**TIME SERIES FORECASTING**

**Forecast Model Building for a real-world time series.**

Series 22 (D449)

1. **Executive Summary**

The report discusses the model building for the time series 22 i.e., D449. In the previous project the data exploration at the descriptive stage was done. The data was analysed at different aggregate levels i.e., daily, quarterly, weekly, monthly. Now in this report we are discussing various model developments to forecast 14 days ahead. Since we need to forecast 14days ahead we are dealing with daily time series to do the model development.

As per the previous analysis the time series consists of no trend and seasonality. The presence of seasonality is obviously not there though we can consider a slight effect on trend which is different in first and second half. The initial time series comprises of outliers and level shifts as per the analysis done earlier. In order to reduce the influence of outliers and level shifts and to robustify the model evaluation “rolling origins” have been introduced. We choose a certain size for origin based on visualising our original time series. We will choose an origin size in such a way that the in-sample data has influence of outliers and level shifts so that the forecast is accurate. We will then set the horizon which will be the constant holdout sample size. Based on these sizes we will do the forecasts, calculate the preferred error measures, and choose the best performing train and test lengths for other model developments.

So now the entire time series is divided into two parts, train dataset and test dataset. Various models are fitted on the train data set and their accuracy is checked based on the accuracy measures (using AICc for ARIMA and ETS). The best model in each family i.e., ARIMA, ETS and Regression is chosen by comparing their various accuracy measures. The automatic functions are also used to produce one automatically built model in each of these model families. The iterative process to reach to a final most accurate manual model for each family is defined in detail in this report. All these models are then compared against the Naïve forecasting model. Out of all these models, one best model is chosen which is the most accurate. This model selection is done based on root mean squared error. The model which gives us the minimum RMSE is the best of all keeping in mind the other parameters discussed in detail in the report. This model is finally used to produce 14 days ahead forecast of our original daily time series.

In summary, all the steps which are performed in the iterative modelling process are covered and some good fitting models are produced and the best of all is used to forecast the time series as expected.

1. **Table of Contents**
2. Executive Summary2
3. Table of Contents………………………………………………………………………………………………………………….3
4. Introduction4
5. Exponential Smoothing methods4

**4.1** Rolling Origin4

**4.2** Naïve Method………………………………………………………………………………………………………………….4

**4.3** Simple Moving Average…………………………………………………………………………………………………..5

**4.4** Exponential Smoothing……………………………………………………………………………………………………6

**5.** Arima Modelling Methods8

**6.** Regression Modelling Methods…………………………………………………………………………………………..10

**7.** Model Comparison………………………………………………………………………………………………………………12

**8.** Conclusion…………………………………………………………………………………………………………………………..13

**9.** Appendices………………………………………………………………………………………………………………………….15

1. **Introduction**

This is the extension of the previous data exploration of series 22. The main objective is to find out the most accurate forecasting model and give 14 days ahead forecast of our time series.

The data exploration was done in the last project based on which we got to know that the time series does not exhibit any trend or seasonality. The forecasting models from different families such as ARIMA, ETS, Regression are built one by one. The best model from each family is selected based on various accuracy parameters. After the best model from each family is selected, they are then compared against each other to get the most accurate forecasting model. This comparison is also done against Naïve forecasting model. The accuracy is measured based on root mean squared error. Whichever model has the least RMSE is the best forecasting model. Based on this model the 14 days ahead forecast of our original time series is done.

1. **Exponential Smoothing Methods**
   1. **Rolling origin**

The daily time series has no obvious trend and seasonality but there exists outliers and level shift in it. To reduce the influence of outliers and level shifts in our estimation we are using rolling origin so that the in-sample data contains the impact of the outliers and level shifts in it and the forecasted result accounts the presence of these outliers and level shift. For this purpose, we are choosing number of rolling origins as 500 and horizon as 100. We will now set the test and train length based on these and form the train time series and test vector accordingly. We will define the holdout and forecast matrix for this. After defining the column names, we will produce 500 forecasts using ETS(A,N,N) model with the horizon of 100. After the holdout and forecast matrix are produced, we will now calculate the absolute error. Once we obtain the error vector we will now look where this error is minimum. We found that the error is minimum at origin number 299. So now 299 is our origin size and we will now calculate the test and train length based on this origin size. By using rolling origins, we have obtained our most preferred model train and test size. This helps us to divide the time series in the most accurate size of train and test.

So now using origin size as 299 and horizon as 100 our new train length is 3183 and test length is 398 which is approximately a 90:20 split of the actual time series.

* 1. **Naïve Method**

Naïve method is used for benchmarking. The forecasts produced using Naïve will help us to compare the accuracy and efficiency of various upcoming models. If the basic model is the best for us, then we don’t need to go ahead with complex models. We will divide the dataset into train and test based on the length calculated using rolling origins. The naïve method will be then applied to the training data set with horizon size equal to the test dataset length. The forecast will then be calculated, and the root mean squared error will be calculated using the test dataset. This root mean squared error will act as a benchmark value to compare with all other forecasting models. Here, the root mean squared error obtained is 2103.597 and mean absolute error is 1732.624.

