Forecasting Solar Radiation

Rushil Sirur

Contents

Objective

The aim of the project is to use Forecasting techniques to model and predict the series of monthly average horizontal solar radiation between January 1960 and December 2014. The task is to find the best fitting forecasting model and then give 2 years ahead forecast.

Libraries

```
# Load libraries
library(TSA,quietly = TRUE)
```

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

```
## This is mgcv 1.8-23. For overview type 'help("mgcv-package")'.
```

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

```
## The following objects are masked from 'package:stats':
##
##     acf, arima
## The following object is masked from 'package:utils':
##
##     tar
```

```r
library(fpp2,quietly = TRUE)
```

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

```r
library(tidyverse,quietly = TRUE)
```

```
## -- Attaching packages --------------------------------------------- tidyverse 1.2.1 --
## v tibble  1.4.2     v purrr   0.2.5
## v tidyr   0.8.0     v dplyr   0.7.5
## v readr   1.1.1     v stringr 1.2.0
## v tibble  1.4.2     v forcats 0.3.0
## -- Conflicts ------------------------------------------------ tidyverse_conflicts() --
## x dplyr::collapse() masks nlme::collapse()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
## x readr::spec()     masks TSA::spec()
```

Data Preparation

Reading in Data

```r
# Reading in Data
solar <- read.csv("data1.csv",header = TRUE)
head(solar)
```

```
##       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
```

```r
# Subsetting Variable of Interest
solar <- solar$solar

# Structure of Dataset
str(solar)
```

```
##  num [1:660] 5.05 6.42 10.85 16.93 24.03 ...
```

Missing Values

Check for missing values in the series.

```
# Missing Values

is.na(solar) %>% summary()
```

```
##    Mode   FALSE
## logical     660
```

There are no missing values in the series.

Create Train and Test Split

Create a train and test split with approximately 80% train and rest test. But this value will be adjusted so that the first observation of the test series is January. Because the data is seasonal (monthly) we try to find the row number which gives a remainder of 1 when divided by 12 (nearest to the 80% of the data).

```
# Observation No to Split
length(solar) * 0.80
```

```
## [1] 528
```

```
# Modulus of 528
528 %% 12
```

```
## [1] 0
```

```
# Modulus of 529
529 %% 12 # 1. The test set starts at the 529th observation of solar series.
```

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

The test set should start at the 529th observation and the train set should end at the 528th observation.

```
# Create Train Set
solarTrain <- window(x = solar,end = 528)

# Create Test Set
solarTest <- window(x = solar,start = 529)
```

Create TS Object

The data is monthly, so the frequency of the series is 12. Convert both the train and test sets to a time series object with frequency of 12.

```
# Train Series Conversion to TS
solarTrain <- ts(data = solarTrain,start = c(1960,1),frequency = 12)
head(solarTrain)
```

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

```
str(solarTrain)
```

```
##  Time-Series [1:528] from 1960 to 2004: 5.05 6.42 10.85 16.93 24.03 ...
```

```r
# Test Series Conversion to TS
solarTest <- ts(data = solarTest,start = c(2004,1),frequency = 12)
head(solarTest)
```

```
##            Jan       Feb       Mar       Apr       May       Jun
## 2004  5.281160  6.479864 11.939498 15.568476 18.544593 19.822544
```

```r
str(solarTest)
```

```
##  Time-Series [1:132] from 2004 to 2015: 5.28 6.48 11.94 15.57 18.54 ...
```

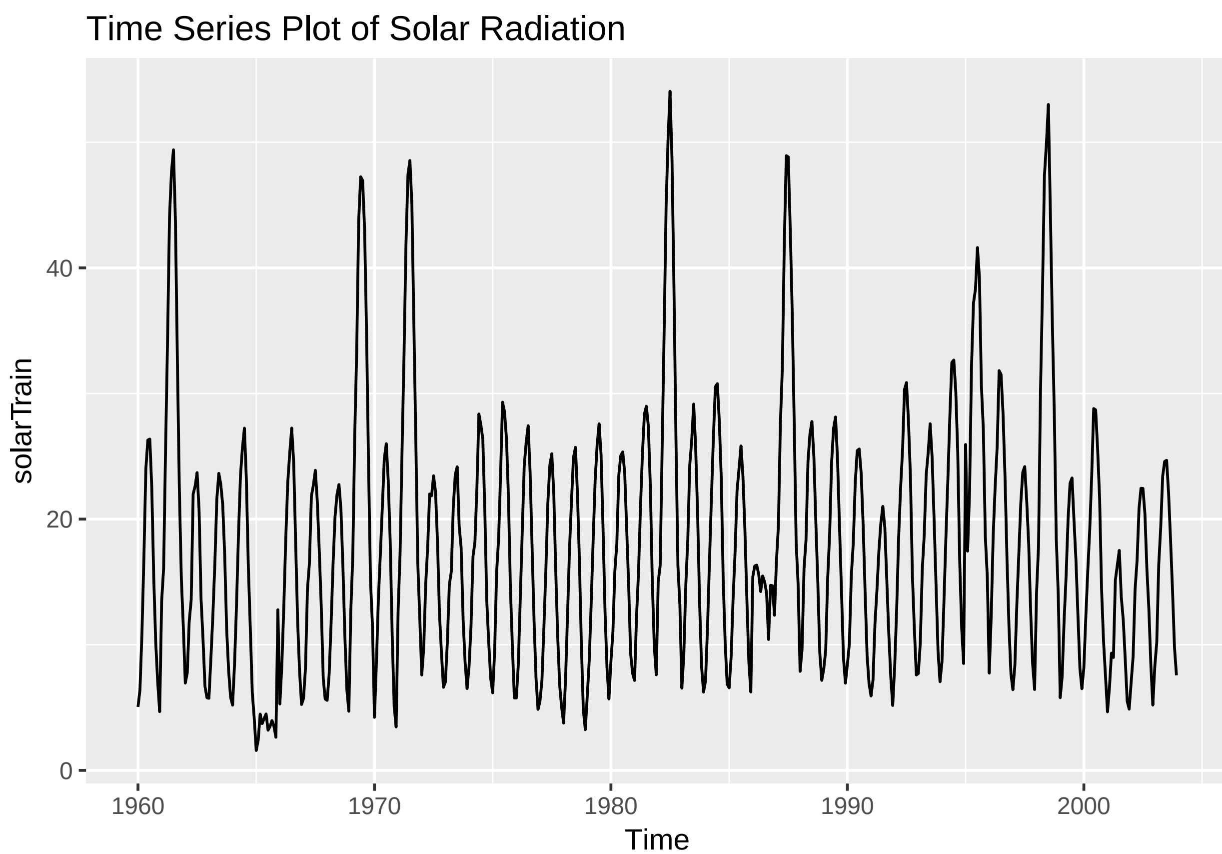
Data Exploration

The data exploration step involves trying to understand the characteristics of the series.

Time Series Plot

The time series plot of training set is below.

```r
# Time Series Plot
autoplot(solarTrain) +
    ggtitle("Time Series Plot of Solar Radiation")
```



The time series plot shows the following characteristics:

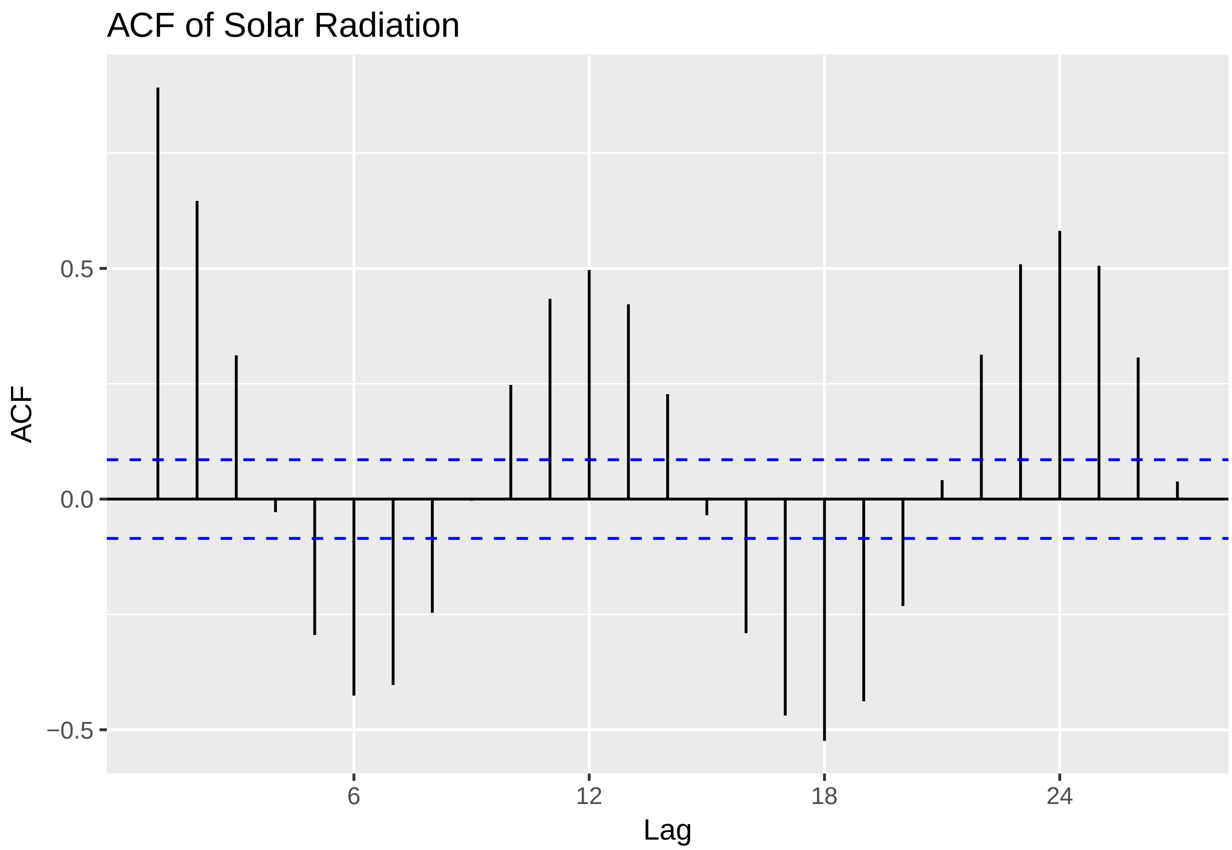
1. Seasonal Pattern: A clear seasonal pattern is seen in the plot.

2. Trend: There is no visible trend in the data.
3. The seasonal patterns vary across the series.

ACF Plot

Plot the ACF Plot for the training set and check for any characteristics.

```
# ACF of Train Set
ggAcf(solarTrain) + ggtitle("ACF of Solar Radiation")
```

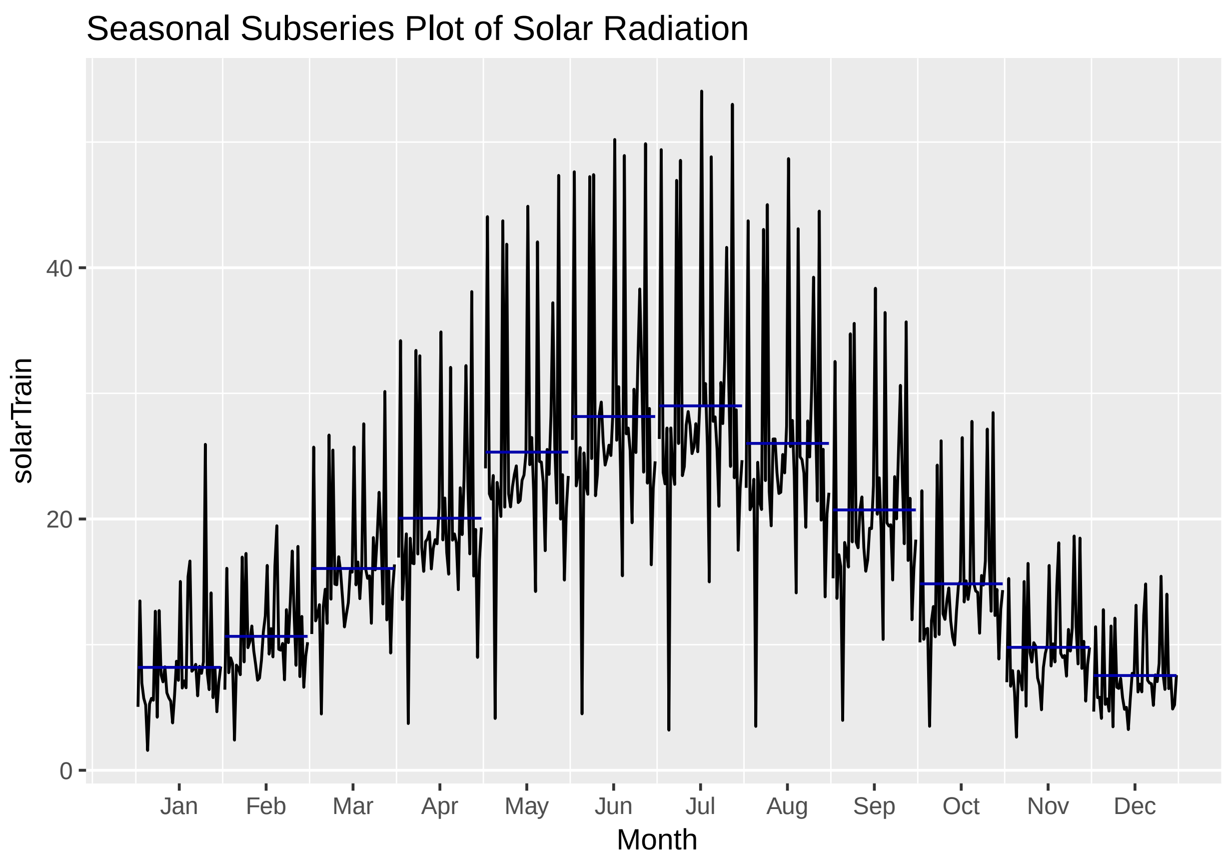


The ACF plot shows a seasonal pattern with frequency 12. Also, there is no sign of trend.

Seasonal Subseries Plot

Plot the seasonal subseries plot to check the seasonal pattern between seasons and within seasons.

```
# Seasonal Subseries Plot
ggsubseriesplot(x = solarTrain) + ggtitle("Seasonal Subseries Plot of Solar Radiation")
```



The mean radiation is lowest during the start and the end of the year and highest during mid year (June, July). Within each month there is a lot of variation in the solar radiation values.

ARMA Characteristics

We check the possible ARMA characteristics for the series.

```
# Stationarity of Solar Radiation Data

# AdfTest
ar(diff(solarTrain)) # Selected 27 lags
```

```
##
## Call:
## ar(x = diff(solarTrain))
##
## Coefficients:
##       1        2        3        4        5        6        7        8
##  0.1358   0.2634   0.0498  -0.0971  -0.2374  -0.1657  -0.0717  -0.1681
##       9       10       11       12       13       14       15       16
## -0.0366  -0.0076   0.0895  -0.0626  -0.0885  -0.0529  -0.1469  -0.0915
##      17       18       19       20       21       22       23       24
## -0.0119  -0.0175  -0.0283  -0.0719  -0.1159   0.0018   0.0157   0.1081
##      25       26       27
##  0.0131  -0.0557  -0.1258
##
```

```
## Order selected 27  sigma^2 estimated as  6.082
```

```r
fUnitRoots::adfTest(solarTrain,lags = 27)# P Value 0.36, so accept null hypothesis of non stationarity
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 27
##   STATISTIC:
##     Dickey-Fuller: -0.7813
##   P VALUE:
##     0.3673
##
## Description:
##  Sat Jan 05 16:13:43 2019 by user: Rushil Sirur
```

```r
# Seasonal Difference Series
solarDiff <- diff(solarTrain,lag = 12)

# Adf of Differenced Series
ar(diff(solarDiff)) # Lags selected 24
```

```
##
## Call:
## ar(x = diff(solarDiff))
##
## Coefficients:
##       1        2        3        4        5        6        7        8
##  0.0661   0.2038   0.1639   0.0176  -0.1825  -0.1399  -0.0780  -0.1004
##       9       10       11       12       13       14       15       16
##  0.0447  -0.0248   0.0730  -0.7913  -0.0379   0.0816   0.0902   0.0069
##      17       18       19       20       21       22       23       24
## -0.0850  -0.0824  -0.0731  -0.0719   0.0036  -0.0024   0.0775  -0.3488
##
## Order selected 24  sigma^2 estimated as  7.003
```

```r
fUnitRoots::adfTest(x = solarDiff,lags = 24)#p value 0.01 Accept alternate hypothesis of stationarity
```

```
## Warning in fUnitRoots::adfTest(x = solarDiff, lags = 24): p-value smaller
## than printed p-value
```

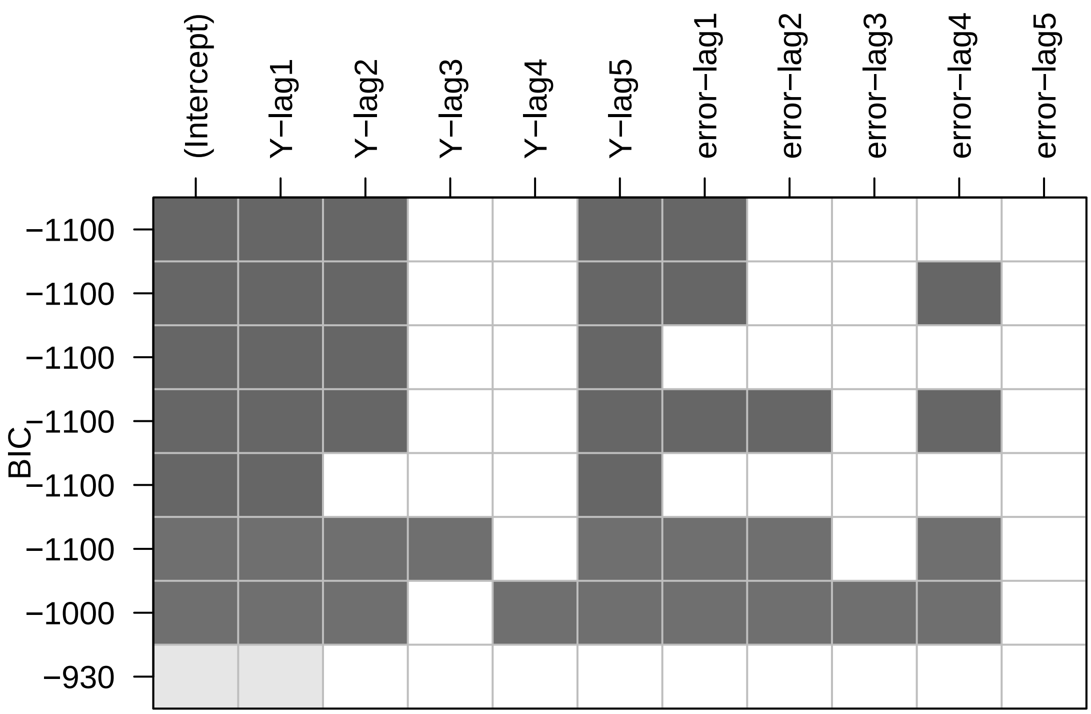
```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 24
##   STATISTIC:
##     Dickey-Fuller: -4.6155
##   P VALUE:
##     0.01
##
## Description:
```

```
##  Sat Jan 05 16:13:43 2019 by user: Rushil Sirur
```

```r
# EACF of Differenced Series
eacf(z = solarDiff)# indicates AR 2 and MA 2
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x x x o x x x x  x  x  x
## 1 x x x x x x o x x x x  x  x  x
## 2 x x o x x o x x x x o  x  x  o
## 3 x x x x x o x o o o x  x  x  o
## 4 x x x x o o x o o o x  x  x  o
## 5 o o x o x x o o o o o  x  x  x
## 6 x o x o x x o o o o o  x  x  x
## 7 o o x x x x o o o o o  x  x  o
```

```r
# ARMA Subsets
plot(armasubsets(solarDiff,nar = 5,nma = 5))# AR 2 and MA 1
```



The possible SARIMA Models indicated are ARIMA(0,0,0)x(2,1,1 or 2). The idea here is not to find an ARIMA model, but the above indicates that ARIMA models can also be explored in addition to models incorporatin Trend and Seasonality.

