCF', main="Sampl
e CFF after prewhitening of the
Property Price Index and population change series")
```



Interpretation ## As there is no significant correlation factor in the resultant prewhitened series, thus, it seems that ## Property Price Indices and population change are not at all correlated, and the strong cross-correlation ## pattern found between the raw data series is spurious.

```
MASE.dynlm <- function(model, ... ){

    options(warn=-1)

    if(!missing(...)) {# Several models
      models = list(model, ...)
      m = length(models)
      for (j in 1:m){
        if ((class(models[[j]])[1] == "polyDlm") | (class(models[[j]])[1] == "d
lm") | (class(models[[j]])[1] == "koyckDlm") | (class(models[[j]])[1] == "ardlD
lm")){
          Y.t = models[[j]]$model$model$y.t
          fitted = models[[j]]$model$fitted.values
        } else if (class(models[[j]])[1] == "lm"){
          Y.t = models[[j]]$model[,1]
          fitted = models[[j]]$fitted.values
        } else if (class(models[[j]])[1] == "dynlm"){
            Y.t = models[[j]]$model$Y.t
            fitted = models[[j]]$fitted.values
        } else {
          stop("MASE function works for lm, dlm, polyDlm, koyckDlm, and ardlDl
m objects. Please make sure that you are sending model object directly or send
a bunch of these objects to the function.")
        }
        # Y.t = models[[j]]$model$y.t
        # fitted = models[[j]]$fitted.values
        n = length(fitted)
        e.t = Y.t - fitted
        sum = 0
        for (i in 2:n){
          sum = sum + abs(Y.t[i] - Y.t[i-1] )
        }
        q.t = e.t / (sum/(n-1))
        if (j == 1){
          MASE = data.frame( n = n , MASE = mean(abs(q.t)))
          colnames(MASE) = c("n" , "MASE")
        } else {
          MASE = rbind(MASE, c(n , mean(abs(q.t))))
        }
      }
      Call <- match.call()
      row.names(MASE) = as.character(Call[-1L])
      MASE
    } else { # Only one model
      if ((class(model)[1] == "polyDlm") | (class(model)[1] == "dlm") | (class
(model)[1] == "koyckDlm") | (class(model)[1] == "ardlDlm")){
        Y.t = model$model$model$y.t
        fitted = model$model$fitted.values
```

```
      } else if (class(model)[1] == "lm"){
        Y.t = model$model[,1]
        fitted = model$fitted.values
      } else if (class(model)[1] == "dynlm"){
        Y.t = model$model$Y.t
        fitted = model$fitted.values
      } else {
        stop("MASE function works for lm, dlm, polyDlm, koyckDlm, and ardlDlm o
bjects. Please make sure that you are sending model object directly or send on
e of these objects to the function.")
      }
      n = length(fitted)
      e.t = Y.t - fitted
      sum = 0
      for (i in 2:n){
        sum = sum + abs(Y.t[i] - Y.t[i-1] )
      }
      q.t = e.t / (sum/(n-1))
      MASE = data.frame( MASE = mean(abs(q.t)))
      colnames(MASE) = c("MASE")
      Call <- match.call()
      row.names(MASE) = as.character(Call[-1L])
      MASE
    }

}
```