Chart

Description automatically generated

Fig 4.1 Forecast produced by Naïve method on train dataset

* We will be using root mean square error (RMSE) to do the model comparison within the family. This RMSE help us to capture the effect of outliers since our data has outliers. Also, RMSE is useful in obtaining that how far is our forecast from the test dataset. RMSE helps us to capture the effect of even the most insignificant error value as squaring a value will increase its overall value and the model accuracy can be impacted based on this.
  1. **Simple Moving Average**

SMA is used to smooth the data, since there is no significant trend in the time series, SMA is not very effective in capturing anything. Due to presence of level shift SMA is not producing effective results and when SMA of train data is calculated and compared to the test data it is not even close to the level of test data. Due to this discrepancy in the output the SMA model is not fitting well to our train data. We have tried various moving average lengths to see the results, but SMA is not able to perform well in capturing the level shift. For reference we are using ma=500 and on plotting the SMA on train data we get the following graph:

This is of no use since the red line is far below the values in test data and hence the forecast will not be accurate. Even after increasing the order such as ma=1000, ma=2000 we aren’t getting any expected results.

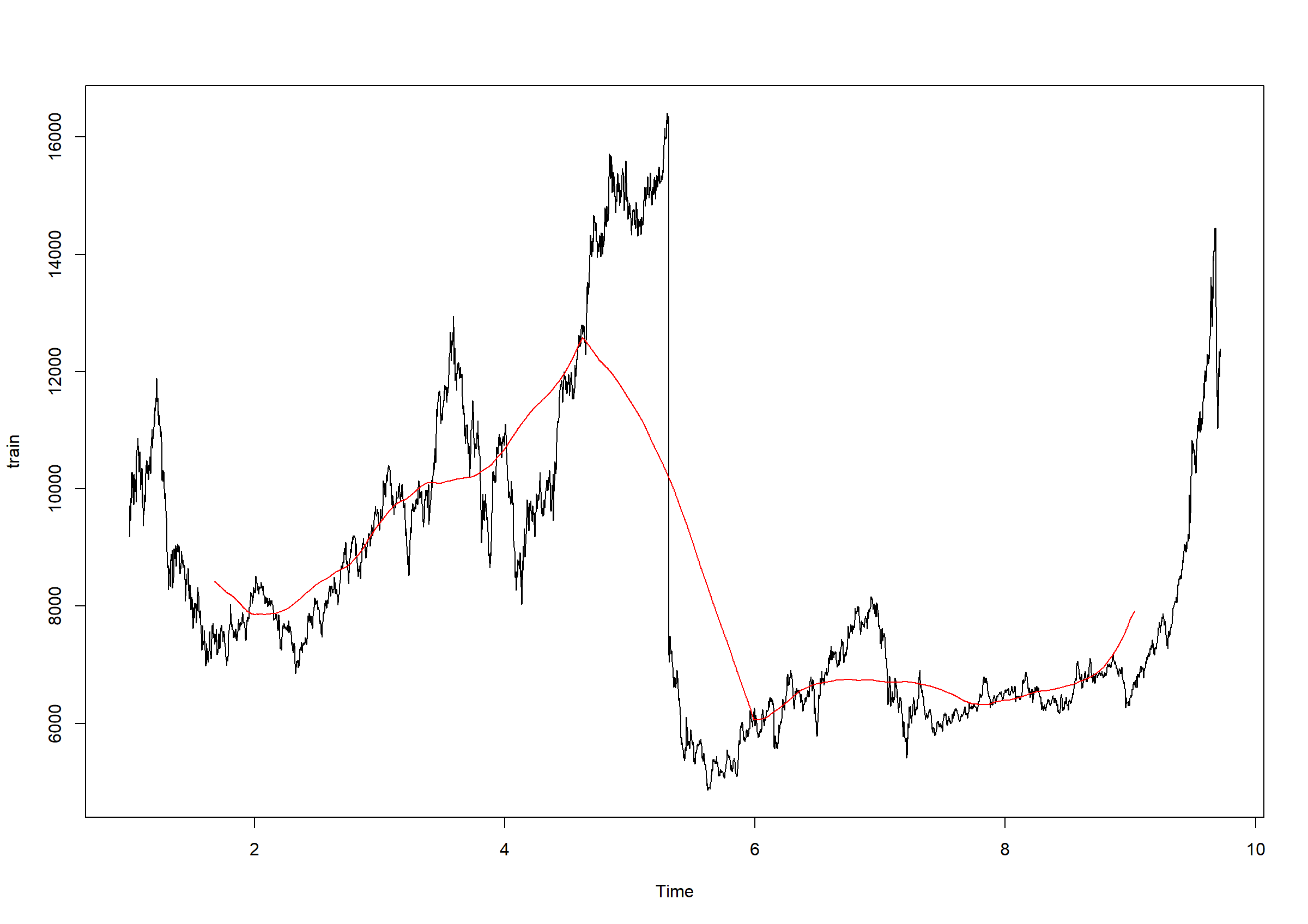


Fig 4.2 Simple moving average on train set

* 1. **Exponential Smoothing**

We will now fit various ETS models in our data. The time series is divided into test and train dataset based on the length obtained from rolling origin method. Once the data is divided into train and test the following graph is generated for our original time series:

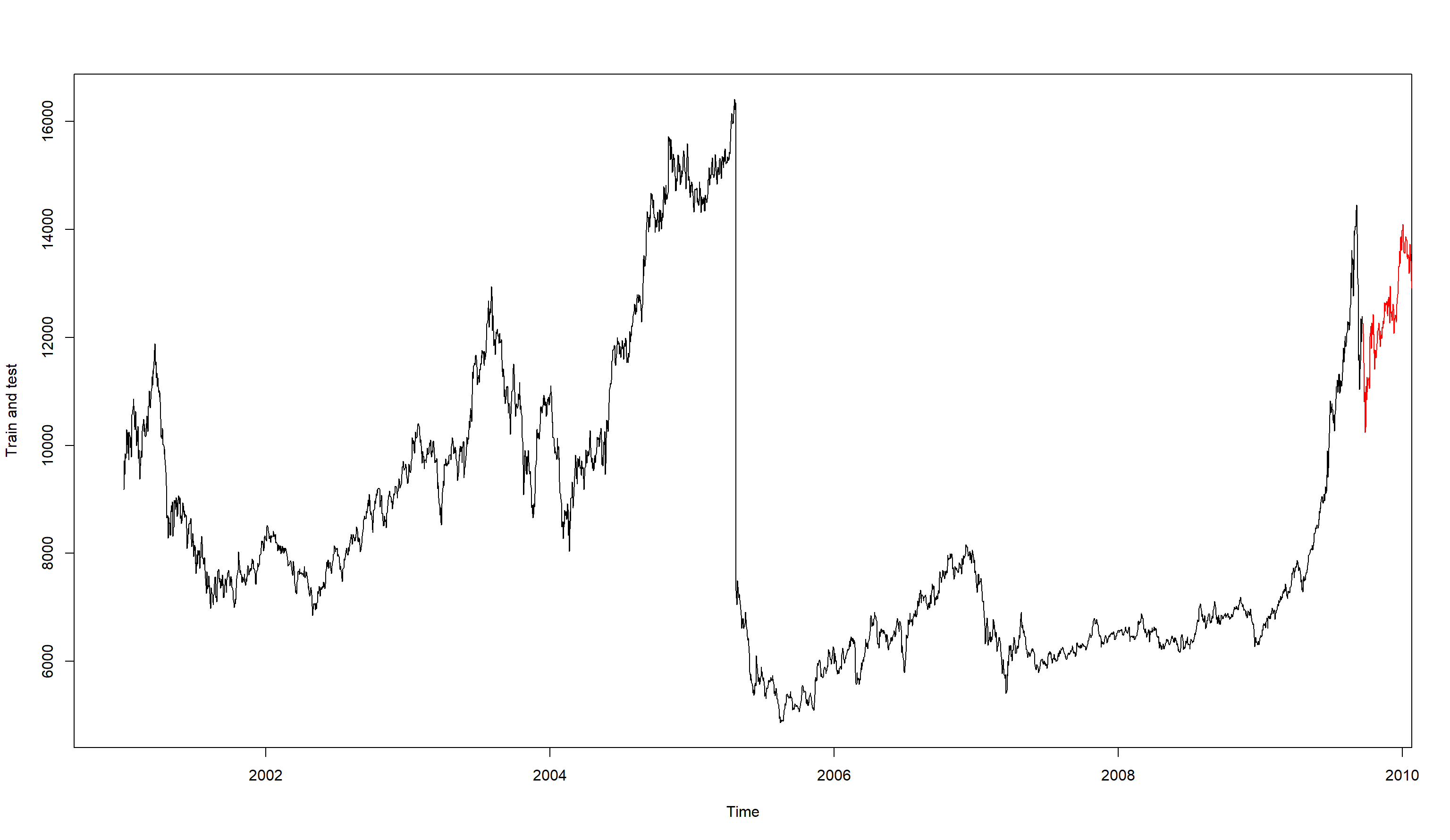


Fig 4.3 Plot of train and test set after bifurcation

Now, we will try fitting various ETS models on our train dataset and compare the forecasted values with the test dataset. In the previous assignment analysis, on decomposition of the series we found that there was not very significant difference in the error of multiplicative and additive decomposition so here we are considering both the models i.e., with additive error and multiplicative error.

Since, the trend is not very significantly present in the series we are still trying to fit the ETS model with additive damped trend because the trend pattern differs in first and second half of the series, this helps to see whether the trend parameter might be useful in forecasting values. Since there exists no seasonality, we will not be touching the seasonal parameter at all, and it will remain none at all times. Then later we are fixing the seed to see the effect on the forecast.

Once we manually attain an ETS model we will try to get the model which is automatically built by the system. To find out the best ETS model we are comparing various accuracy parameters of each model and comparing the forecasted values with the test dataset. We are trying to find the minimum error when comparing the forecasted values with the test dataset. Below table shows the values of accuracy parameter for each ETS model:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **AIC** | **AICc** | **BIC** | **RMSE(train)** | **RMSE(test)** |
| ETS(ANN) | 59568.72 | 59568.72 | 59586.91 | 205.119 | 2103.607 |
| ETS(MNN) | 57116.75 | 57116.76 | 57134.95 | 205.119 | 2103.606 |
| ETS(AAdN) | 59574.92 | 59574.95 | 59611.32 | 205.1257 | 2103.668 |
| ETS(MAdN) | 57121.05 | 57121.07 | 57157.44 | 205.1421 | 2105.395 |
| ETS(ANN) fixed seed | 42924.93 | 42924.93 | 42930.99 | 1.000000e+00 | 2103.597 |
| ETS(MNN) fixed seed | **40472.94** | **40472.95** | **40479.01** | **1.000000e+00** | **2103.597** |

Table 4.1 Model comparison for ETS

Based on the tabular data presented above we will check the AIC, AICc, BIC and RMSE for both train and test. Whichever model shows the least for all of it will be our most suitable ETS model. As we can see in the table that ETS(MNN) with fixed seed shows the minimum value for all of these. Hence, the best model in the ETS family would be ETS(MNN) with fixed seed.