Analysis

We consider models which deal with characteristics of trend and seasonality and also models that deal with Autocorrelation structure.

Initially we apply some basic forecasting methods before moving to advanced methods. We use the MASE values to compare the models.

Methodology

The following methodology is applied to all models.

1. Model Fitting

In this step we fit the different models to the training data.

2. Plot of Model Fit

The model is plotted by overlaying the fitted model on the Training series.

3. Accuracy on Training Data

The accuracy on Training data using MASE values in measured.

4. Residual Plots

We check the residual plots. Mainly we check the autocorrelation in the ACF of the residuals and time plot of residuals. In the autocorrelation we would like to see no autocorrelation and in the time plot we would ideally like to see no patterns and no changing variance of the residuals.

5. Residual Statistics

We check the mean of residuals and Shapiro Wilk test to check normality of residuals. The mean should be zero and normality is ideal.

In the residuals, the autocorrelation and mean of residuals are important as far the point forecasts are concerned. The normality and changing variance affect the prediction intervals of the point forecast.

6. Test Prediction

We get the forecast for the test series and then get the accuracy using MASE between the predictions and actual test series. If the residuals are not normal then we can use the bootstrap method to calculate prediction intervals.

7. Forecast Plot

The forecasts are plotted to how it compares with the series.

8. Observations

Summarize the model information and update the results in the results data frame.

The results of the models will be stored in a data frame.

Creating the data frame below:

```
# Results Data Frame
modelResults <- data.frame(
    Model = character(),
    TrainAcc. = numeric(),
    TestAcc. = numeric(),
    Res.Autocorr = character(),
    Res.Variance = character(),
```

```r
    Res.Timeplot = character(),
    Res.Mean = numeric(),
    Res.Normality = character(),
    stringsAsFactors = FALSE
)


# Setting Forecast Length
h <- length(solarTest)
```

Seasonal Naive Model

In the seasonal Naive method the forecast is set to the value observed in the same month of the last year.

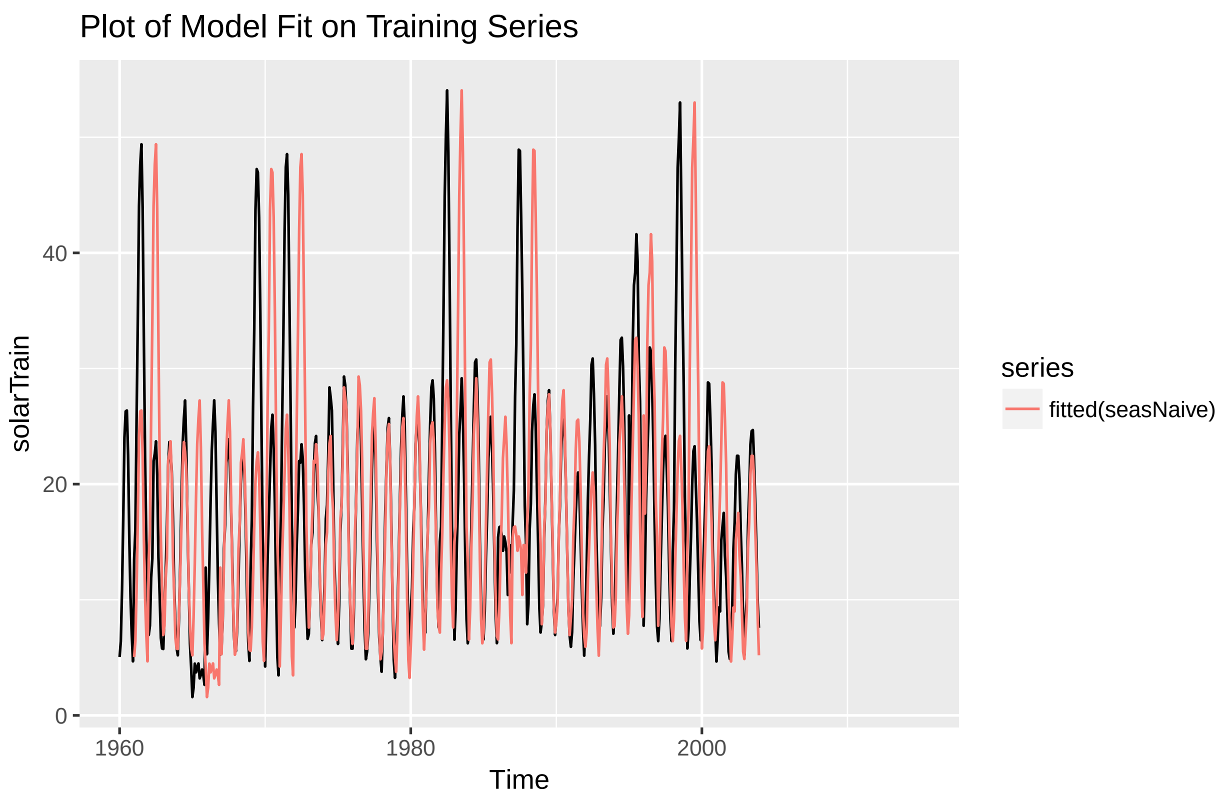
Model Fitting

```r
# Seasonal Naive Method on Original Data
seasNaive <- snaive(y = solarTrain,h = h)
```

Plot of Model Fit

```r
# Plot of Model Fit on Train Data
autoplot(seasNaive,fcol = NA,PI = FALSE) +
    autolayer(object = fitted(seasNaive)) +
    ggtitle("Plot of Model Fit on Training Series")
```

```
## Warning: Removed 12 rows containing missing values (geom_path).
```



Accuracy on Training Data

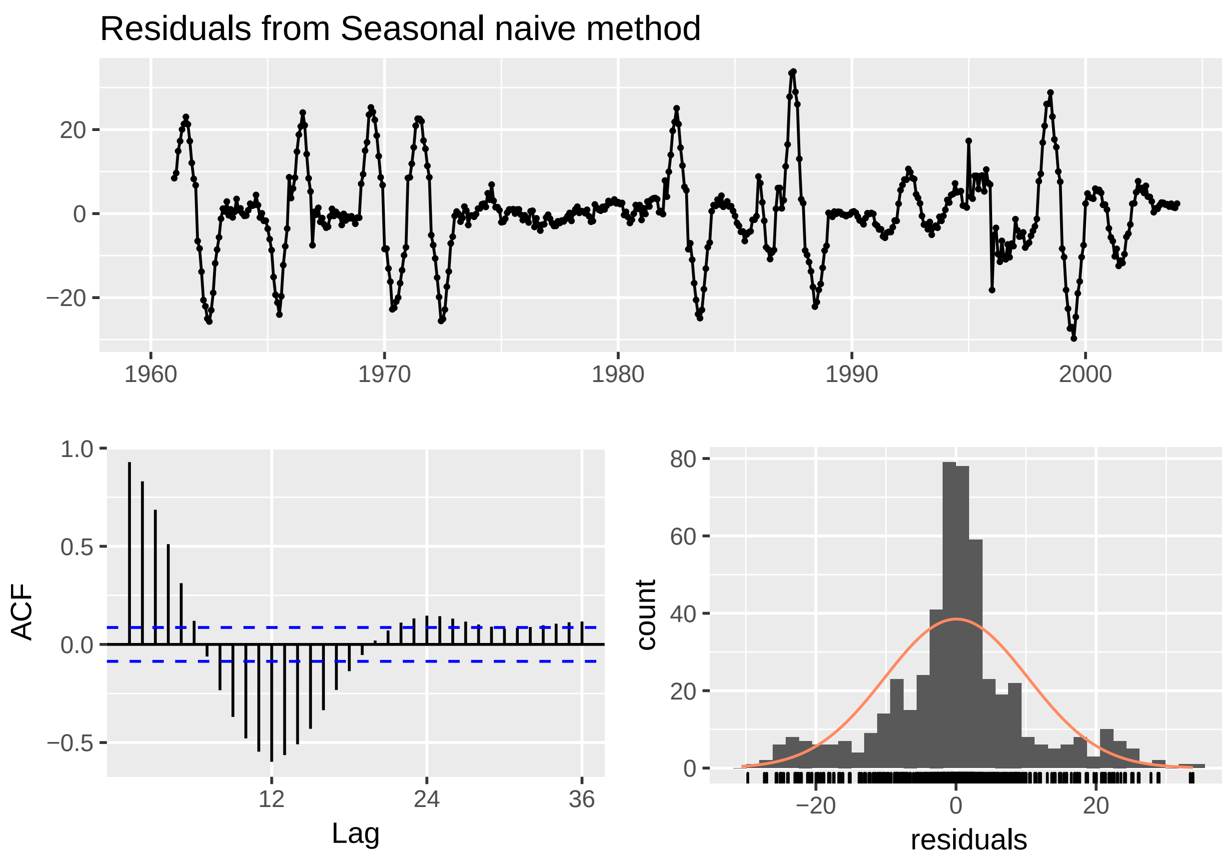
```r
# Accuracy on Training Set
accuracy(f = seasNaive)# MASE 1
```

```
##                      ME     RMSE      MAE       MPE     MAPE MASE
## Training set 0.04539614 10.21634 6.997806 -16.34367 45.38632    1
##                   ACF1
## Training set 0.9289044
```

Residual Plots

Analyse the residuals plots of the Model Fit.

```r
# Residual Plots of Train Data
checkresiduals(seasNaive)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Seasonal naive method
## Q* = 2355, df = 24, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 24
```

Autocorrelation: The ACF plot and Ljung Box test indicate presence of autocorrelation in the series.

Constant Variance: The variance of the residuals is also not constant as seen in the time plot of residuals. Also patterns in the time plot shows the fit is not good.

Residual Statistics

```
# Mean of Residuals of Train Data
mean(resid(seasNaive),na.rm = TRUE)
```

```
## [1] 0.04539614
```

```
# Normality of Residuals of Train Data
shapiro.test(x = seasNaive$residuals)# Null: Normality
```

```
##
##  Shapiro-Wilk normality test
##
## data:  seasNaive$residuals
```

```
## W = 0.95134, p-value = 5.321e-12
```

Mean of Residuals: The mean of residuals should ideally be close to 0. But here it is 0.045, so the fit is not very good.

Normality of Residuals: The Shapiro Wilk test indicates that the residuals are not normal.

Test Prediction

We apply the model on the test set and get accuracy measures.

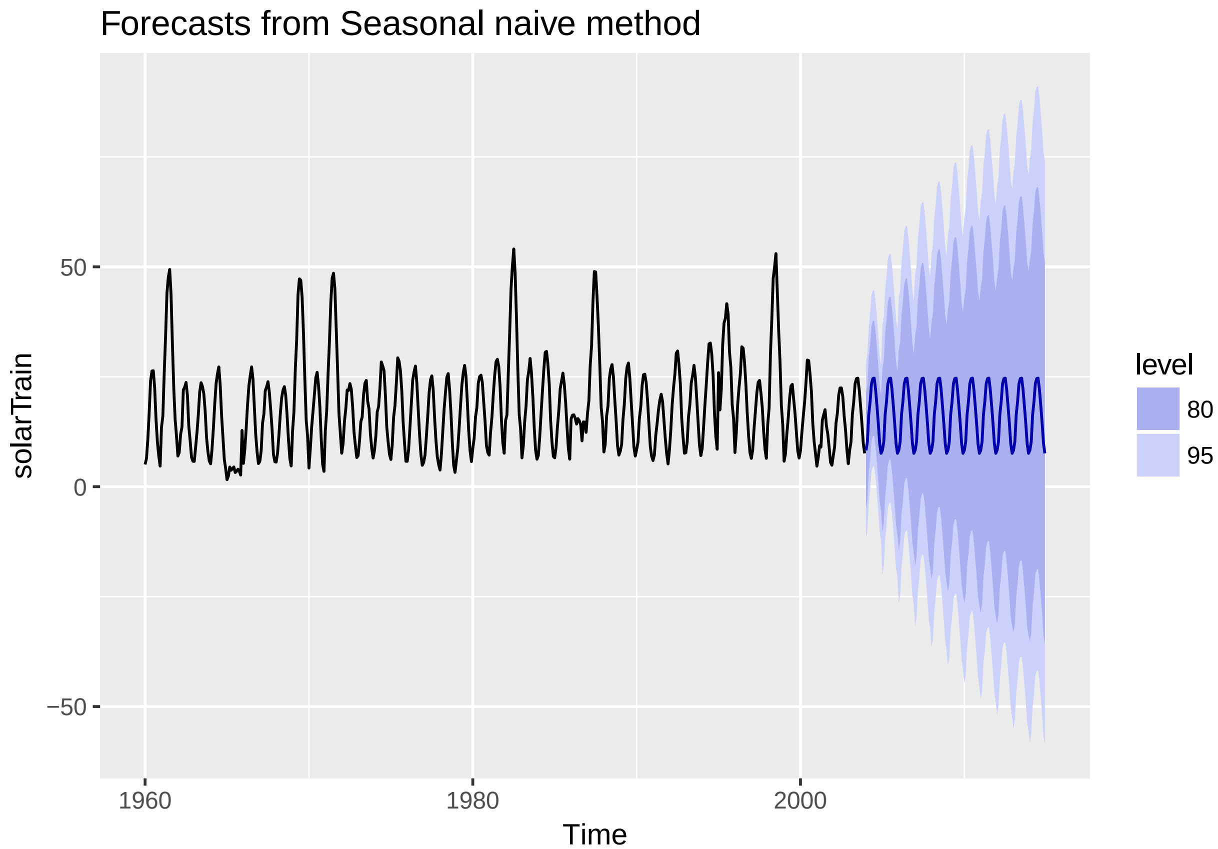
```r
# Generate forecasts from model
seasNaiveFcast <- forecast(object = seasNaive,h = h,simulate = TRUE,bootstrap = TRUE)



# Compare Accuracy With Test Set for Predictions
seasNaiveTestAcc <- accuracy(f = seasNaiveFcast,x = solarTest)
seasNaiveTestAcc
```

```
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 0.04539614 10.21634 6.997806 -16.343674 45.38632 1.0000000
## Test set     0.23414855  3.68921 2.971007  -6.026305 19.50929 0.4245627
##                   ACF1 Theil's U
## Training set 0.9289044        NA
## Test set     0.8688421 0.9053853
```

The MASE value for the training set is 1, while for the test set it is 0.424, which means it actually does better on the test set.

Forecast Plot

```r
autoplot(seasNaiveFcast)
```



Observations

The seasonal naive method gives a MASE of 0.424 on the test data but there are issues with the residuals with autocorrelation, non normality.

Adding the results to Results dataframe.

```
# Update modelResults

#c("Model","TrainAcc.","TestAcc.","Res.Autocorr","Res.Variance","Res.Timeplot","Res.Mean","Res.Normalit

# Train Data Results
seasNaiveResult <- data.frame("Seasonal Naive",1,0.424,"Present","Present","Pattern",0.045,"Non Normali

names(seasNaiveResult) <- c("Model","TrainAcc.","TestAcc.","Res.Autocorr","Res.Variance","Res.Timeplot"


modelResults <- rbind(modelResults,seasNaiveResult)


rm(seasNaive,seasNaiveFcast,seasNaiveResult,seasNaiveTestAcc)
```

Linear Seasonal Model

This model fits a linear model to the data, with the predictor being the seasonal periods in the data.