We can now see the plot of forecast on train data set by applying ETS(MNN) with fixed seed.

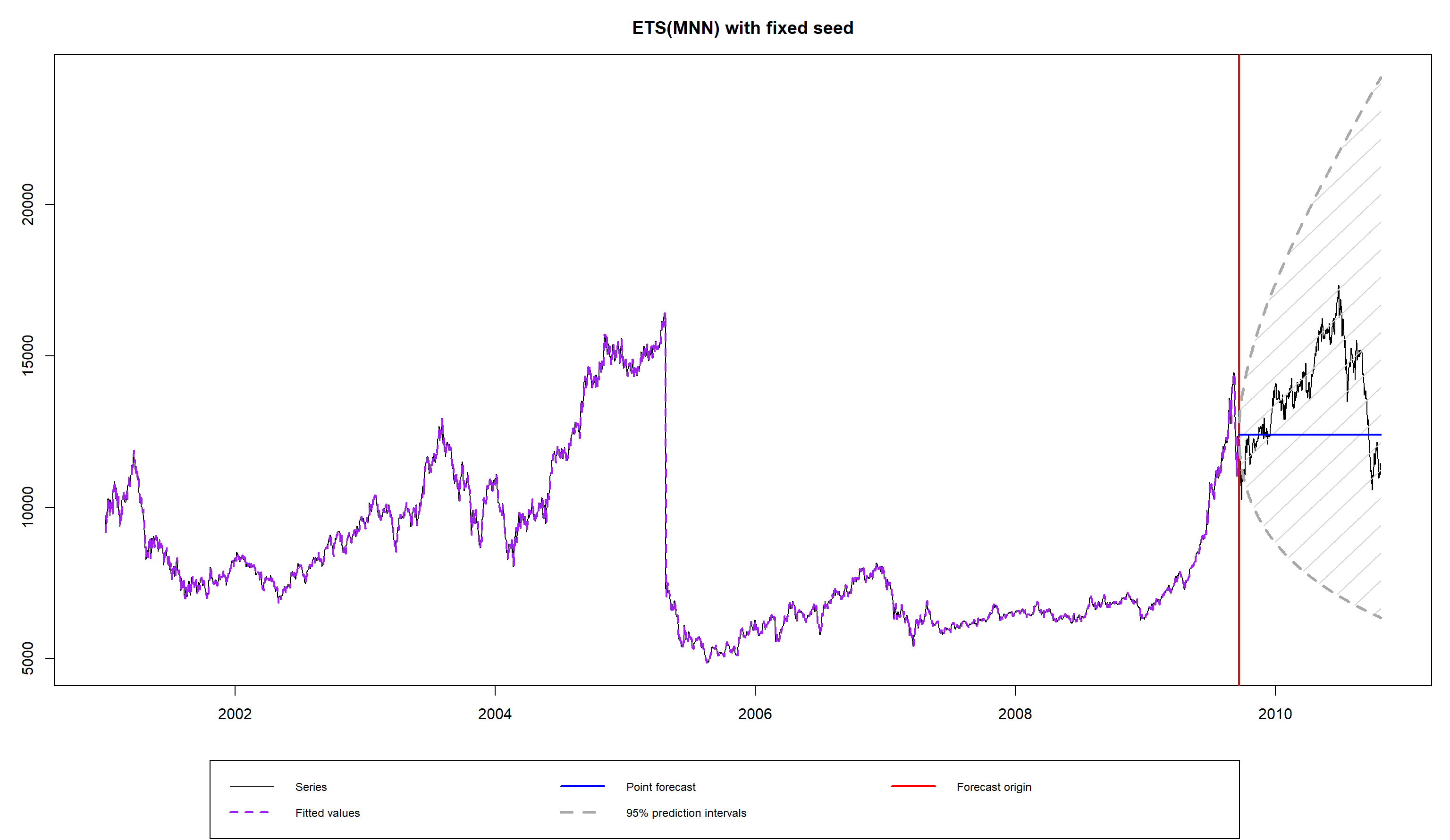


Fig 4.4 Forecast produced by ETS(MNN) with fixed seed on train dataset

Now we will see what ETS model is automatically built by the system when we do not pass any specific model type. Since the data has no seasonality, we can pass seasonality parameter as none so that the system can quickly find the optimised model.

On using model = ZZN we get the optimised ETS model as ETS(M,N,N) which is also covered in our manual model development approach.

Comparison with Naïve model

When the best ETS model is compared with Naïve model we will compare their root mean squared error for test data. On comparison of these parameters, we found that the values of RMSE are equal for both Naïve and ETS i.e., 2103.597. Hence, we can say that both models are equivalent in terms of accuracy.

1. **ARIMA Modelling Methods**

The time series should be stationary to find a good forecasting model using ARIMA. Based on KPSS and ADF tests we have found that the time series is not stationary. We can also use these statistical tests on train data to check for stationarity. On applying these tests on train dataset, we found that the data set is not stationarity, so we took the first difference of our train dataset. We again performed KPSS and ADF tests on the differenced data set. Now we found that the data is stationary. We will now proceed with Box-Jenkins method before using the auto ARIMA function. It is now time to check the peaks in the ACF and PACF to determine the order of AR & MA for our model.

On plotting the ACF and PACF of the differenced data we get the following result:

Chart

Description automatically generated

Fig 5.1 tsdisplay of differenced training data

From this picture you can see that there are peaks in both ACF and PACF. The single peaks in both ACF and PACF at the start are very significant so this helps us to determine the order of AR and MA. Since there is a single peak in PACF at the beginning, the order of AR would be taken as 1 and since there is a single peak in ACF too at the beginning, the order of MA would be taken as 1. The data is differenced once so we will try fitting ARIMA(1,1,1) to our train dataset. We will also try fitting ARIMA(0,1,1) by ignoring the peak of AR for a while and just considering the peak of MA. Again we will try fitting ARIMA(1,1,0) by ignoring the peak of MA for a while and just considering the peak of AR. We are also trying to fit ARIMA(0,1,0) random walk model, by ignoring the peaks of both AR and MA and just considering the differenced data. When we plot the residuals of each of these fits there isn’t anything which can be concluded visually so we decide the best model by looking at their accuracy parameters. We will also compare the forecasts performed by each model and the test data set. The model producing minimum error will be contested as the best model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **AIC** | **AICc** | **BIC** | **RMSE(test)** |
| ARIMA(1,1,1) | 42915.78 | 42915.79 | 42933.98 | 2102.838 |
| ARIMA(0,1,1) | 42913.8 | 42913.8 | 42925.93 | 2102.337 |
| ARIMA(1,1,0) | **42913.8** | **42913.8** | **42925.93** | **2102.274** |
| ARIMA(0,1,0) | 42912.44 | 42912.44 | 42918.51 | 2103.597 |

Table 5.1 Model Comparison for ARIMA

Based on the above tabular data we can see that the accuracy parameters are least for ARIMA(0,1,0) but the RMSE on test data is maximum in this case whereas in ARIMA(1,1,0) and ARIMA(0,1,1) we can see that the accuracy parameters are equal and not the least when compared amongst the four. But also, the RMSE is lesser for ARIMA(1,1,0) when compared to RMSE for ARIMA(0,1,1). So in that case we found our best model as ARIMA(1,1,0).

The residuals for the ARIMA(1,1,0) fit are plotted below but nothing can be concluded by visualising the residuals of any of the fits.

Chart

Description automatically generated

Fig 5.2 residuals for best fit ARIMA model

The forecasts produced by this fit i.e., ARIMA (1,1,0) are given below:

Chart

Description automatically generated

Fig 5.3 Forecasts produced by best fit ARIMA model on train set

Comparison with Naïve model

When the best ARIMA model is compared with Naïve model we will compare their root mean squared error for test data. On comparison of these parameters, we found that the values of RMSE for the best ARIMA model i.e., ARIMA(1,1,0) is lesser than the RMSE of Naïve. Hence, we can say that here ARIMA model is better performing than the Naïve model.

1. **Regression Modelling Methods**

This dataset doesn’t have explanatory variables, so we don’t have any opportunity to predict a dependent variable using explanatory variables. The time series doesn’t exhibit any seasonality so we can’t add any seasonal dummies also. Since there is some trend exhibited by the time series, we are adding time trend column to our original time series.

Now the original dataset is no more a time series, we will now divide this dataset into train and test datasets.

We now have two variables in our train and test datasets i.e., original data and trend.

We are now trying to fit various regression models on the train dataset and find the best regression model. We will then use this model to see its accuracy on test data.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Residual std. error** | **Adjusted R squ.** | **RMSE (test)** |
| All variables inclu. | 2459 | 0.09133 | 6989.351 |
| Intercept removed | 5504 | 0.6158 | 1751.408 |
| No variables | 2580 | - | 5509.303 |

Table 6.1 Model comparison for linear regression

Based on the above tabular data we can see that the regression model with maximum adjusted R squared is the one where intercept is removed but still the fit is approx. 62% which may not be considered a good fit. Also, if we see the residual standard error in this case its is the maximum of all models.

Now we generate a model which is automatically built by the system. In this case, we get the model which includes all variables and as per the adjusted R squared value this model is a poor model. Going ahead with the best of three model which is the one where intercept is removed, we are now calculating the residuals and plotting the histogram and QQ plot of the residuals to check whether the residuals are normally distributed or not.

Chart, histogram

Description automatically generated

Fig 6.1 Histogram of residuals of best fit regression model

Diagram

Description automatically generated

Fig 6.2 QQ plot of residuals of best fit regression model

Based on these plots, the residuals do not follow normal distribution, hence this model is not a good fit since it is failing the assumptions to be covered. In this case we can conclude that in any case regression model is not going to be a good fit. We are not going ahead with regression and will proceed with ARIMA and ETS models for further calculation.

1. **Model Comparison**

After finding the best fit model in each family we are now finding the best model which will be used for final forecasts. We dropped the regression model earlier based on assumptions followed by residuals. The residuals of the linear regression model were violating the assumptions of normal distribution hence, linear regression is not a good model. We are now left with two family of forecasting models i.e., ARIMA and ETS.

We will now try finding out the best forecasting model out of these two. To do so we are considering two error metrics which are Mean Absolute Error and Root Mean Square Error for comparison.

We are going with Mean Absolute Error as it will help us get the absolute value of all the errors calculated. The error calculated can be positive or negative so if we are considering errors just like that the positive and negative values tend to nullify each other to some extent, this can impact the overall mean value of the error and give us wrong indication about the accuracy of model.