Model Fitting

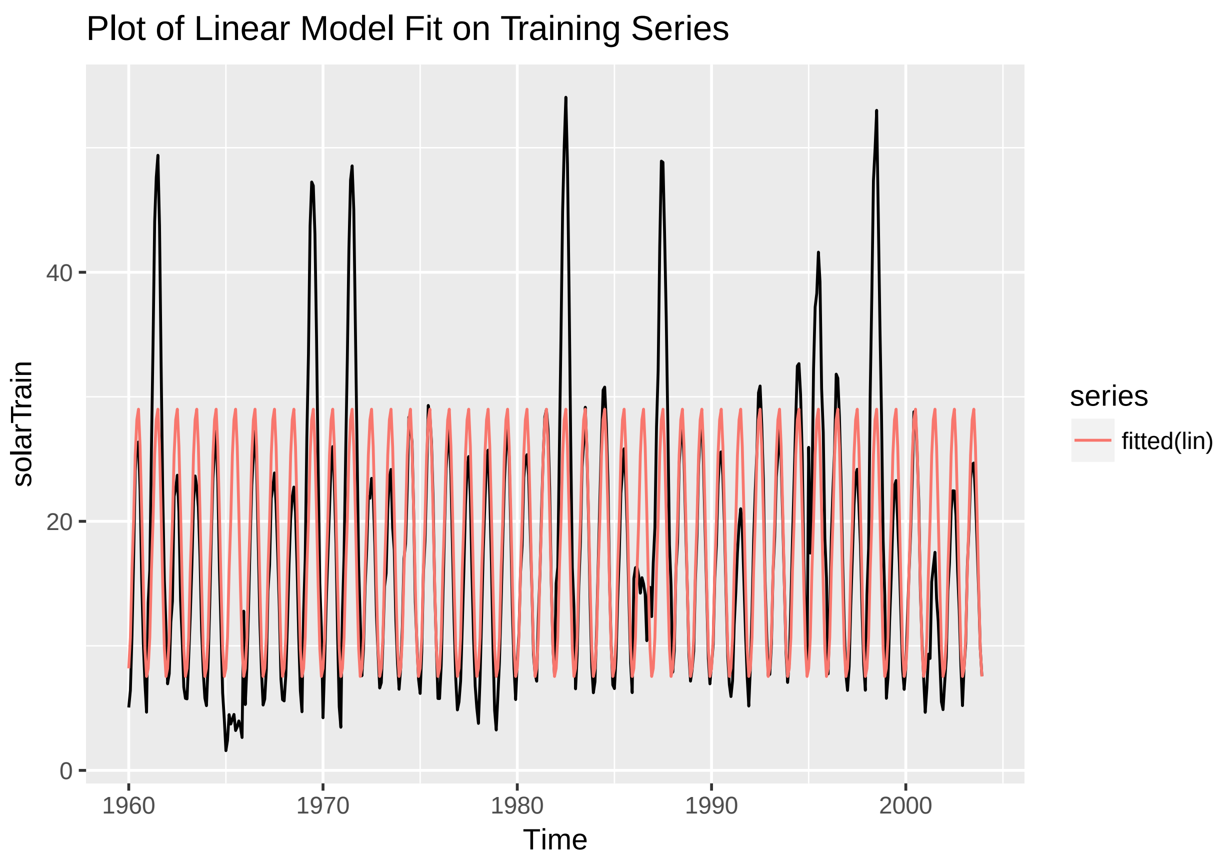
```
# Linear Seasonal Model on Train Data
lin <- tslm(formula = solarTrain ~ season)
summary(lin)
```

```
##
## Call:
## tslm(formula = solarTrain ~ season)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -25.8073  -3.2949  -1.4320   0.8299  25.0511
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.1912     1.0347   7.916 1.51e-14 ***
## season2       2.4678     1.4633   1.686   0.0923 .
## season3       7.8695     1.4633   5.378 1.14e-07 ***
## season4      11.8696     1.4633   8.111 3.68e-15 ***
## season5      17.1312     1.4633  11.707  < 2e-16 ***
## season6      19.9663     1.4633  13.644  < 2e-16 ***
## season7      20.8141     1.4633  14.224  < 2e-16 ***
## season8      17.8251     1.4633  12.181  < 2e-16 ***
## season9      12.5335     1.4633   8.565  < 2e-16 ***
## season10      6.6524     1.4633   4.546 6.82e-06 ***
## season11      1.5940     1.4633   1.089   0.2765
## season12     -0.6505     1.4633  -0.445   0.6568
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.864 on 516 degrees of freedom
## Multiple R-squared:  0.557,  Adjusted R-squared:  0.5475
## F-statistic: 58.97 on 11 and 516 DF,  p-value: < 2.2e-16
```

The linear model on Train data is significant overall and has an adjusted R Square of 0.5475.

Plot of Model Fit

```
# Plot of Model Fit on Train Data
autoplot(solarTrain,fcol = NA,PI = FALSE) +
    autolayer(object = fitted(lin)) +
    ggtitle("Plot of Linear Model Fit on Training Series")
```



Accuracy on Training Data

```
# Accuracy on Training Set
accuracy(f = lin)# MASE 0.5615
```

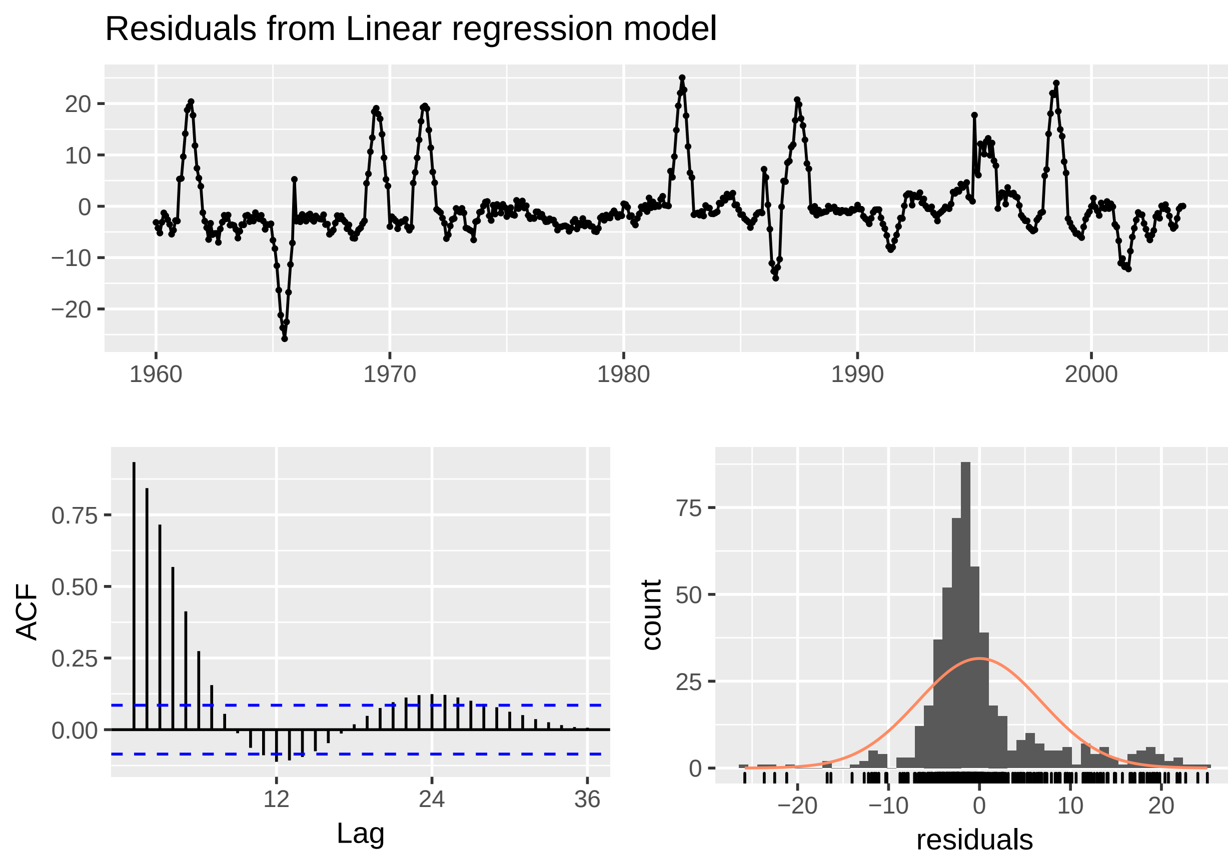
```
##                        ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 1.858797e-16 6.785157 4.499788 -17.36976 33.09337 0.5615581
```

The training accuracy of the linear model is MASE 0.561

Residual Plots

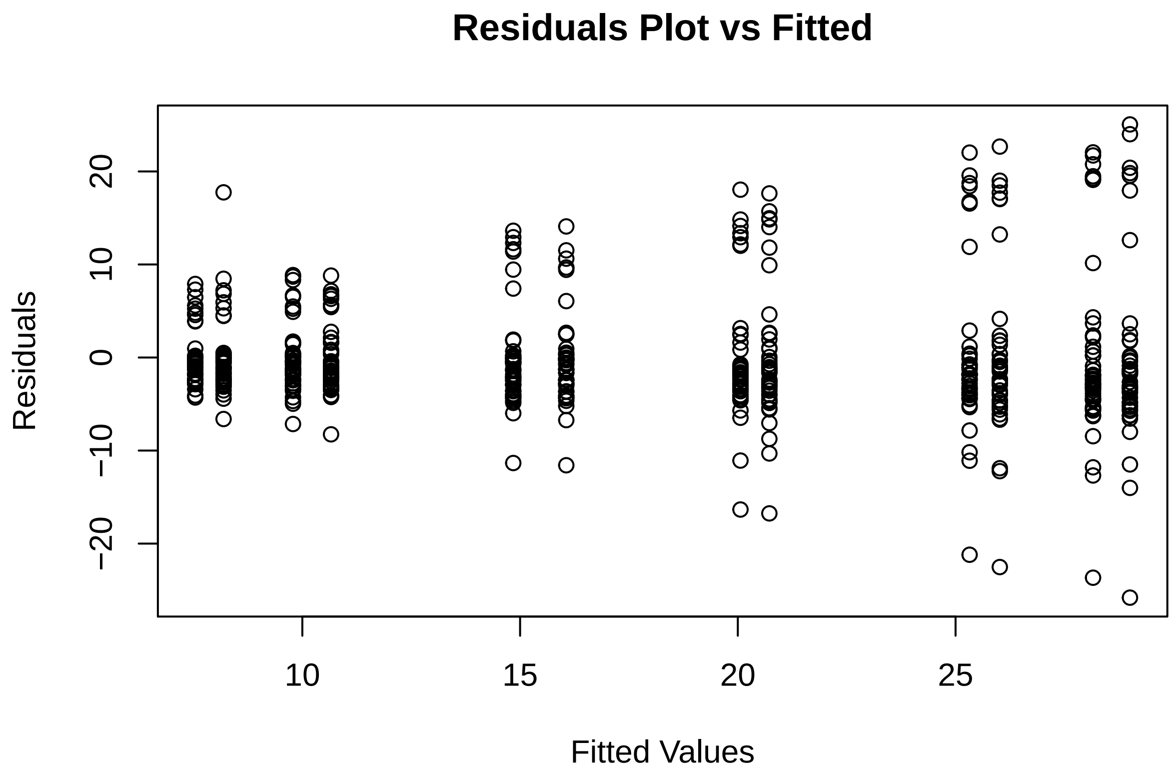
Analyse the residuals plots of the Model Fit.

```
# Residual Plots of Train Data
checkresiduals(lin)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 24
##
## data:  Residuals from Linear regression model
## LM test = 476.17, df = 24, p-value < 2.2e-16
```

```
# Fitted vs Residuals
plot(x = lin$fitted.values,y = lin$residuals ,xlab = "Fitted Values",ylab = "Residuals",main = "Residual
```



Autocorrelation: There is still autocorrelations in the residuals of the model.

Constant Variance: There is changing variance in the residuals. Also the fit of the model is not good since there are patterns in the data.

The fitted values against the predictor also shows a changing variance which indicates the fit could be improved.

Residual Statistics

```r
# Mean of Residuals of Train Data
mean(resid(lin),na.rm = TRUE)
```

```
## [1] 2.414421e-16
```

```r
# Normality of Residuals of Train Data
shapiro.test(x = lin$residuals)# Null: Normality
```

```
##
##  Shapiro-Wilk normality test
##
## data:  lin$residuals
## W = 0.84562, p-value < 2.2e-16
```

The model has almost zero mean for residuals and non normal residuals.

Test Prediction

We apply the model on the test set and get accuracy measures.

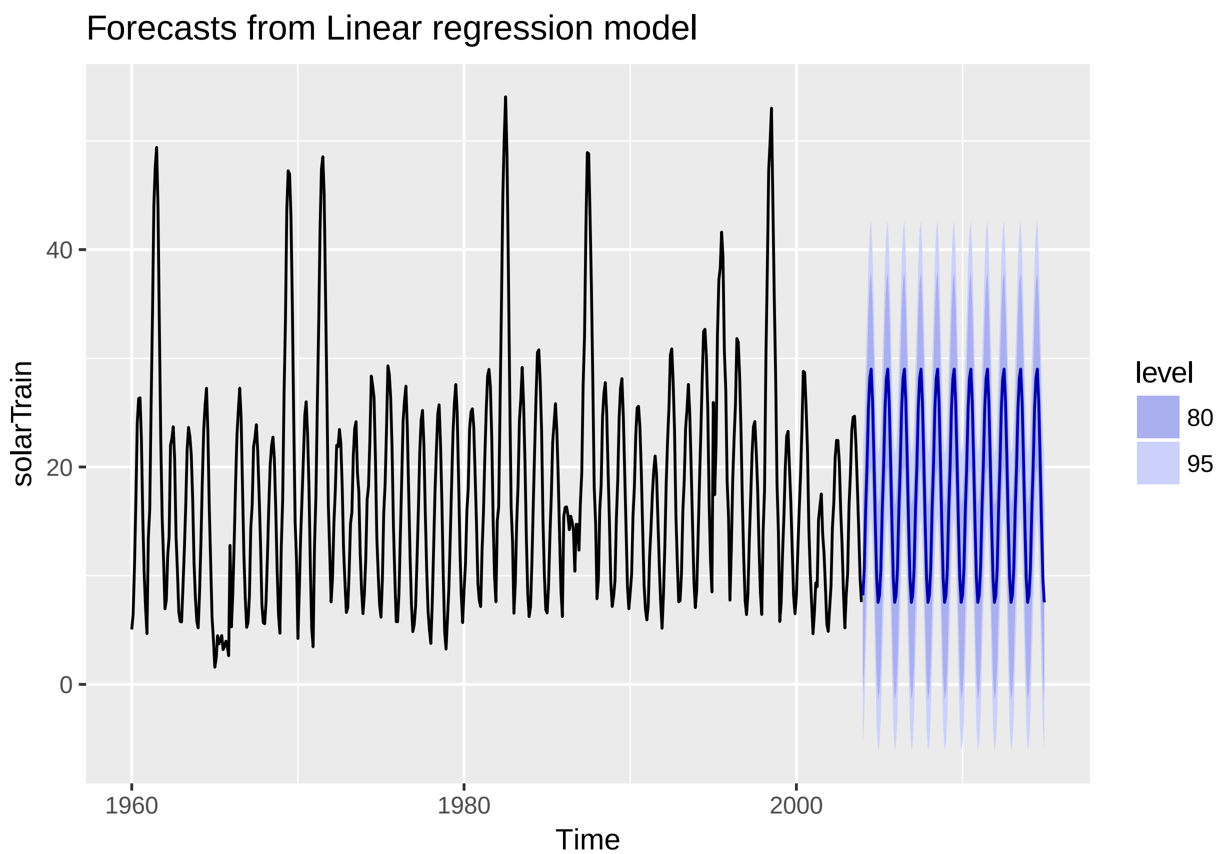
```
# Generate forecasts for Train Data
linFcast <- forecast(object = lin,h = h,simulate = TRUE,bootstrap = TRUE)

# Compare Accuracy With Test Set
linTestAcc <- accuracy(f = linFcast,x = solarTest)
linTestAcc
```

```
##                         ME     RMSE      MAE       MPE     MAPE      MASE
## Training set  1.858797e-16 6.785157 4.499788 -17.36976 33.09337 0.6430284
## Test set     -1.213427e+00 3.667776 2.592286 -12.49986 19.10203 0.3704427
##                   ACF1 Theil's U
## Training set 0.9340587        NA
## Test set     0.8971213 0.9805108
```

The linear model gives a MASE of 0.3704 on the test set.

Forecast Plot

```
autoplot(object = linFcast)
```



Observations

The linear model gives a better result than the Seasonal Naive model on the test set with a MASE of 0.370, but the residuals again have correlation which indicates that the model fitting is not very good.

Adding the results to Results dataframe.

```
# Update modelResults

#c("Model","TrainAcc.","TestAcc.","Res.Autocorr","Res.Variance","Res.Timeplot","Res.Mean","Res.Normalit

# Train Data Results
linResult <- data.frame("Linear Model",0.5615,0.370,"Present","Present","Pattern",2.414*10^-16,"Non Nor

names(linResult) <- c("Model","TrainAcc.","TestAcc.","Res.Autocorr","Res.Variance","Res.Timeplot","Res.


modelResults <- rbind(modelResults,linResult)


rm(lin,linFcast,linResult,linTestAcc)
```

STL Decomposition

This models is the Seasonal and Trend Decomposition using loess method. It is primarily a decomposition method but can be used to produce forecasts.

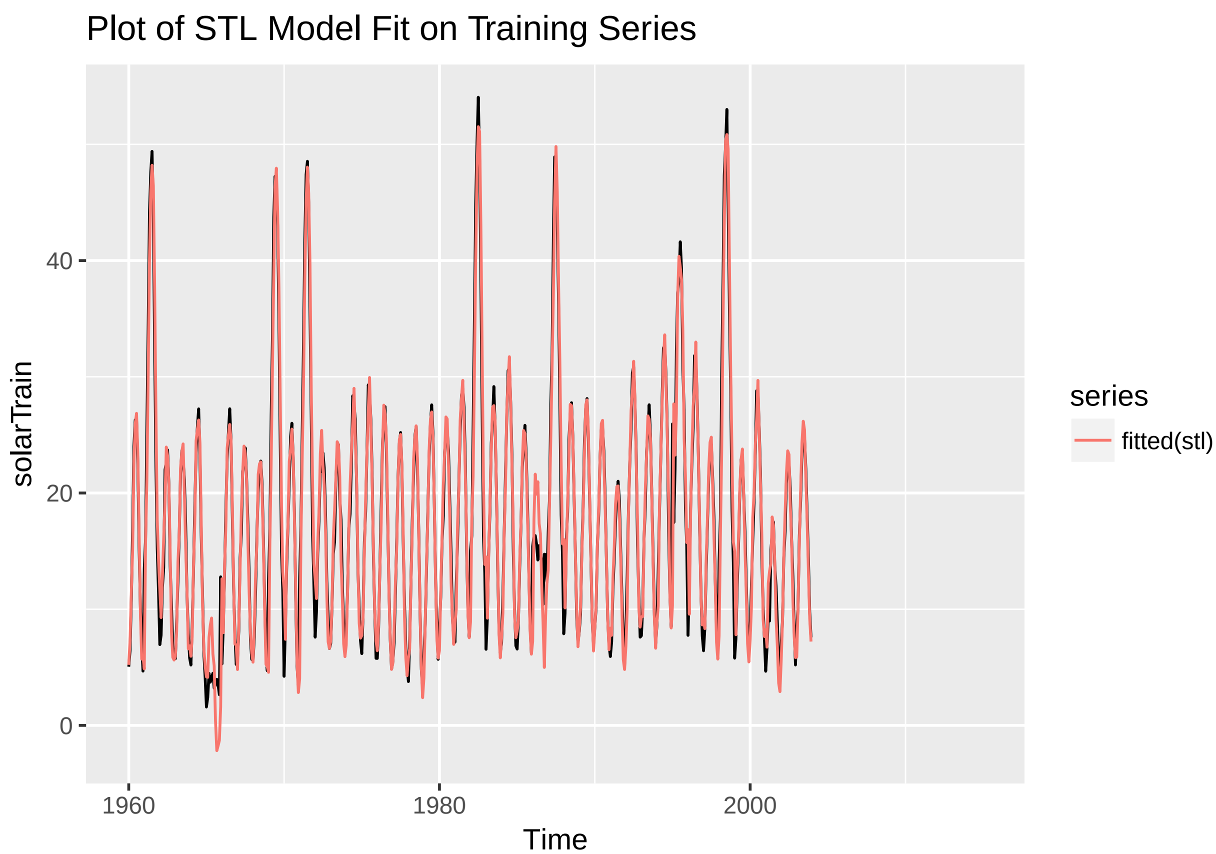
Model Fitting

```
# STL Model on Train Data
stl <- stlf(y = solarTrain,h = h,s.window = 15,method = "ets",etsmodel = "ZNN")
```

The STLF function applies the STF method. The seasonal component is estimated and then the seasonally adjusted data is estimated using a trend model. Then the trend and seasonal values are combined to give forecasts.