Similarly, considering root mean square error help us to increase the significance of the smaller error. When small error is present, squaring it would increase its value, thus helping us to determine its relevance in our model. Therefore, all the error are firstly squared, then their mean is obtained and then their square root is calculated so that the final value obtained has the uniform unit throughout.

MAE and RMSE help us to capture all the effects of the error on the accuracy of our model. These values help us get the correct information about our model accuracy. The presence of outliers can affect the overall accuracy of the model so using RMSE and MAE we are also trying to incorporate the effect of outliers on our model. All the errors calculated will have the significance on overall model, hence, MAE and RMSE are the best measure to compare accuracy of the models.

The tabular data below gives all the details of the errors:

|  |  |  |
| --- | --- | --- |
| Model | Mean Absolute Error | Root Mean Square Error |
| ARIMA(1,1,0) | 1731.598 | 2102.274 |
| ETS(M,N,N) fixed seed | 1732.624 | 2103.597 |
| Naive | 1732.624 | 2103.597 |

Table 7.1 Final model comparison

Based on the above tabular data we can clearly conclude that ARIMA(1,1,0) is our best fit model out of all since it has lowest MAE and RMSE. We will now apply this model to our original time series and fit it into that to do 14 days ahead forecast.

On applying ARIMA(1,1,0) to our original time series we get the following coefficients:

Text

Description automatically generated

The forecast function is used to generate the forecast for horizon of size 14.

Text, letter

Description automatically generated with medium confidence

The plot of our daily time series with the 14 days ahead forecasts is given below:

Chart

Description automatically generated

Fig 7.1 Plot for daily time series with 14 days ahead forecast

1. **Conclusion**

The main objective of this assignment is to find out the best forecasting model and produce forecasted values for 14-days ahead. Firstly, we chose an origin size of 500 and horizon size of 100 by visualising our data and ensuring that the effect of outliers and level shifts is captured in our train data set. Based on these values we obtain train and test length and we then split our data into train and test sets. Forecasting is done based on ETS model and this forecasting is performed from 500 consecutive origins. The forecasted values are then compared with the holdout values and a point where minimum error is obtained is found. We then make use of this point to get more accurate train and test length. So, using rolling origins we got the most accurate train and test length for our further mode development. Rolling origin help us to make our model more robust to changes and the effect of outliers and level shift is also captured to help us do better forecasts. We have then divided the daily time series into test and train sets based on the lengths obtained using rolling origin. The Naïve method is applied, and forecast is produced. This Naïve method is used for benchmarking.

Next, we tried calculating Simple Moving Average with various orders but since the data has no trend but obvious level shifts, SMA failed to capture its effect in forecast. The forecast produced by SMA is far away from the test dataset. So, we can conclude that SMA is not a good model to be considered for this time series.

Then multiple ETS models are fitted to the train dataset and the forecasts for our test horizon are calculated. Each of these forecasts are compared to the test data and errors are calculated. We are using RMSE to compare the accuracy of models. For ETS since the decomposition of time series showed that there is hardly any difference between additive and multiplicative decomposition, so we started with using ETS(ANN) and ETS(MNN). The time series exhibits little trend, which is different in first and second half, so ETS(AAdN) and ETS(MAdN) is also applied to out train set. We have then fixed the seed value and tried obtaining the best fit. So ETS(ANN) and ETS(MNN) are again applied but with fixed seed. Finally, we compare models based on their accuracy parameters and RMSE in comparison to test data. We can see that there is not very significant difference in the accuracy parameters and RMSE value for model with additive error, no trend and no seasonality and model with additive error, damped additive trend and no seasonality. Similarly, we can see that there is not very significant difference in the accuracy parameters and RMSE value for model with multiplicative error, no trend and no seasonality and model with multiplicative error, damped additive trend and no seasonality. But when we see these values for ETS(ANN) and ETS(MNN) with fixed seed we can see a significant difference. So, we obtain ETS(MNN) with fixed seed as out best model though the model built automatically by the system is ETS(MNN).

In the next step the best model from ARIMA family is obtained. Since, the time series is not stationary, it is differenced one time and statistical tests are performed to check its stationarity. Once the time series is stationary, ACF and PACF is analysed to derive order for ARIMA. Based on ACF and PACF analysis various ARIMA orders such as (1,1,1), (1,1,0), (0,1,1) and (0,1,0) are applied. The various ARIMA models are fitted to the train data sets and forecasts are produced. These forecasted values are compared to the test data set and various accuracy parameters and RMSE is used to find the best fit model. We found that ARIMA(1,1,0) and ARIMA(0,1,1) have same values for AIC, AICc, BIC but ARIMA(1,1,0) has lesser RMSE for test compared to ARIMA(0,1,1) so overall we found ARIMA(1,1,0) to be the best fitting model in the ARIMA family.

We then move to regression where we try to obtain various regression models. Due to lack of independent variables, we are confined with only single variable which is a dependent variable. The data doesn’t exhibit any seasonality so we can’t use dummy variables for comparison. So, a time trend is added to the series and the newly formed series is divided into train and test set. Then various regression models are applied such as one which includes all variables, one where intercept is removed and one where no variables are considered. Out of all these models we found one best fit model which has the maximum adjusted R-squared value, but residual standard error was also maximum which raised doubts regarding the model. On further analysis of residuals of this fit, we found that the residuals do not follow the normal distribution assumption which is a necessary thing to follow for the residuals of a correct linear regression model. Based on these observations we can conclude that linear regression is not a good model that can be used for forecasting this time series. So, we do not go ahead with regression.