Plot of Model Fit

```
# Plot of Model Fit on Train Data
autoplot(stl,fcol = NA,PI = FALSE) +
    autolayer(object = fitted(stl)) +
    ggtitle("Plot of STL Model Fit on Training Series")
```



Accuracy on Training Data

```
# Accuracy on Training Set for Train Data
accuracy(f = stl)# MASE 0.217
```

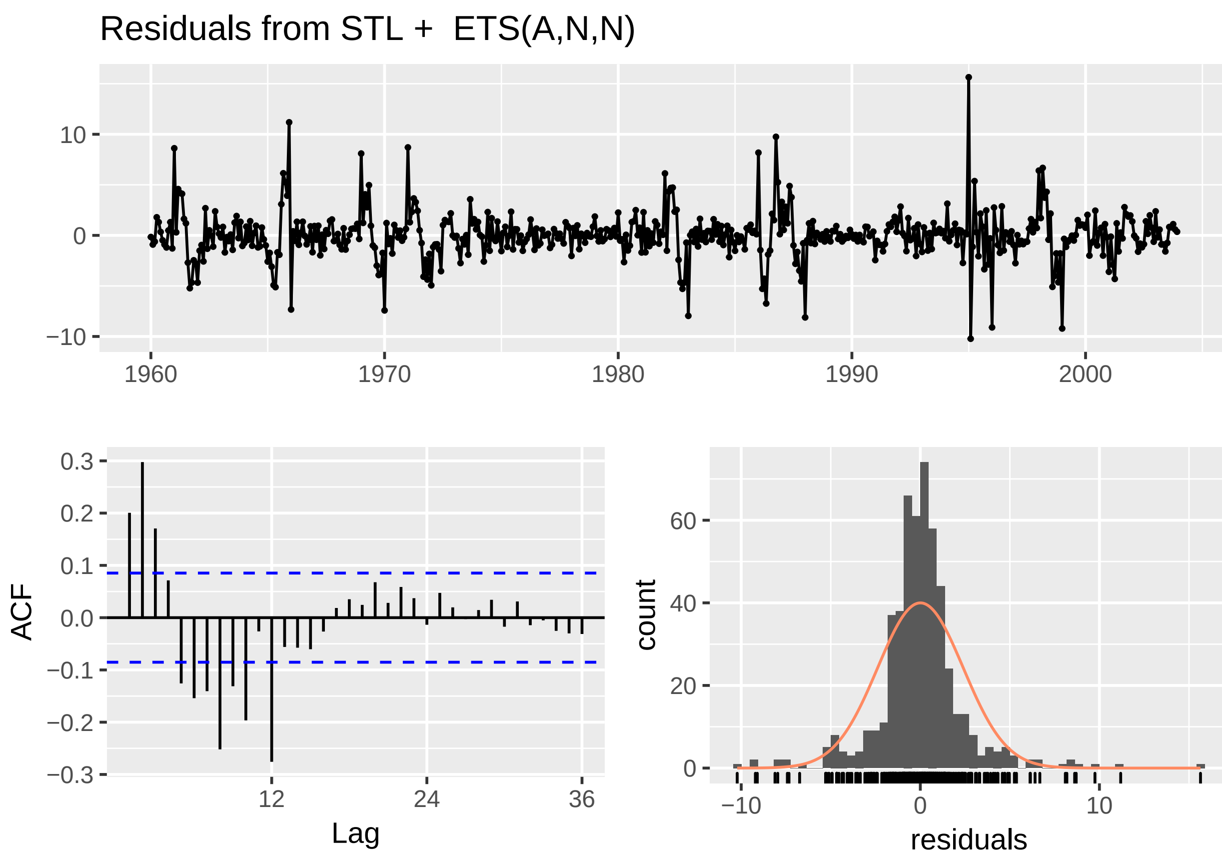
```
##                       ME    RMSE      MAE       MPE     MAPE      MASE
## Training set 0.006162009 2.38838 1.520788 -1.484613 12.75084 0.2173236
##                   ACF1
## Training set 0.2005732
```

The training accuracy of the linear model is MASE 0.217

Residual Plots

Analyse the residuals plots of the Model Fit.

```
# Residual Plots of Train Data
checkresiduals(stl)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from STL +  ETS(A,N,N)
## Q* = 237.27, df = 22, p-value < 2.2e-16
##
## Model df: 2.   Total lags used: 24
```

Autocorrelation: There is still autocorrelations in the residuals of the model.

Constant Variance: There is changing variance in the residuals. Also the fit of the model is not good since there are patterns in the data.

But this model seems better than the previous models in terms of residuals even though this is also not error free.

Residual Statistics

```
# Mean of Residuals of Train Data
mean(resid(stl),na.rm = TRUE)
```

```
## [1] 0.006162009
```

```
# Normality of Residuals of Train Data
shapiro.test(x = stl$residuals)# Null: Normality
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  stl$residuals
## W = 0.88287, p-value < 2.2e-16
```

The model has almost zero mean for residuals and non normal residuals.

Test Prediction

We apply the model on the test set and get accuracy measures.

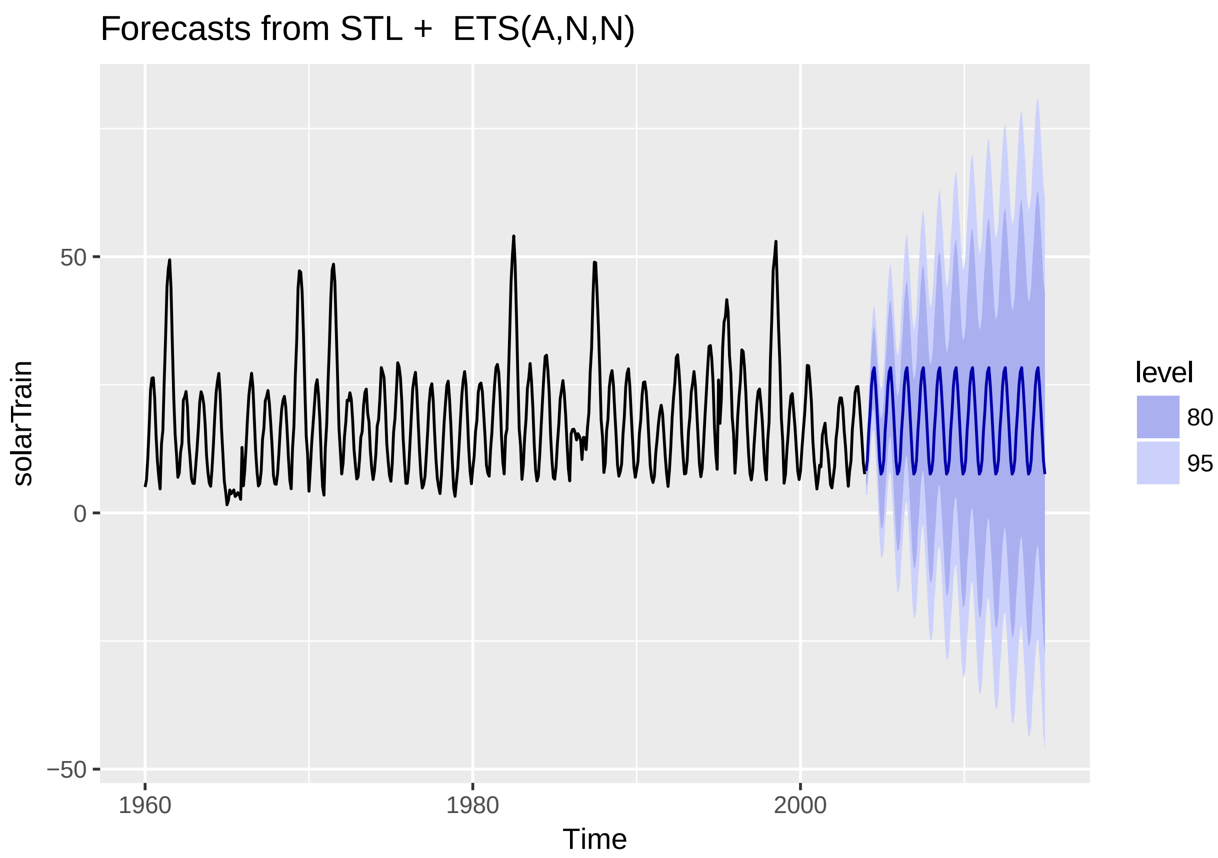
```r
# Generate forecasts for Train Data
stlFcast <- forecast(object = stl,h = h,simulate = TRUE,bootstrap = TRUE)

# Compare Accuracy With Test Set
stlTestAcc <- accuracy(f = stlFcast,x = solarTest)
stlTestAcc
```

```
##                        ME     RMSE      MAE        MPE     MAPE      MASE
## Training set  0.006162009 2.388380 1.520788  -1.484613 12.75084 0.2173236
## Test set     -0.944449702 3.590508 2.663839 -11.493091 19.35647 0.3806677
##                   ACF1 Theil's U
## Training set 0.2005732        NA
## Test set     0.8885955 0.9520292
```

The STL model gives a MASE of 0.380 on the test set.

Forecast Plot

```r
autoplot(object = stlFcast)
```



Observations

The STL Model has almost equivalent performance on the test set as compared to the Linear Model, but gives better residuals although the residuals of this model also have autocorrelation and changing variance and non normality.

Adding the results to Results dataframe.

```r
# Update modelResults

#c("Model","TrainAcc.","TestAcc.","Res.Autocorr","Res.Variance","Res.Timeplot","Res.Mean","Res.Normalit

# Train Data Results
stlResult <- data.frame("STL Model",0.217,0.380,"Present","Present","Pattern",0.00616,"Non Normality",s

names(stlResult) <- c("Model","TrainAcc.","TestAcc.","Res.Autocorr","Res.Variance","Res.Timeplot","Res.


modelResults <- rbind(modelResults,stlResult)


rm(stl,stlFcast,stlResult,stlTestAcc)
```

Holt Winters Model

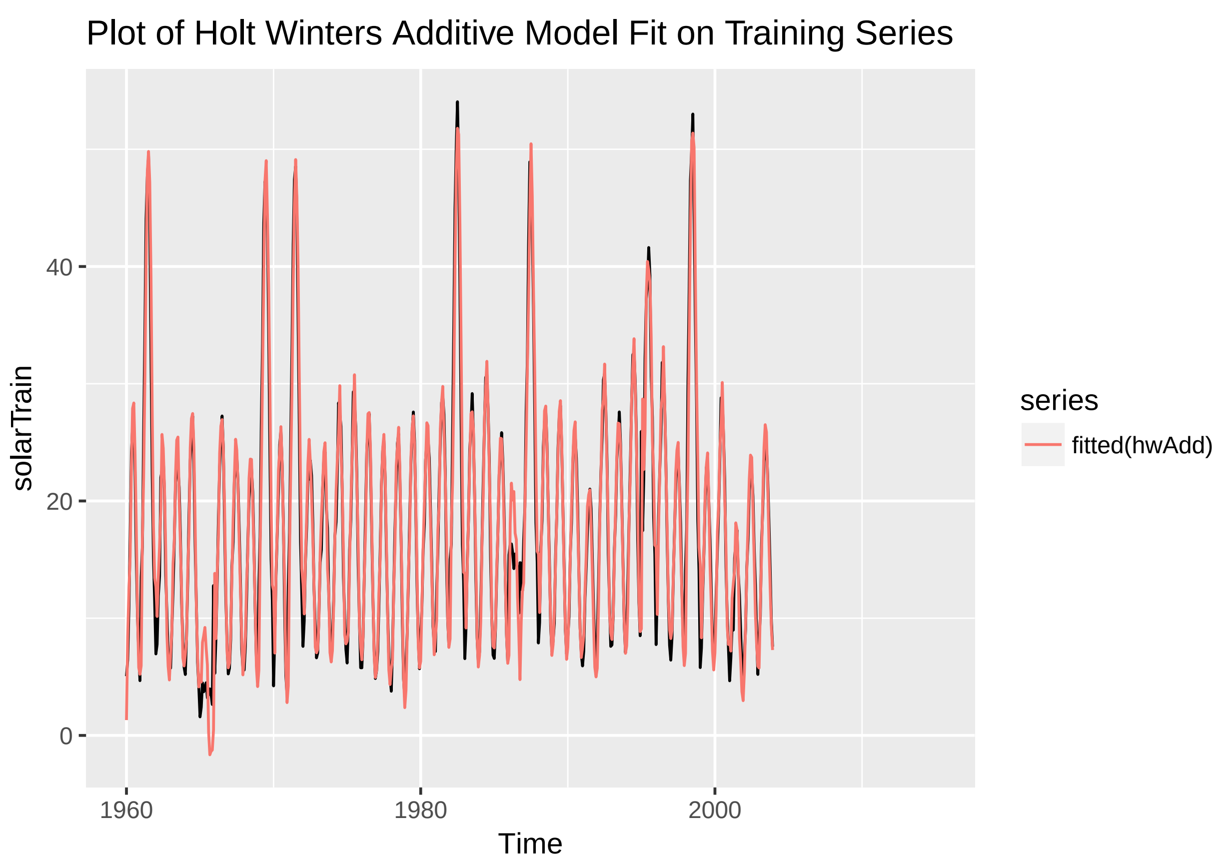
Here we apply the Holt Winters Seasonal Model. There are two models we apply, one with additive seasonality and one with multiplicative.

Model Fitting

```
# Holt Winters Additive Model on Train Data
hwAdd <- hw(y = solarTrain,h = h,seasonal = "additive",initial = "optimal")

# Holt Winters Multiplicative Model on Train Data
hwMult <- hw(y = solarTrain,h = h,seasonal = "multiplicative",initial = "optimal")
```

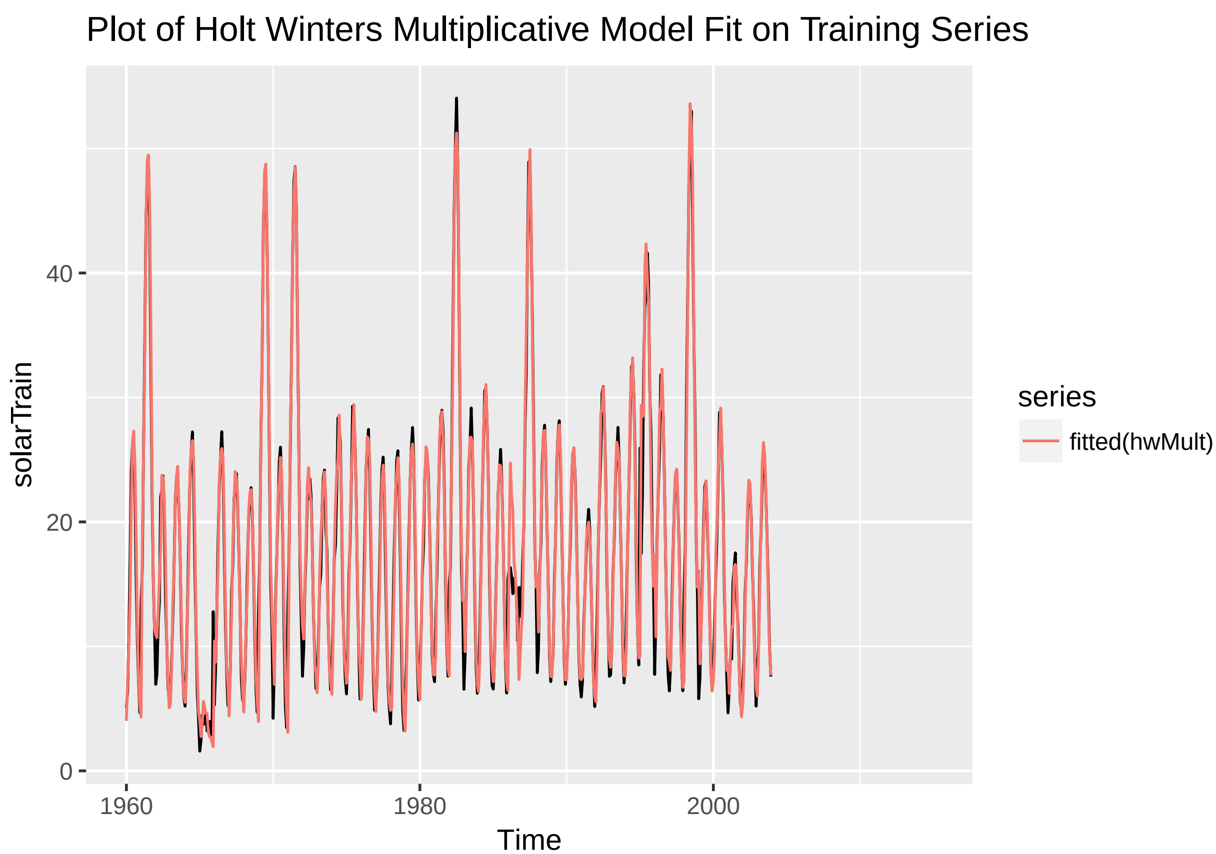
Plot of Model Fit

```
# Plot of Additive Model Fit on Train Data
autoplot(hwAdd,fcol = NA,PI = FALSE) +
    autolayer(object = fitted(hwAdd)) +
    ggtitle("Plot of Holt Winters Additive Model Fit on Training Series")
```




```
# Plot of Multiplicative Model Fit on Train Data
autoplot(hwMult,fcol = NA,PI = FALSE) +
```

```r
    autolayer(object = fitted(hwMult)) +
    ggtitle("Plot of Holt Winters Multiplicative Model Fit on Training Series")
```



Accuracy on Training Data

```r
# Accuracy on Training Set for Additive Model
accuracy(f = hwAdd)# MASE 0.189
```

```
##                      ME     RMSE     MAE       MPE     MAPE      MASE
## Training set -0.1435394 2.498745 1.61087 -2.420677 13.52726 0.2301965
##                   ACF1
## Training set 0.1893765
```

```r
# Accuracy on Training Set for Multiplicative Model
accuracy(f = hwMult)# MASE 0.1871
```

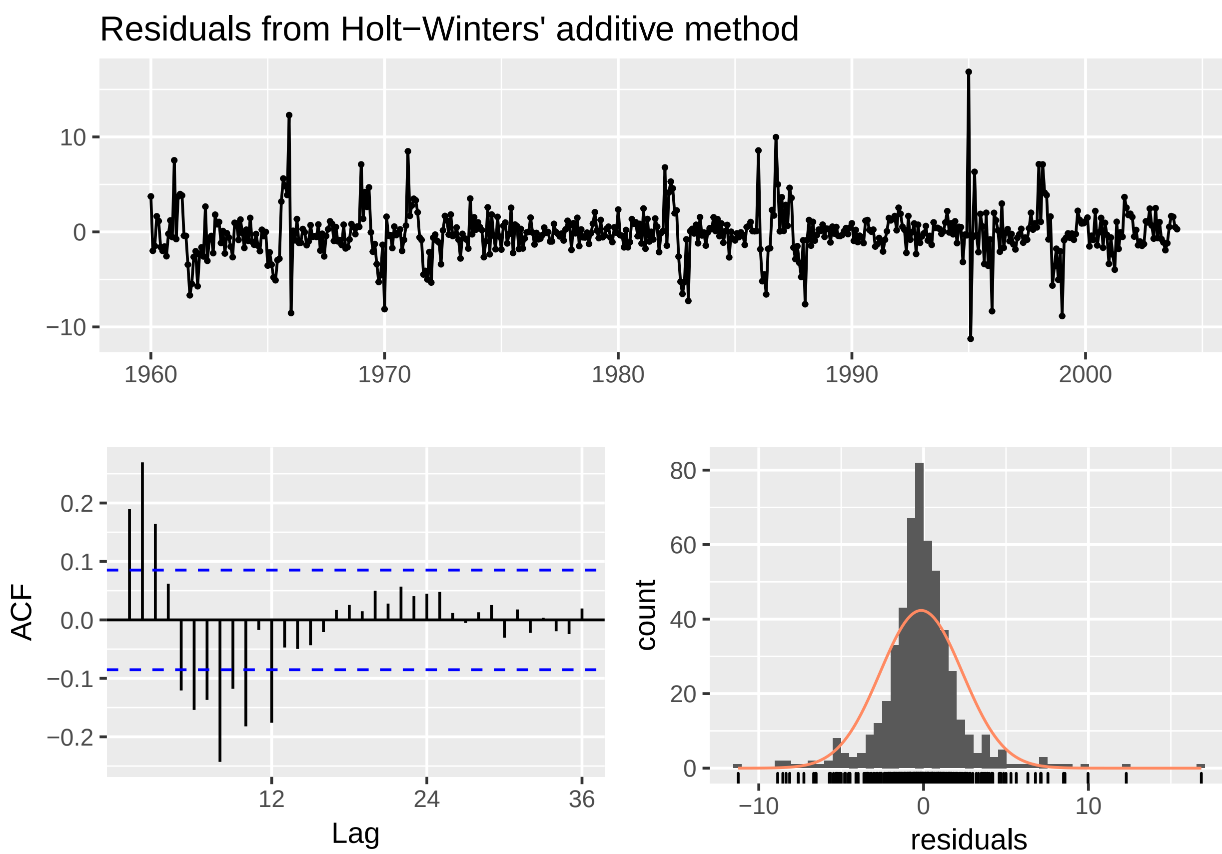
```
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.09352173 2.191704 1.309591 -2.517529 10.79732 0.1871431
##                      ACF1
## Training set -0.005335593
```

The training accuracy of the Additive Model is 0.189 and the Multiplicative Model is 0.187

Residual Plots

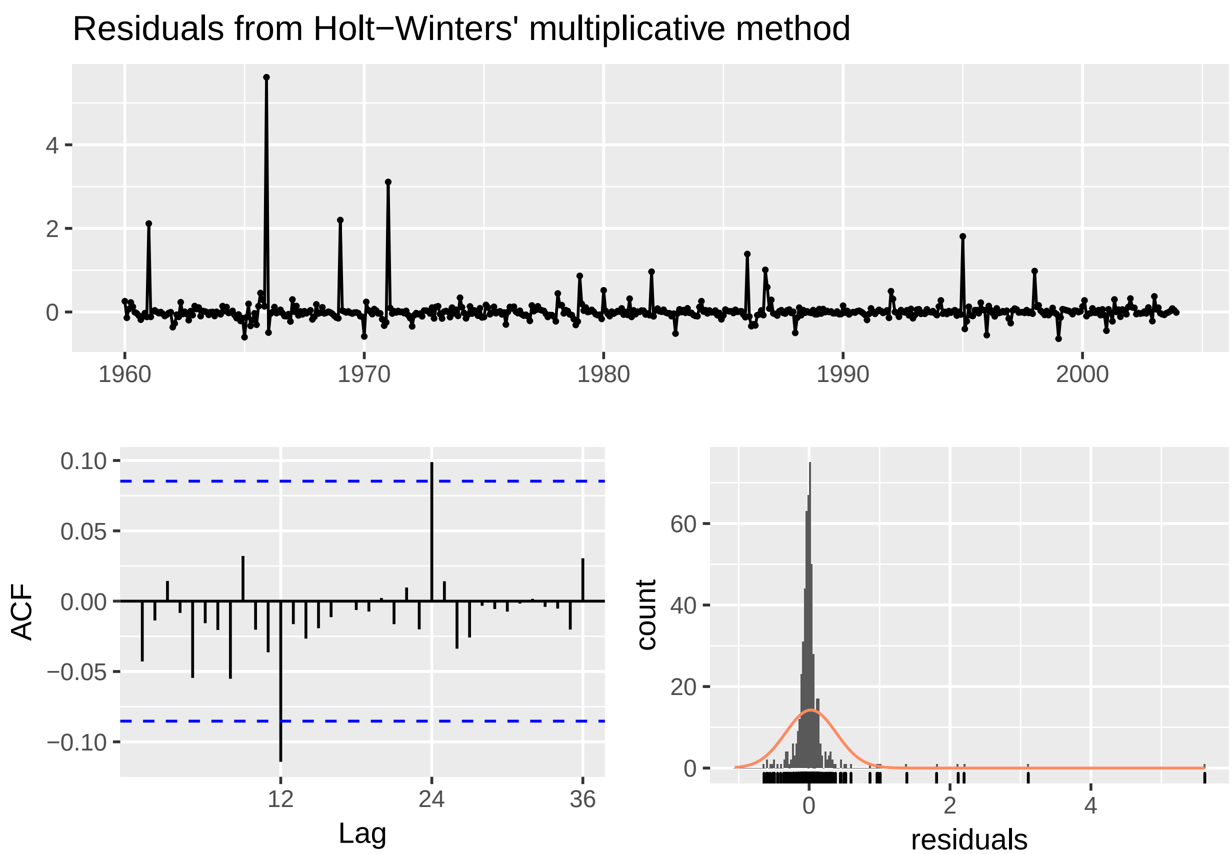
Analyse the residuals plots of the Model Fit.

```
# Residual Plots of Additive Model
checkresiduals(hwAdd)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' additive method
## Q* = 188.91, df = 8, p-value < 2.2e-16
##
## Model df: 16.   Total lags used: 24
```

```
# Residual Plots of Multiplicative Model
checkresiduals(hwMult)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' multiplicative method
## Q* = 20.082, df = 8, p-value = 0.01003
##
## Model df: 16.   Total lags used: 24
```

Both Models have some seasonal autocorrelation left in them as seen with the positive lags in the ACF at Lag 12 and 24 and also indicated in Ljung Box test which is significant at 0.05.

Both Models also show changing variance in the residuals.

Residual Statistics

```r
# Mean of Residuals of Additive Model
mean(resid(hwAdd),na.rm = TRUE)
```

```
## [1] -0.1435394
```

```r
# Normality of Residuals of Additive Model
shapiro.test(x = hwAdd$residuals)# Null: Normality
```

```
##
##  Shapiro-Wilk normality test
##
## data:  hwAdd$residuals
```

```
## W = 0.88886, p-value < 2.2e-16
```

```r
# Mean of Residuals of Multiplicative Model
mean(resid(hwMult),na.rm = TRUE)
```

```
## [1] 0.02493787
```

```r
# Normality of Residuals of Additive Model
shapiro.test(x = hwMult$residuals)# Null: Normality
```

```
##
##  Shapiro-Wilk normality test
##
## data:  hwMult$residuals
## W = 0.3658, p-value < 2.2e-16
```

Both models have non normal residuals and the mean of residuals is almost zero for the multiplicative model but for the additive model -0.1 which may indicate bias in the fit.

Test Prediction

We apply the model on the test set and get accuracy measures.

```r
# Generate forecasts for Additive Model
hwAddFcast <- forecast(object = hwAdd,h = h,simulate = TRUE,bootstrap = TRUE)

# Generate forecasts for Multiplicative Model
hwMultFcast <- forecast(object = hwMult,h = h,simulate = TRUE,bootstrap = TRUE)


# Compare Accuracy of Additive Model With Test Set
hwAddTestAcc <- accuracy(f = hwAddFcast,x = solarTest)
hwAddTestAcc
```

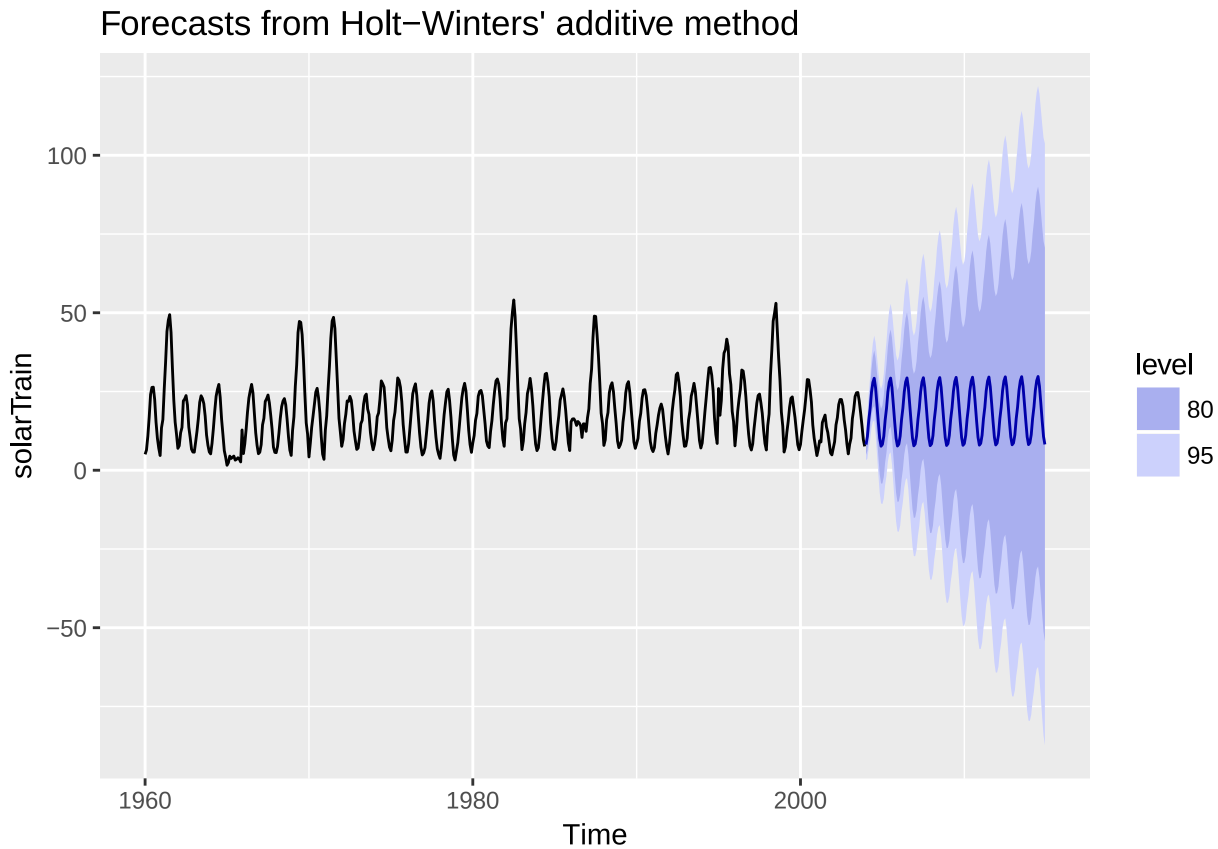
```
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.1435394 2.498745 1.610870  -2.420677 13.52726 0.2301965
## Test set     -1.4721587 3.785931 2.638743 -14.953302 20.23272 0.3770814
##                   ACF1 Theil's U
## Training set 0.1893765        NA
## Test set     0.8994513  1.036343
```

```r
# Compare Accuracy With Multiplicative Model Test Set
hwMultTestAcc <- accuracy(f = hwMultFcast,x = solarTest)
hwMultTestAcc
```

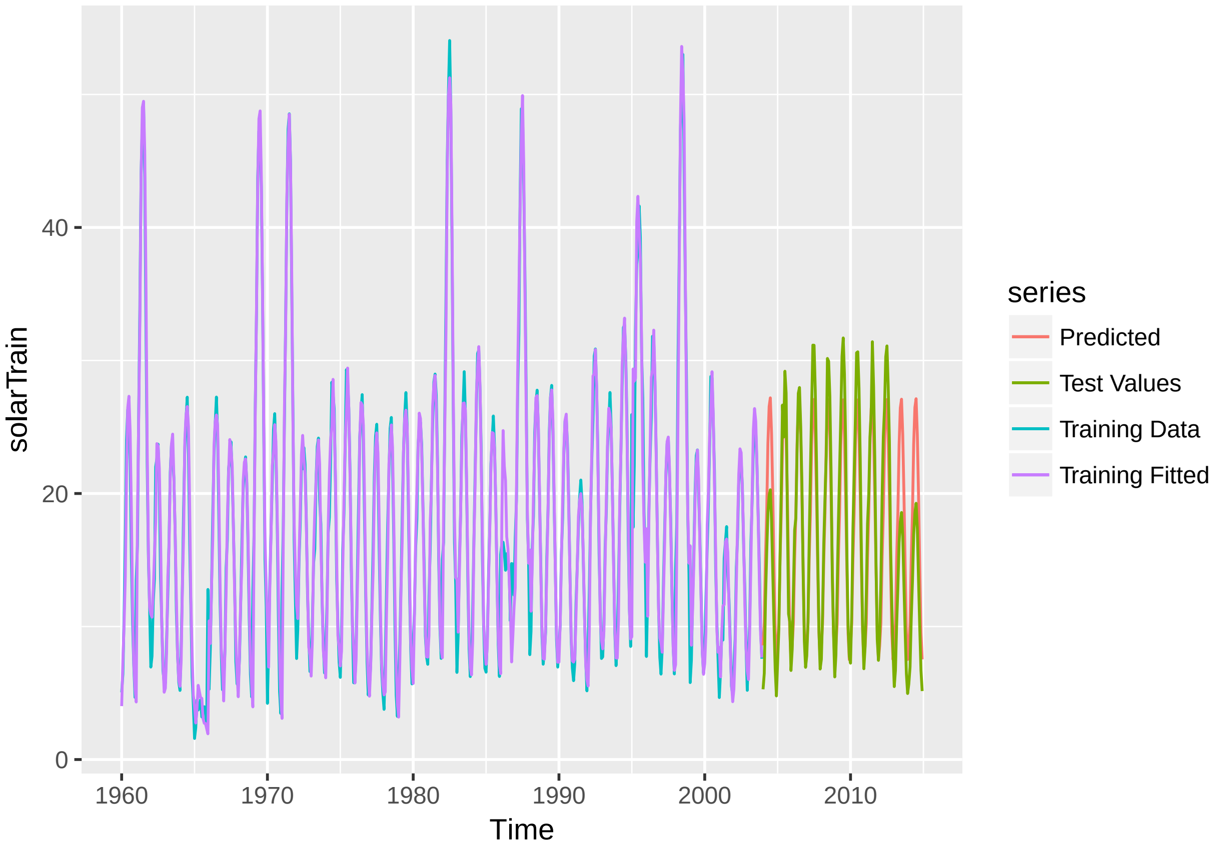
```
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.09352173 2.191704 1.309591 -2.517529 10.79732 0.1871431
## Test set     -0.30204028 3.500766 2.791901 -8.103540 19.38496 0.3989681
##                     ACF1 Theil's U
## Training set -0.005335593        NA
## Test set      0.889322700 0.8954755
```

Forecast Plot

```r
# Plot of Additive Model
autoplot(object = hwAddFcast)
```




```
# Plot of Multiplicative Model
autoplot(object = solarTrain,series = "Training Data") +
    autolayer(hwMult$fitted,series = "Training Fitted") +
    autolayer(hwMultFcast$mean,series = "Predicted") +
    autolayer(solarTest,series = "Test Values")
```



Observations

Adding the results to Results dataframe.

```
# Update modelResults

#c("Model","TrainAcc.","TestAcc.","Res.Autocorr","Res.Variance","Res.Timeplot","Res.Mean","Res.Normalit

# Additive Model Result
hwAddResult <- data.frame("HW Additive",0.23,0.377,"Present","Present","Pattern",-0.143,"Non Normality"

names(hwAddResult) <- c("Model","TrainAcc.","TestAcc.","Res.Autocorr","Res.Variance","Res.Timeplot","Re



# Multiplicative Model Result
hwMultResult <- data.frame("HW Multiplicative",0.187,0.398,"Present","Present","Pattern",0.024,"Non Nor

names(hwMultResult) <- c("Model","TrainAcc.","TestAcc.","Res.Autocorr","Res.Variance","Res.Timeplot","R

modelResults <- rbind(modelResults,hwAddResult,hwMultResult)


rm(hwAdd,hwMult,hwAddFcast,hwMultFcast,hwAddResult,hwMultResult,hwAddTestAcc,hwMultTestAcc)
```

ETS Models

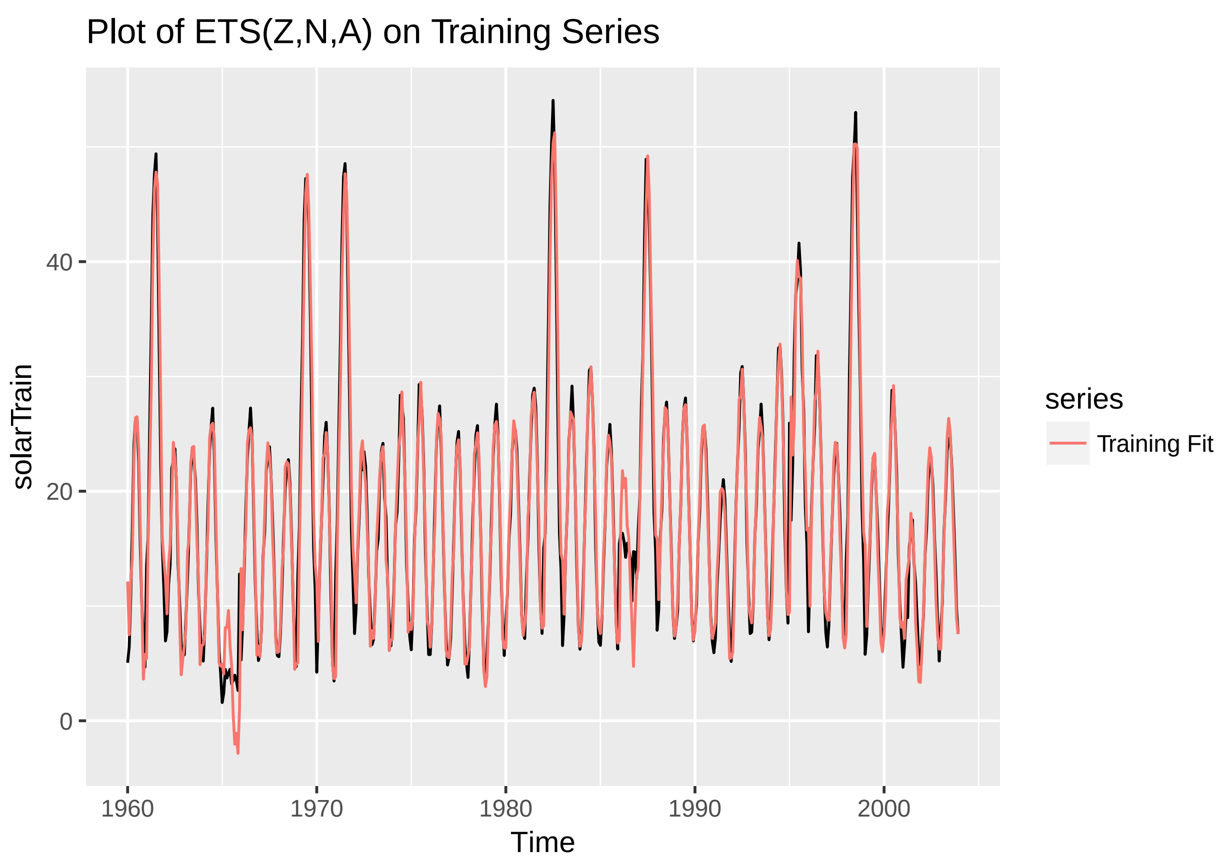
The ETS Models are a group of models based on the Exponential Smoothing models of which Holt Winters is a part. The ETS models also model the error terms and provide different options for the optimization criterion for parameter estimation.