Now the best models from ARIMA and ETS are compared based on two error metrics that are mean absolute error and root mean square error. All these models are compared with Naïve too as Naïve is our benchmark model. On this comparison we found that Naïve and our best ETS model are equivalent and produce same forecasts. Out of all these models ARIMA (1,1,0) comes out to be the most accurate model.

The 14 days ahead forecasts of our time series are produced by using ARIMA (1,1,0).

The forecasted values are given below:

11109.72 11109.72 11109.72 11109.72 11109.72 11109.72 11109.72 11109.72 11109.72 11109.72 11109.72 11109.72 11109.72 11109.72

1. **Appendices**

**R Script**

mydata <- read.csv("D449.csv")

#mydata$Day <- as.Date(mydata$Day, format="%d/%m/%Y")

View(mydata)

daily\_ts <- ts(mydata$Micro, frequency = 365, start = c(2001,1,3))

plot(daily\_ts)

#rolling origins definition

# Set horizon and number of rolling origins

h <- 100

origins <- 299

daily\_ts\_length <- length(daily\_ts)

#setting test and train length

train\_length <- daily\_ts\_length - h - origins + 1

test\_length <- h + origins - 1

daily\_ts\_train <- ts(daily\_ts[1:train\_length], frequency=frequency(daily\_ts), start=start(daily\_ts))

daily\_ts\_test <- daily\_ts[(train\_length+1):daily\_ts\_length]

#Matrix for holdout and forecasts

daily\_ts\_forecasts <- matrix(NA, nrow=origins, ncol=h)

daily\_ts\_holdout <- matrix(NA, nrow=origins, ncol=h)

colnames(daily\_ts\_forecasts) <- paste0("horizon",c(1:h))

rownames(daily\_ts\_forecasts) <- paste0("origin",c(1:origins))

dimnames(daily\_ts\_holdout) <- dimnames(daily\_ts\_forecasts)

for(i in 1:origins){

# Create a ts object out of the daily\_ts data

our\_train\_set <- ts(daily\_ts[1:(train\_length+i-1)],

frequency=frequency(daily\_ts),

start=start(daily\_ts))

# Write down the holdout values from the test set

daily\_ts\_holdout[i,] <- daily\_ts\_test[i-1+(1:h)]

# Produce forecasts and write them down

daily\_ts\_forecasts[i,] <- forecast(ets(our\_train\_set,"ANN"),h=h)$mean

}

View(daily\_ts\_forecasts)

View(daily\_ts\_holdout)

#Matrix for error

error <- rowMeans(abs(daily\_ts\_holdout - daily\_ts\_forecasts))

error

min(error)

#splitting the series

# Find the total number of observations

daily\_ts\_length <- length(daily\_ts)

# Write down size of training set

train\_length <- 3183

# And the forecasting horizon

h <- 398

# Create the training set

train <- ts(daily\_ts[1:train\_length], frequency=365, start = c(2001,1,3))

plot(train)

# Create the test set

test <- ts(daily\_ts[(train\_length+1):daily\_ts\_length],frequency=365, start = c(2009,265))

#Naive method

naive\_method <- naive(train, h=h)

naive\_forecast <- naive\_method$mean

Naive\_RMSE <- sqrt(mean((test-naive\_forecast)^2))

Naive\_RMSE

Naive\_MAE <- mean(abs(test-naive\_forecast))

Naive\_MAE

tsdisplay(daily\_ts)

#test stationarity

kpss.test(train)

# you can reject the Null Hypothesis of the data being stationary.

#So KPSS test tells us that we have enough evidence to say that data is not stationary.

adf.test(train)

#The p-value displayed is approximately equal to 0.5416

#which means that we fail to reject the Null Hypothesis of non-stationarity at 5%.

#Calculating first difference since ts is not stationary.

diff\_train <- diff(train)

kpss.test(diff\_train)

adf.test(diff\_train)

#the test shows that data is stationary and now we can check for ACF and PACF

tsdisplay(diff\_train)

#Since one peak in PACF so we will start with ARI(1,1) i.e. Arima(1,1,0)

first\_fit <- Arima(train, order=c(1,1,1))

# Plot series, ACF and PACF of the residuals

tsdisplay(residuals(first\_fit))

first\_fit

#since ACF & PACF both don't have anymore peaks, this looks like a best fit so we will proceed with this.

#We have found the best model and we can now forecast using this.

auto.arima(train) #ARIMA(1,1,0)

#using auto.arima also we get the same order for our best fit.

#Find best method via AIC

auto.arima(train, ic="aic") #ARIMA(1,1,0)

#Find best method with ADF Test

auto.arima(train, test="adf") #ARIMA(1,1,0)

second\_fit <- Arima(train, order=c(0,1,1))

tsdisplay(residuals(second\_fit))

second\_fit

third\_fit <- Arima(train, order=c(1,1,0))

third\_fit

tsdisplay(residuals(third\_fit))

plot(forecast(third\_fit, h=398))

fourth\_fit <- Arima(train, order = c(0,1,0))

fourth\_fit

arima\_fit\_forecast <- forecast(third\_fit, h=398)$mean

plot(forecast(first\_fit, h=398))

acf(third\_fit)

#Calculate errors

errors <- test - arima\_fit\_forecast

errors\_ME <- mean(errors)

errors\_MSE <- mean(errors ^ 2)

errors\_RMSE <- sqrt(errors\_MSE)

errors\_MAPE <- 100 \* mean(abs(errors)/test)

errors\_RMSE

errors\_MAPE

MAE <- mean(abs(errors))