Model Fitting

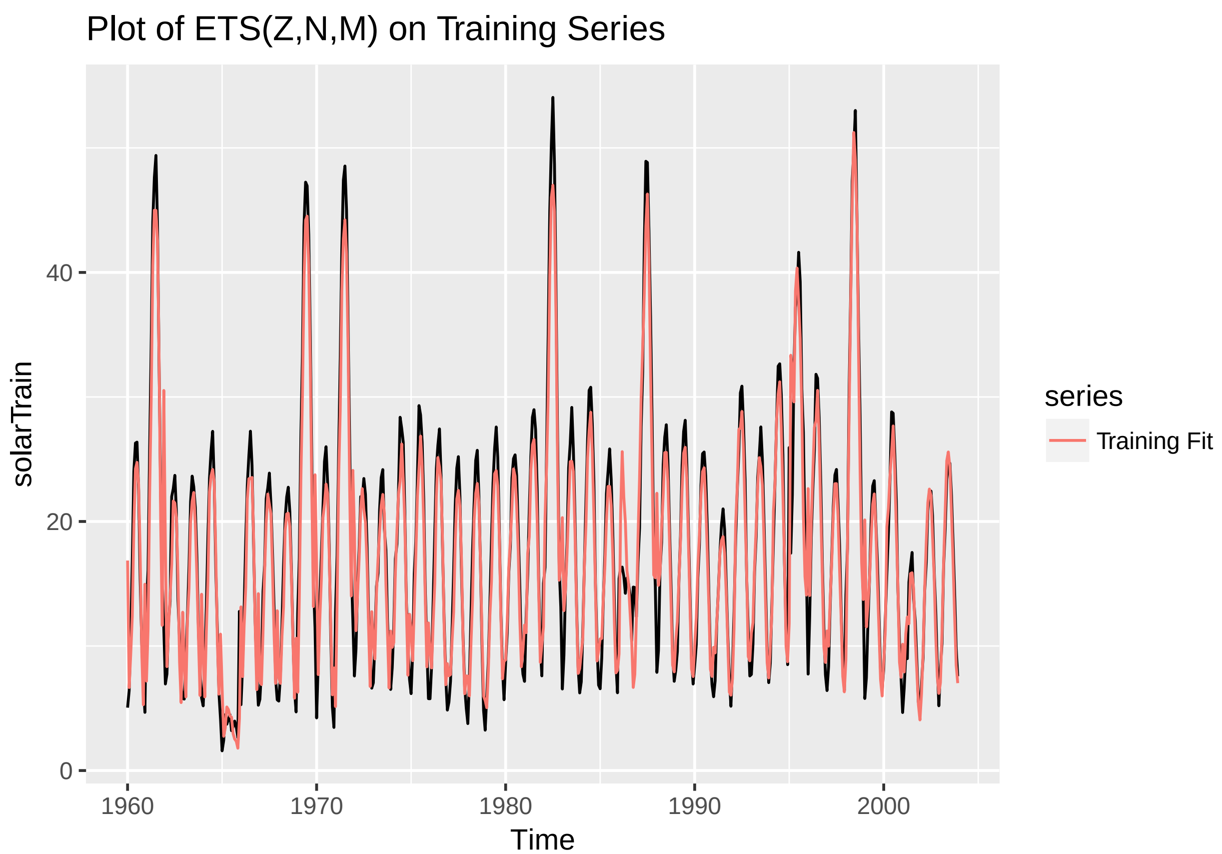
```
# ETS Model (Z,N,A)
etsZNA <- ets(y = solarTrain,model = 'ZNA')

# ETS Model (Z,N,M)
etsZNM <- ets(y = solarTrain,model = 'ZNM')

# Auto Ets
etsauto <- ets(y = solarTrain)
```

Plot of Model Fit

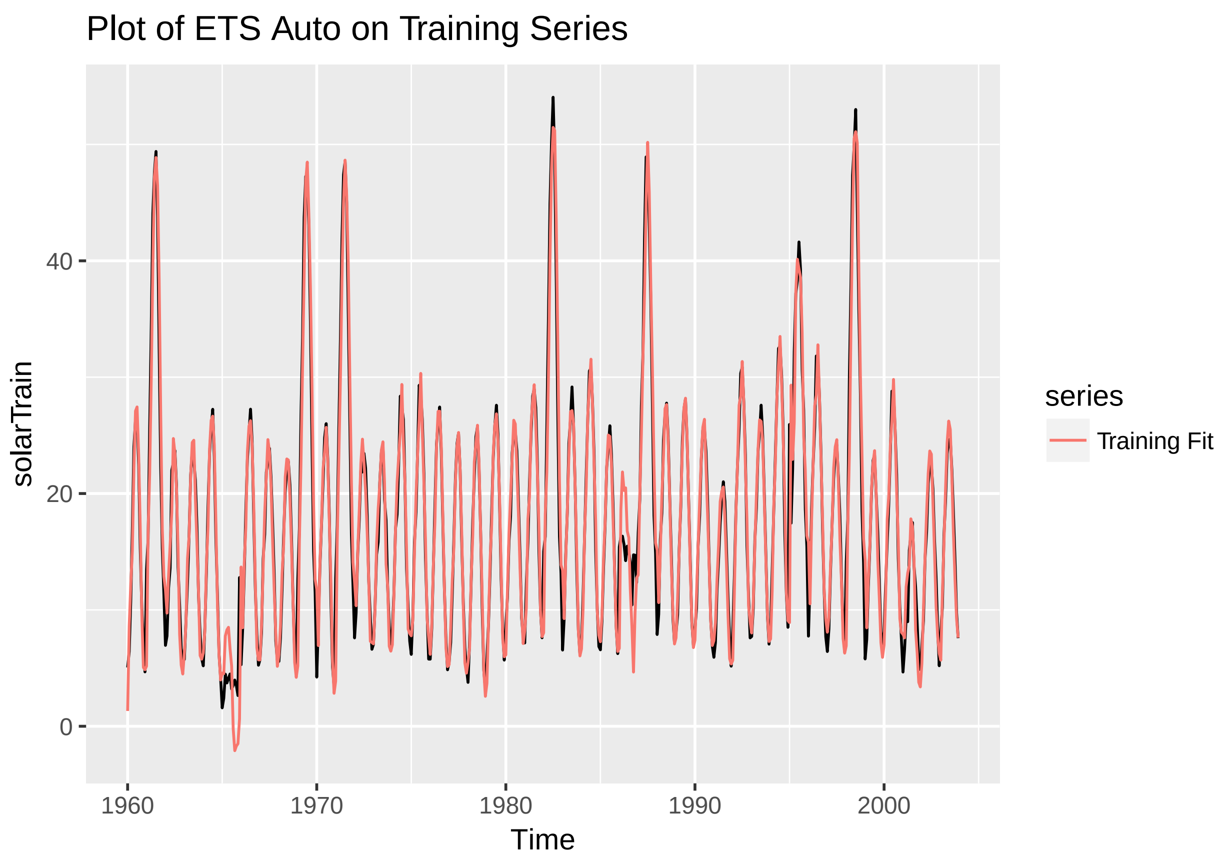
```
# Plot of ETS(Z,N,A)
autoplot(solarTrain,fcol = NA,PI = FALSE) +
    autolayer(fitted(etsZNA),series = "Training Fit") +
 ggtitle("Plot of ETS(Z,N,A) on Training Series")
```



```r
# Plot of ETS(Z,N,M)
autoplot(solarTrain,fcol = NA,PI = FALSE) +
    autolayer(fitted(etsZNM),series = "Training Fit") +
 ggtitle("Plot of ETS(Z,N,M) on Training Series")
```




```r
# Plot of ETS()
autoplot(solarTrain,fcol = NA,PI = FALSE) +
    autolayer(fitted(etsauto),series = "Training Fit") +
 ggtitle("Plot of ETS Auto on Training Series")
```



Accuracy on Training Data

```
# Accuracy of ETS(Z,N,A)
accuracy(f = etsZNA)# MASE 0.2366
```

```
##                        ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.006301355 2.536733 1.655721 -2.126895 14.11349 0.2366057
##                   ACF1
## Training set 0.1913559
```

```
# Accuracy of ETS(Z,N,M)
accuracy(f = etsZNM)# MASE 0.3267
```

```
##                     ME     RMSE      MAE       MPE    MAPE      MASE
## Training set 0.3919208 3.383854 2.286655 -5.916812 20.2172 0.3267674
##                   ACF1
## Training set 0.2817463
```

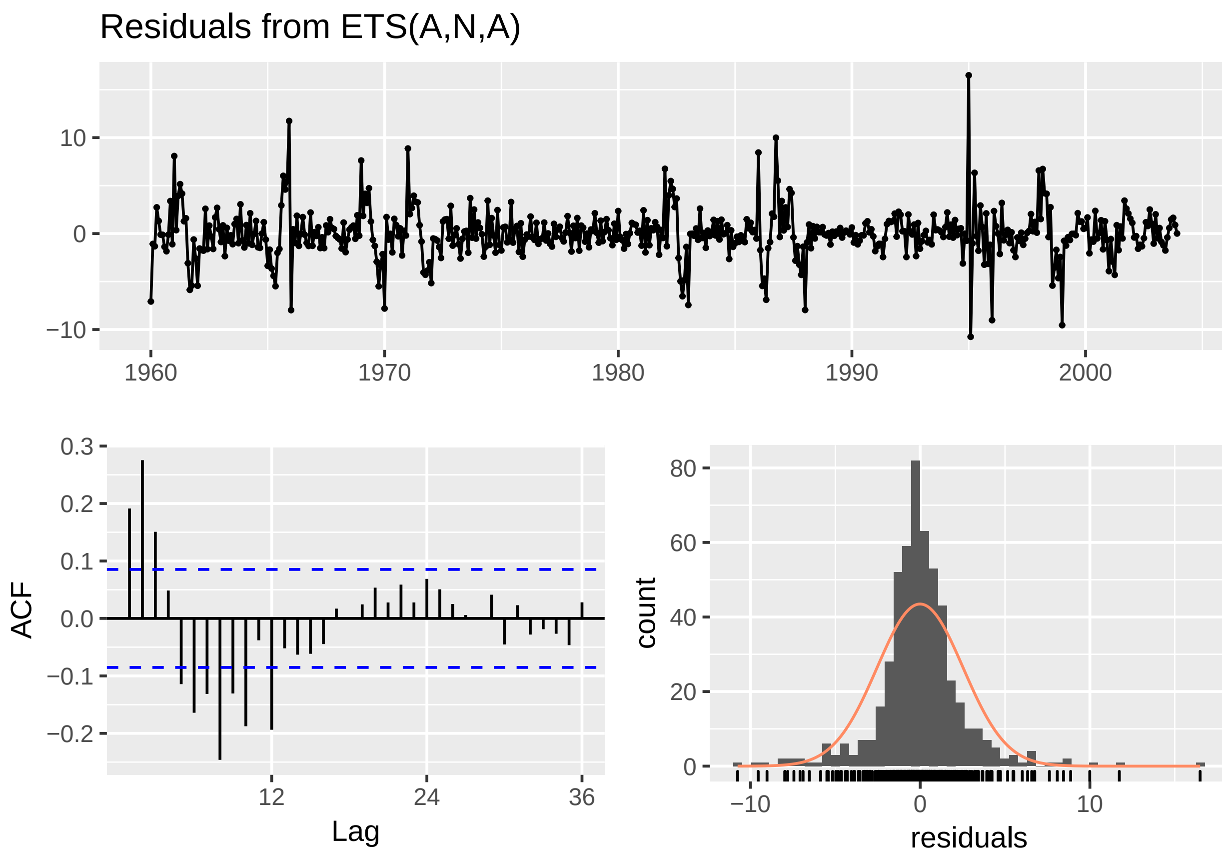
```
# Accuracy of ETS Auto
accuracy(f = etsauto)# MASE 0.2227
```

```
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.007964832 2.460185 1.565806 -1.408905 13.15062 0.2237567
##                   ACF1
## Training set 0.1531124
```

Residual Plots

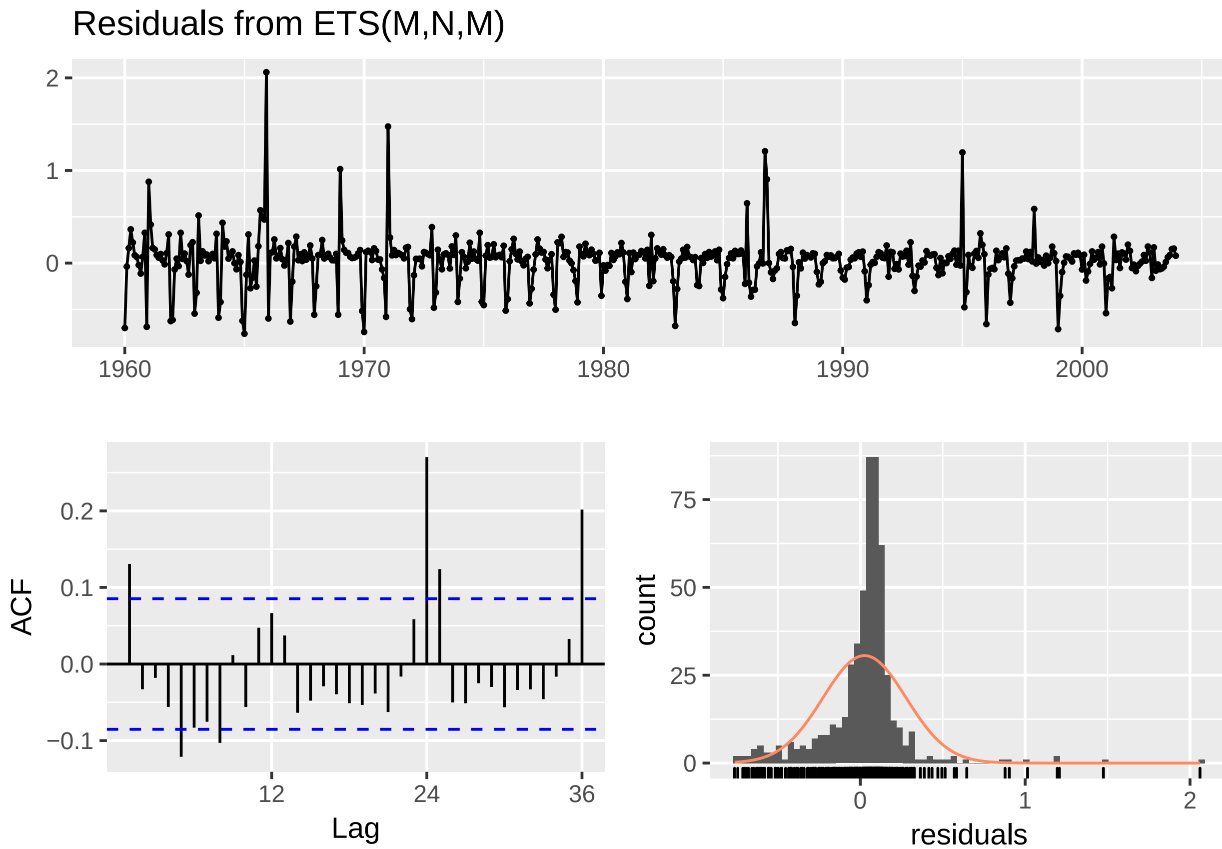
Analyse the residuals plots of the Model Fit.

```
# Residual Plots of ETS(Z,N,A) Model
checkresiduals(etsZNA)
```



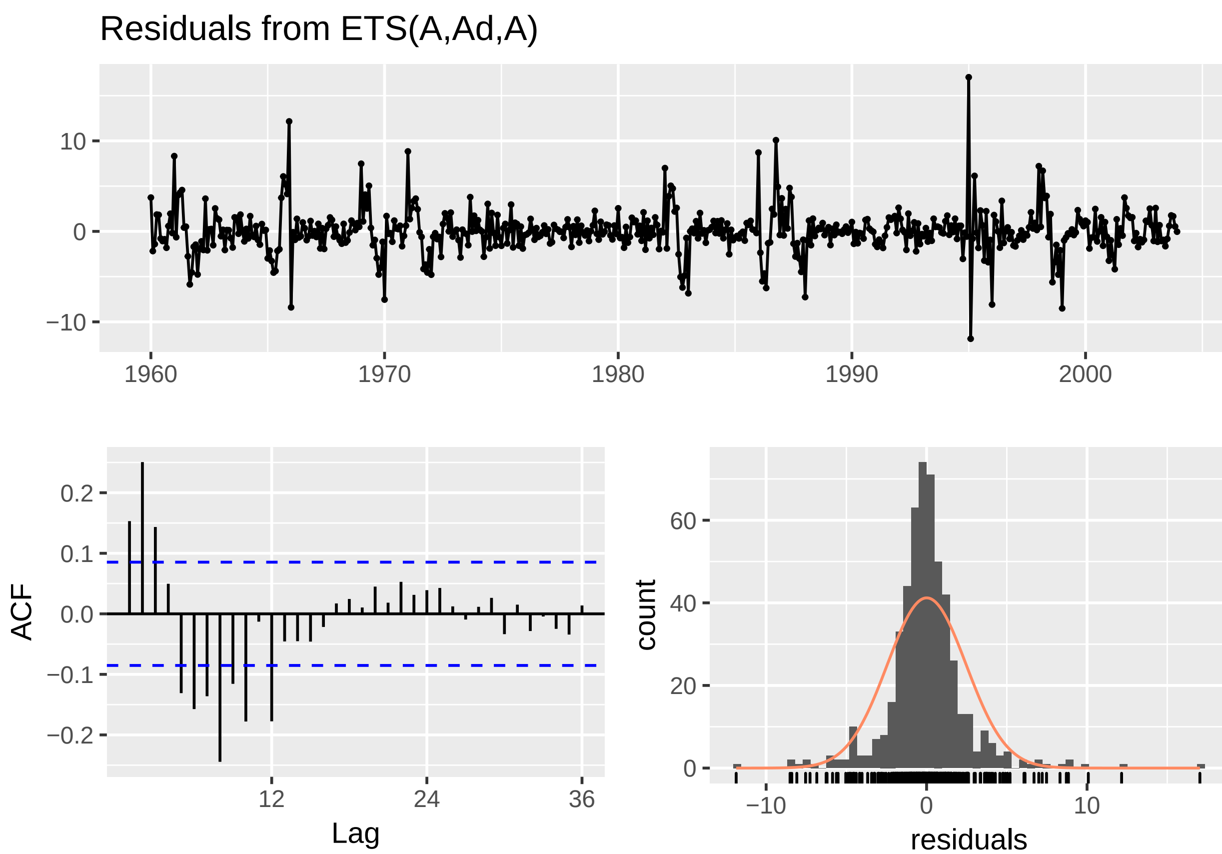

```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,A)
## Q* = 200.1, df = 10, p-value < 2.2e-16
##
## Model df: 14.   Total lags used: 24
```

```
# Residual Plots of ETS(Z,N,M) Model
checkresiduals(etsZNM)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,M)
## Q* = 91.317, df = 10, p-value = 2.887e-15
##
## Model df: 14.   Total lags used: 24
```

```
# Residual Plots of ETS Auto Model
checkresiduals(etsauto)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,Ad,A)
## Q* = 172.64, df = 7, p-value < 2.2e-16
##
## Model df: 17.   Total lags used: 24
```

Residual Statistics

```
# Mean and Normality of Residuals of ETS(Z,N,A)
mean(resid(etsZNA),na.rm = TRUE);shapiro.test(x = resid(etsZNA))
```

```
## [1] -0.006301355
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(etsZNA)
## W = 0.90139, p-value < 2.2e-16
```

```
# Mean and Normality of Residuals of ETS(Z,N,M)
mean(resid(etsZNM),na.rm = TRUE);shapiro.test(x = resid(etsZNM))
```

```
## [1] 0.02563535
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  resid(etsZNM)
## W = 0.79246, p-value < 2.2e-16
```

```r
# Mean and Normality of Residuals of ETS Auto
mean(resid(etsauto),na.rm = TRUE);shapiro.test(x = resid(etsauto))
```

```
## [1] 0.007964832
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(etsauto)
## W = 0.8817, p-value < 2.2e-16
```

Test Prediction

We apply the model on the test set and get accuracy measures.

```r
# Generate forecasts for ETS(Z,N,A) Model
etsZNAFcast <- forecast(object = etsZNA,h = h,simulate = TRUE,bootstrap = TRUE)

# Generate forecasts for ETS(Z,N,M) Model
etsZNMFcast <- forecast(object = etsZNM,h = h,simulate = TRUE,bootstrap = TRUE)


# Generate forecasts for ETS Auto Model
etsAutoFcast <- forecast(object = etsauto,h = h,simulate = TRUE,bootstrap = TRUE)


# Compare Accuracy of ETS(Z,N,A)
etsZNATestAcc <- accuracy(f = etsZNAFcast,x = solarTest)
etsZNATestAcc
```

```
##                        ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.006301355 2.536733 1.655721  -2.126895 14.11349 0.2366057
## Test set     -1.526287141 3.793348 2.636685 -14.768241 19.92244 0.3767873
##                   ACF1 Theil's U
## Training set 0.1913559        NA
## Test set     0.8910968  1.044685
```

```r
# Compare Accuracy of ETS(Z,N,M)
etsZNMTestAcc <- accuracy(f = etsZNMFcast,x = solarTest)
etsZNMTestAcc
```

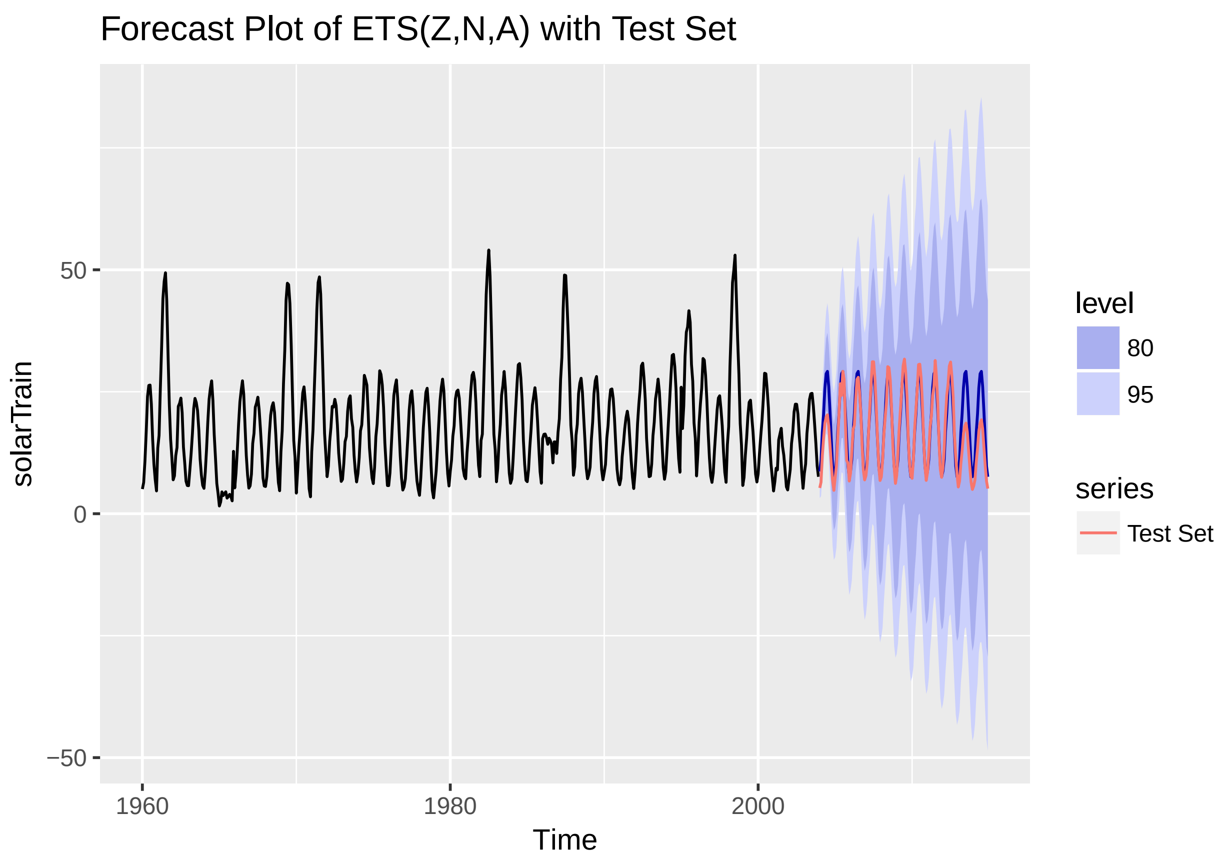
```
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set  0.3919208 3.383854 2.286655  -5.916812 20.21720 0.3267674
## Test set     -5.8758768 7.697715 5.913800 -38.014689 38.39193 0.8450935
##                   ACF1 Theil's U
## Training set 0.2817463        NA
## Test set     0.8684003  1.940096
```

```r
# Compare Accuracy of ETS Auto
etsAutoTestAcc <- accuracy(f = etsAutoFcast,x = solarTest)
etsAutoTestAcc
```

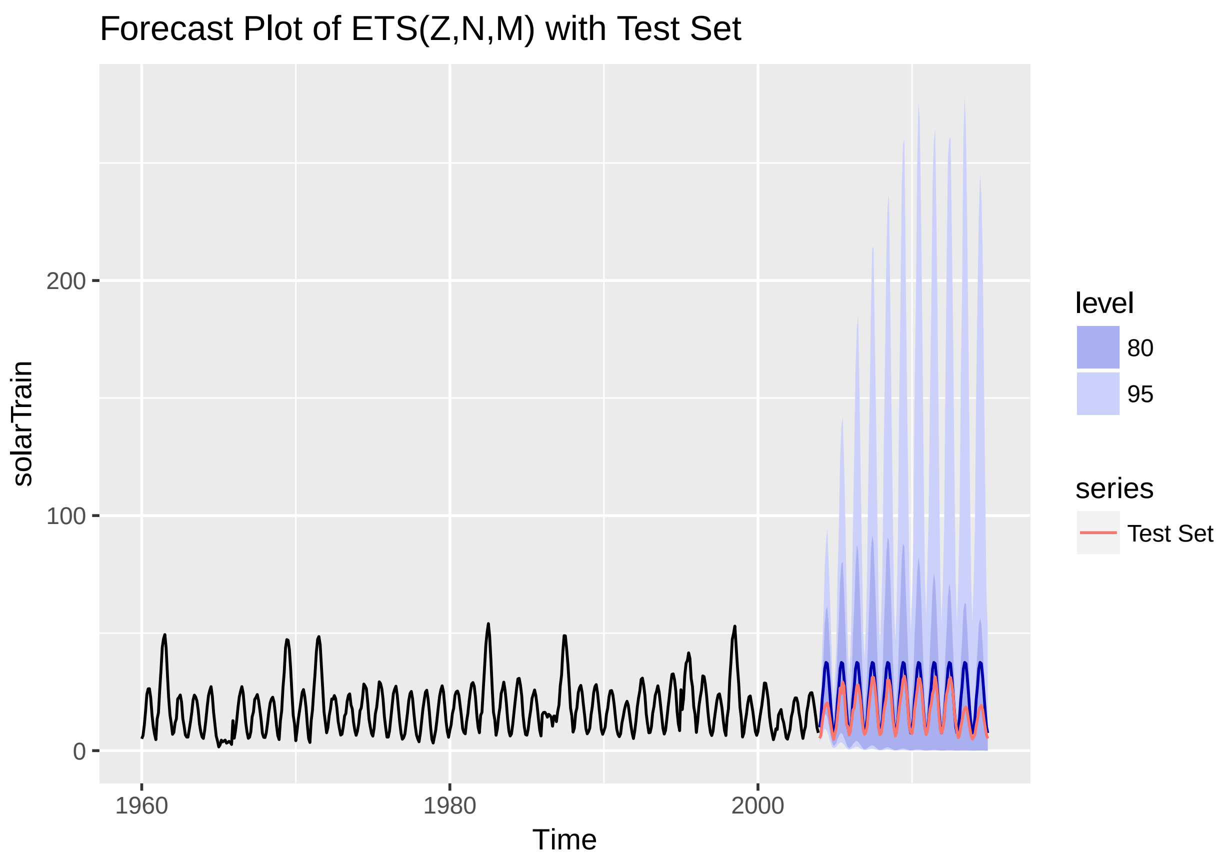
```
##                        ME     RMSE      MAE        MPE     MAPE      MASE
## Training set  0.007964832 2.460185 1.565806  -1.408905 13.15062 0.2237567
## Test set     -1.604026423 3.807692 2.624425 -15.244342 19.95740 0.3750354
##                   ACF1 Theil's U
## Training set 0.1531124        NA
## Test set     0.8977629  1.050822
```

Forecast Plot

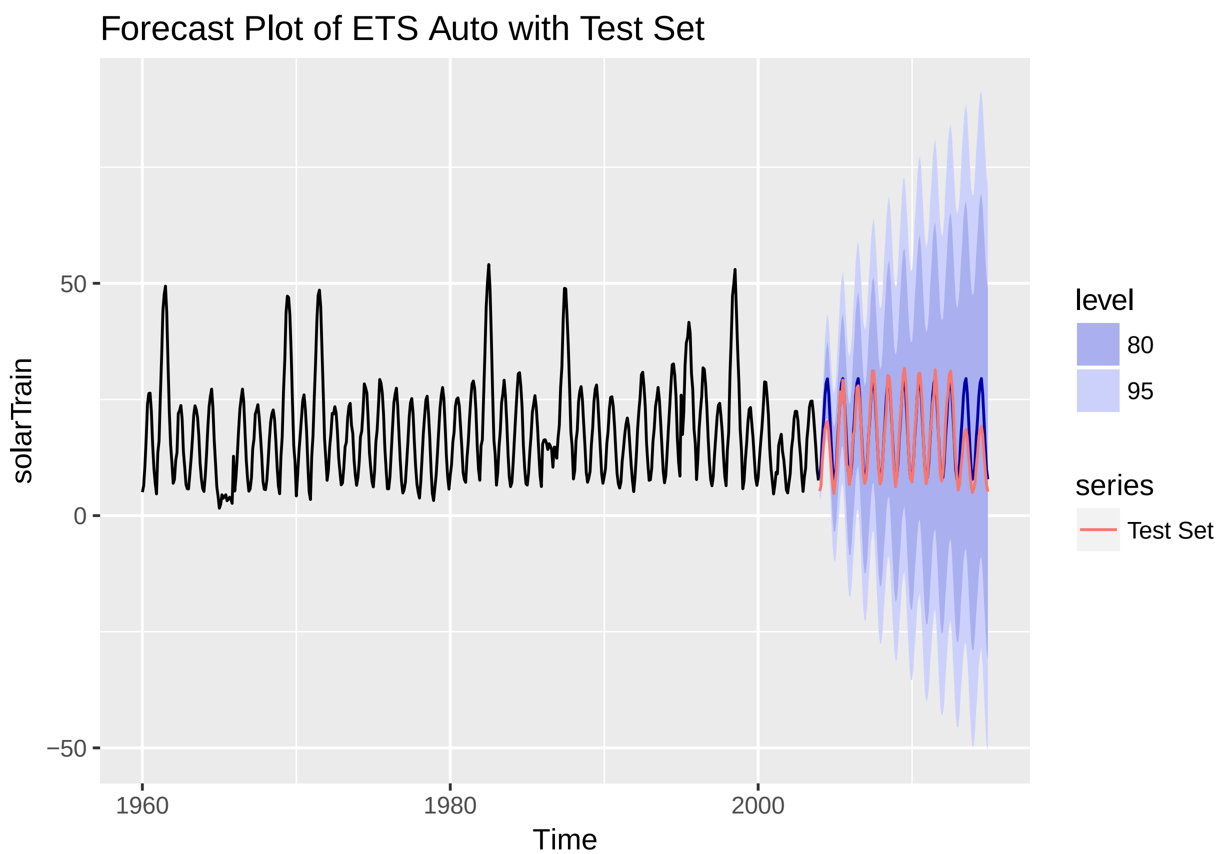
```r
# Plot of ETS(Z,N,A)
autoplot(object = etsZNAFcast) +
    autolayer(solarTest,series = "Test Set") +
    ggtitle("Forecast Plot of ETS(Z,N,A) with Test Set")
```




```r
# Plot of ETS(Z,N,M)
autoplot(object = etsZNMFcast) +
    autolayer(solarTest,series = "Test Set") +
    ggtitle("Forecast Plot of ETS(Z,N,M) with Test Set")
```




```r
# Plot of ETS Auto
autoplot(object = etsAutoFcast) +
    autolayer(solarTest,series = "Test Set") +
    ggtitle("Forecast Plot of ETS Auto with Test Set")
```



Observations

Adding the results to Results dataframe.

```
# Update modelResults

#c("Model","TrainAcc.","TestAcc.","Res.Autocorr","Res.Variance","Res.Timeplot","Res.Mean","Res.Normalit

# ETS(Z,N,A)
etsZNAResult <- data.frame("ETS(Z,N,A)",0.2366,0.3767,"Present","Present","Pattern",-0.006,"Non Normali

names(etsZNAResult) <-c("Model","TrainAcc.","TestAcc.","Res.Autocorr","Res.Variance","Res.Timeplot","Re

# ETS(Z,N,M)
etsZNMResult <- data.frame("ETS(Z,N,M)",0.3267,0.8450,"Present","Present","Pattern",0.025,"Non Normalit

names(etsZNMResult) <-c("Model","TrainAcc.","TestAcc.","Res.Autocorr","Res.Variance","Res.Timeplot","Re

# ETS Auto
etsAutoResult <- data.frame("ETS Auto",0.2237,0.3750,"Present","Present","Pattern",0.007,"Non Normality

names(etsAutoResult) <-c("Model","TrainAcc.","TestAcc.","Res.Autocorr","Res.Variance","Res.Timeplot","R
```

```
modelResults <- rbind(modelResults,etsZNAResult,etsZNMResult,etsAutoResult)

rm(etsZNA,etsZNM,etsauto,etsZNAFcast,etsZNMFcast,etsAutoFcast,etsZNAResult,etsZNMResult,etsAutoResult,et
```

ARIMA

A no of ARIMA Models were fit with different values of AR and MA for the seasonal characteristics. The best model is demonstrated here.

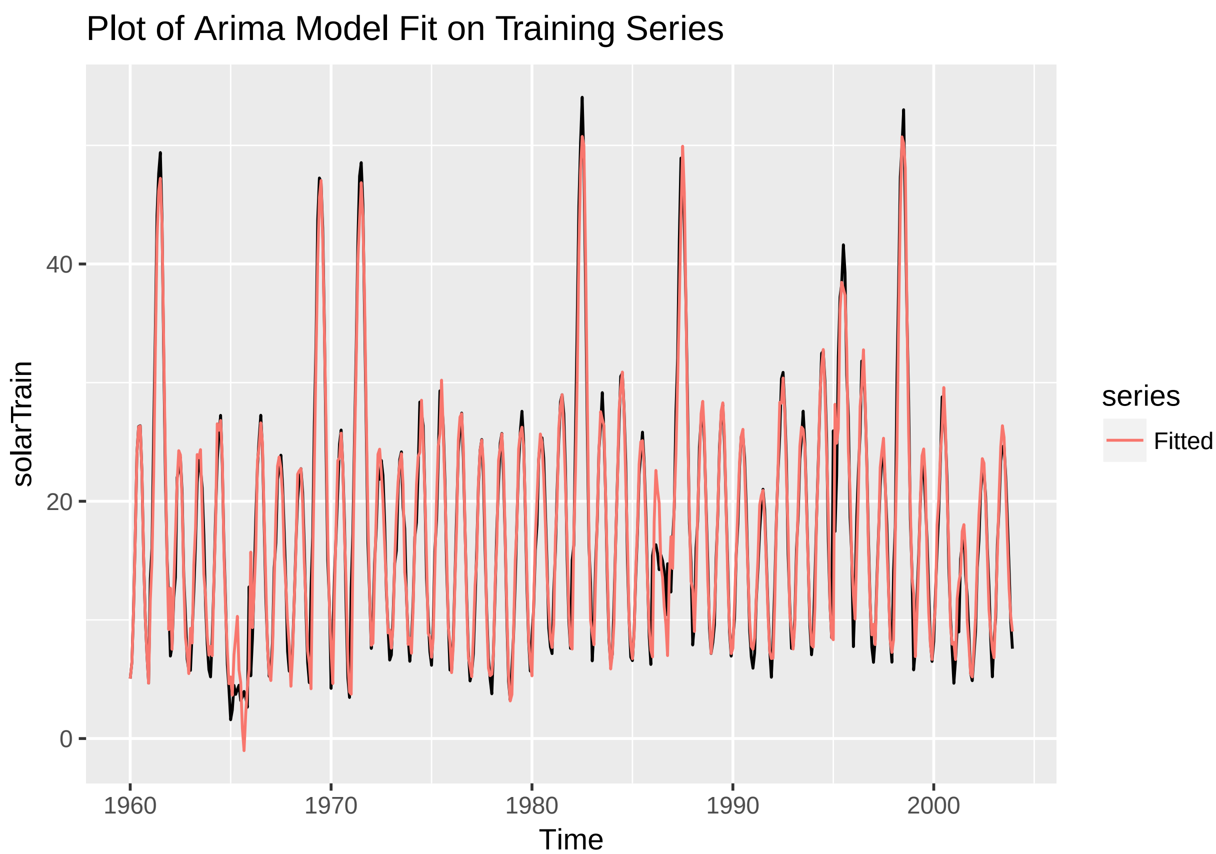
Model Fitting

```
# ARIMA Model
Arima <- Arima(y = solarTrain,order = c(3,0,2),seasonal = c(1,1,1))
summary(Arima)
```

```
## Series: solarTrain
## ARIMA(3,0,2)(1,1,1)[12]
##
## Coefficients:
##          ar1      ar2     ar3      ma1     ma2     sar1     sma1
##       2.1785  -1.7501  0.5207  -1.1533  0.6910  -0.1667  -0.9702
## s.e.  0.1266   0.2337  0.1173   0.1110  0.0834   0.0564   0.0389
##
## sigma^2 estimated as 4.917:  log likelihood=-1159.04
## AIC=2334.08   AICc=2334.37   BIC=2368.05
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.05395142 2.177221 1.420674 -2.634651 11.80047 0.2030171
##                      ACF1
## Training set -0.005893674
```

Plot of Model Fit

```
# Plot of Model Fit on Train Data
autoplot(solarTrain,fcol = NA,PI = FALSE) +
    autolayer(Arima$fitted,series = "Fitted") +
    ggtitle("Plot of Arima Model Fit on Training Series")
```



Accuracy on Training Data

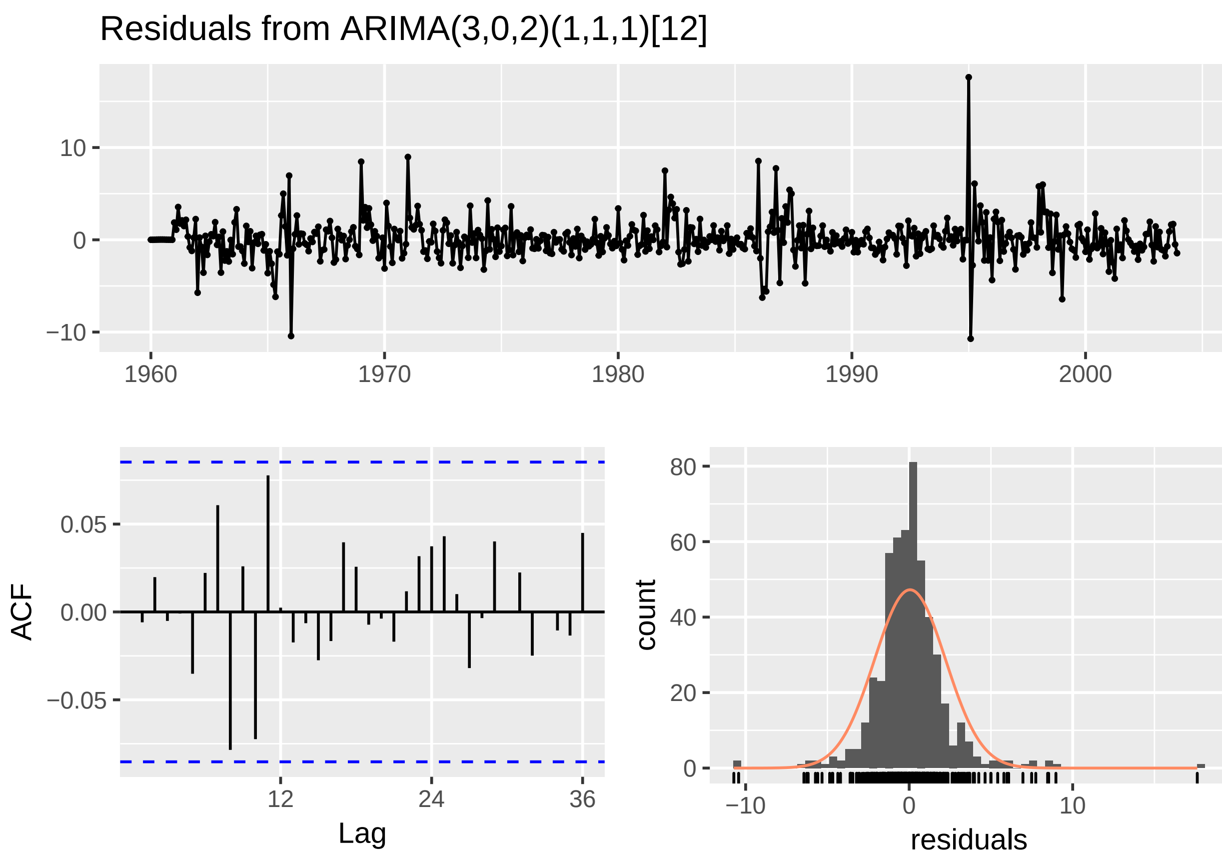
```
# Accuracy on Training Set for Train Data
accuracy(f = Arima)# MASE 0.2030
```

```
##                        ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.05395142 2.177221 1.420674 -2.634651 11.80047 0.2030171
##                       ACF1
## Training set -0.005893674
```

Residual Plots

Analyse the residuals plots of the Model Fit.

```
# Residual Plots of Train Data
checkresiduals(Arima)# No AutoCorr and No Pattern
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,0,2)(1,1,1)[12]
## Q* = 16.513, df = 17, p-value = 0.4878
##
## Model df: 7.   Total lags used: 24
```

Residual Statistics

```
# Mean of Residuals of Train Data
mean(resid(Arima),na.rm = TRUE) # 0.0539
```

```
## [1] 0.05395142
```

```
# Normality of Residuals of Train Data
shapiro.test(x = Arima$residuals)# Null: Normality # Non normal
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Arima$residuals
## W = 0.87754, p-value < 2.2e-16
```

Test Prediction

We apply the model on the test set and get accuracy measures.

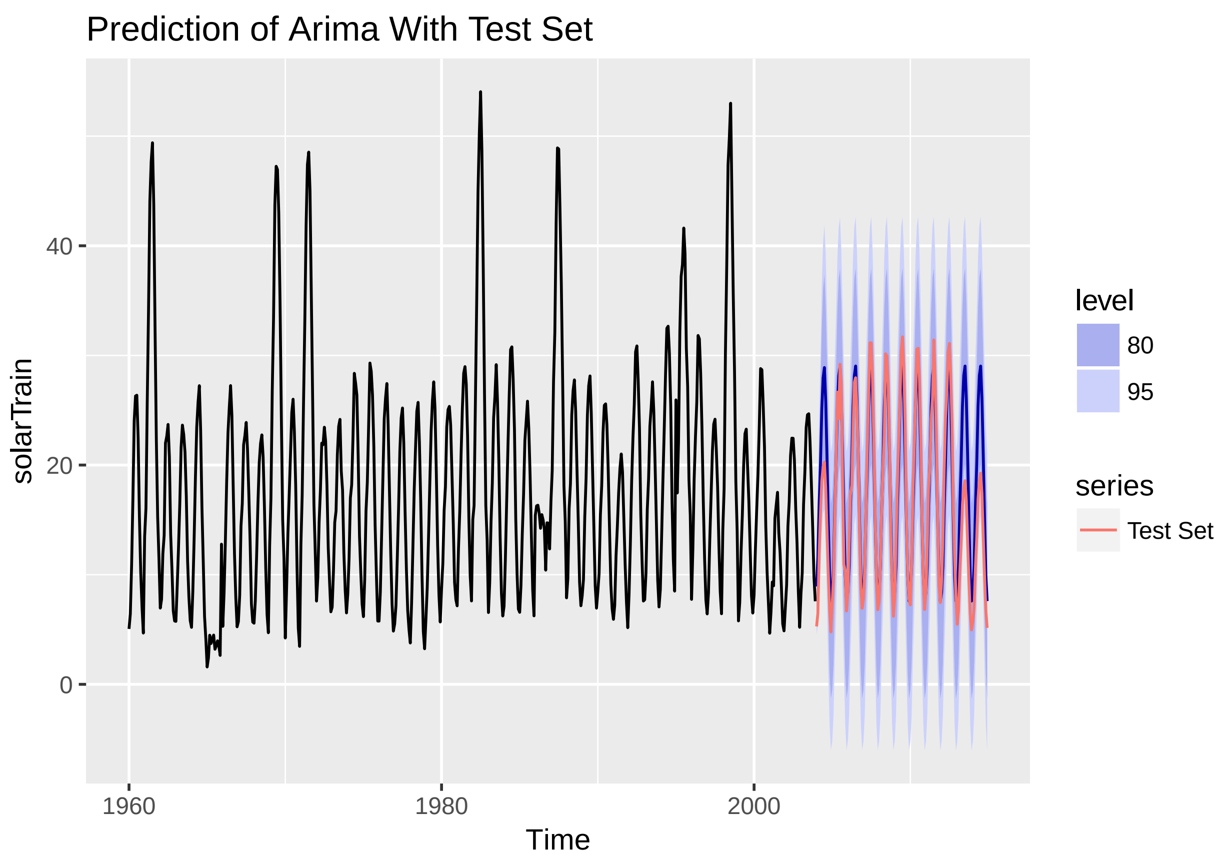
```
# Generate forecasts
ArimaFcast <- forecast(object = Arima,h = h)

# Compare Accuracy With Test Set
ArimaTestAcc <- accuracy(f = ArimaFcast,x = solarTest)
ArimaTestAcc
```

```
##                        ME     RMSE      MAE        MPE     MAPE      MASE
## Training set  0.05395142 2.177221 1.420674  -2.634651 11.80047 0.2030171
## Test set     -1.30499487 3.691279 2.611606 -13.466548 19.56148 0.3732035
##                      ACF1 Theil's U
## Training set -0.005893674        NA
## Test set      0.896156622  1.009999
```

Mase Value on Test Set of 0.3732

Forecast Plot

```
autoplot(object = ArimaFcast)+
    autolayer(solarTest,series = "Test Set") +
    ggtitle("Prediction of Arima With Test Set")
```



Observations

Adding the results to Results dataframe.

```
# Update modelResults

#c("Model","TrainAcc.","TestAcc.","Res.Autocorr","Res.Variance","Res.Timeplot","Res.Mean","Res.Normality

# Train Data Results
ArimaResult <- data.frame(" Arima",0.2030,0.3732,"Not Present","Present","No Pattern",0.053,"Non Normali

names(ArimaResult) <- c("Model","TrainAcc.","TestAcc.","Res.Autocorr","Res.Variance","Res.Timeplot","Res


modelResults <- rbind(modelResults,ArimaResult)


rm(Arima,ArimaFcast,ArimaResult,ArimaTestAcc)
```

Final Model

Different models were fit which gave different MASE values and different residuals. The best models in terms of MASE on Test Set are given below:

```
#Sort Models on MASE on Test Set
modelResults %>% arrange(TestAcc.)
```

```
##               Model TrainAcc. TestAcc. Res.Autocorr Res.Variance
## 1      Linear Model    0.5615   0.3700      Present      Present
## 2             Arima    0.2030   0.3732  Not Present      Present
## 3          ETS Auto    0.2237   0.3750      Present      Present
## 4         ETS(Z,N,A)    0.2366   0.3767      Present      Present
## 5       HW Additive    0.2300   0.3770      Present      Present
## 6         STL Model    0.2170   0.3800      Present      Present
## 7 HW Multiplicative    0.1870   0.3980      Present      Present
## 8    Seasonal Naive    1.0000   0.4240      Present      Present
## 9         ETS(Z,N,M)    0.3267   0.8450      Present      Present
##   Res.Timeplot   Res.Mean Res.Normality
## 1      Pattern  2.414e-16 Non Normality
## 2   No Pattern  5.300e-02 Non Normality
## 3      Pattern  7.000e-03 Non Normality
## 4      Pattern -6.000e-03 Non Normality
## 5      Pattern -1.430e-01 Non Normality
## 6      Pattern  6.160e-03 Non Normality
## 7      Pattern  2.400e-02 Non Normality
## 8      Pattern  4.500e-02 Non Normality
## 9      Pattern  2.500e-02 Non Normality
```

The best two models were the Linear Model and Arima Model in terms of MASE on test set. We select the ARIMA Model as the best model because it has good residuals in terms of autocorrelation even though its accuracy on test set is marginally worse (0.003).

We can deal with the non normality in ARIMA model by generating prediction intervals using bootstrap method. The mean of the ARIMA model residuals is 0.053, which means the forecasts are slightly biased. We can deal with this by adding the mean of residuals to the forecasts.

We do the following below:

Creating the Model

```r
# Model Fit
Arima <- Arima(y = solarTrain,order = c(3,0,2),seasonal = c(1,1,1))

# Summary of Model
summary(Arima)
```

```
## Series: solarTrain
## ARIMA(3,0,2)(1,1,1)[12]
##
## Coefficients:
##          ar1      ar2     ar3      ma1     ma2     sar1     sma1
##       2.1785  -1.7501  0.5207  -1.1533  0.6910  -0.1667  -0.9702
## s.e.  0.1266   0.2337  0.1173   0.1110  0.0834   0.0564   0.0389
##
## sigma^2 estimated as 4.917:  log likelihood=-1159.04
## AIC=2334.08   AICc=2334.37   BIC=2368.05
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.05395142 2.177221 1.420674 -2.634651 11.80047 0.2030171
##                      ACF1
## Training set -0.005893674
```

```r
# Residual Mean
m <- mean(Arima$residuals) # mean of residuals is 0.053
```

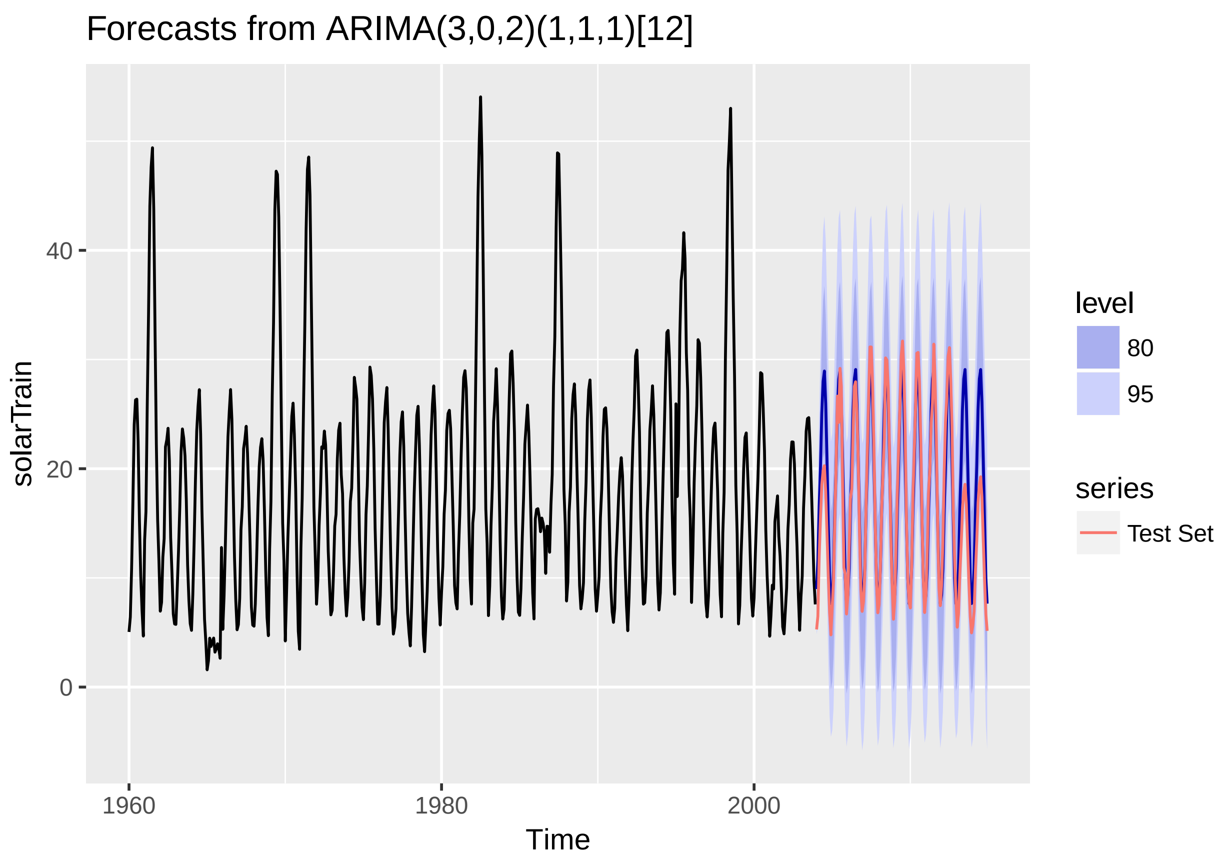
We deal with the non normality by using bootstrap prediction intervals and deal with the mean by adding the mean to the forecasts.

```r
# Model Prediction
ArimaFcast <- forecast(object = Arima,h = h,bootstrap = TRUE)

# Adding the mean to the forecasts
ArimaFcast$mean <- ArimaFcast$mean + m


# Accuracy on Test Data with mean addition
accuracy(f = ArimaFcast,x = solarTest) # MASE 0.3740
```

```
##                       ME     RMSE      MAE        MPE     MAPE      MASE
## Training set  0.05395142 2.177221 1.420674  -2.634651 11.80047 0.2030171
## Test set     -1.35894629 3.710696 2.617328 -13.888127 19.71847 0.3740212
##                      ACF1 Theil's U
## Training set -0.005893674        NA
## Test set      0.896156622  1.019035
```

The plot of the forecasts is as follows:

```r
# Plot of Forecasts
autoplot(ArimaFcast,series = "Forecast") +
    autolayer(object = solarTest,series = "Test Set")
```



With mean addition the MASE slightly increases to 0.374 from 0.3732. We still go ahead with this model since the change is negligible and residuals are good. From the models explored the best model was ARIMA(3,0,2)(1,1,1)[12].