MAE

#fitting simple moving average

SMA <- ma(train, order=1000, centre=FALSE)

plot(train)

lines(SMA, col = "red")

# Firstly we get rid of NA (‘‘Not Assigned’’) values:

SMA\_no\_NAs <- SMA[!is.na(SMA)]

# Then form a forecast:

SMA3\_forecast <- ts(rep(SMA\_no\_NAs[length(SMA\_no\_NAs)],398), frequency=365)

plot(test)

lines(SMA3\_forecast, col = "red")

#fitting simple moving average

SMA <- ma(daily\_ts, order=3, centre=FALSE)

# Firstly we get rid of NA (‘‘Not Assigned’’) values:

SMA\_no\_NAs <- SMA[!is.na(SMA)]

# Then form a forecast:

SMA3\_forecast <- ts(rep(SMA\_no\_NAs[length(SMA\_no\_NAs)],14), frequency=365, start = end(daily\_ts))

plot(daily\_ts)

lines(SMA3\_forecast, col = "red")

#Calculate errors

SMA3\_errors <- test - SMA3\_forecast

SMA3\_ME <- mean(SMA3\_errors)

SMA3\_MSE <- mean(SMA3\_errors ^ 2)

SMA3\_MAE <- mean(abs(SMA3\_errors))

SMA3\_MAPE <- 100 \* mean(abs(SMA3\_errors)/test)

#use MAE and RMSE for error comparison

#RMSE

#Naive method for forecast

naive\_method <- naive(train, h=h)

naive\_forecast <- naive\_method$mean

plot(test)

plot(naive\_forecast, col = "red")

#Exponential smoothing method

ETS\_ANN <- ets(train, "ANN")

ETS\_ANN

summary(ETS\_ANN)

ETS\_ANN\_forecast <- forecast(ETS\_ANN, h=398)$mean

plot(forecast(ETS\_ANN, h=398))

ETS\_MNN <- ets(train, "MNN")

ETS\_MNN

summary(ETS\_MNN)

ETS\_MNN\_forecast <- forecast(ETS\_MNN, h=398)$mean

plot(forecast(ETS\_MNN, h=398))

ETS\_AAdN <- ets(train, model="AAN", damped=TRUE)

ETS\_AAdN

summary(ETS\_AAdN)

ETS\_AAdN\_forecast <- forecast(ETS\_AAdN, h=398)$mean

ETS\_MAdN <- ets(train, model = "MAN", damped = TRUE)

ETS\_MAdN

summary(ETS\_MAdN)

ETS\_MAdN\_forecast <- forecast(ETS\_MAdN, h=398)$mean

es\_ANN\_initial <- es(daily\_ts, model="ANN", initial=train[1], h=h, holdout=TRUE)

es\_ANN\_initial$accuracy

summary(es\_ANN\_initial)

es\_ANN\_initial\_forecast <- forecast(es\_ANN\_initial, h=h)$mean

es\_MNN\_initial <- es(daily\_ts, model="MNN", initial=train[1], h=398, holdout=TRUE)

es\_MNN\_initial$accuracy

summary(es\_MNN\_initial)

es\_MNN\_initial\_forecast <- forecast(es\_MNN\_initial, h=398)$mean

plot(forecast(es\_MNN\_initial, h=398), main="ETS(MNN) with fixed seed")

#Calculate errors

errors <- test - es\_MNN\_initial\_forecast

errors\_ME <- mean(errors)

errors\_MSE <- mean(errors ^ 2)

errors\_RMSE <- sqrt(errors\_MSE)

errors\_MAPE <- 100 \* mean(abs(errors)/test)

errors\_RMSE

errors\_MAPE

MAE <- mean(abs(errors))

MAE

# Fit SES with fixed initial seed

es\_ANN\_initial\_1 <- es(daily\_ts, model="ANN", initial=daily\_ts[1], h=h, holdout=TRUE)

es\_ANN\_initial\_1$accuracy

es\_ANN\_initial\_1\_forecast <- forecast(es\_ANN\_initial\_1, h=h)$mean

plot(forecast(es\_ANN\_initial\_1, h=h))

# Fit SES with optimised Seed

es\_ANN\_opt <- es(daily\_ts, model="ANN", h=h, holdout=TRUE)

es\_ANN\_opt$accuracy

es\_ANN\_opt\_forecast <- forecast(es\_ANN\_opt, h=h)$mean

plot(forecast(es\_ANN\_opt, h=h))

# Fit SES with optimised Seed (Benchmarking)

daily\_ts\_naive <- es(daily\_ts, model="ANN", persistence=1, h=h, holdout=TRUE)

daily\_ts\_naive\_forecast <- forecast(daily\_ts\_naive, h=h)$mean

plot(forecast(daily\_ts\_naive, h=h))

# Calculate an Optimized ETS Method using ets()

ets\_ZZZ <- ets(train, model="ZZN")

ets\_ZZZ

# Do the same using es()

es\_ZZZ <- es(daily\_ts, model="ZZZ")

es\_ZZZ

plot(train, ylab= "Train and test")

lines(test, col = "red")

#Regression Models

lag\_train <- Lag(train, -1)

D447\_reg <- cbind(train, lag\_train)

colnames(D447\_reg)<- c("original", "L1")

fit1 <- lm(original~., data =D447\_reg )

summary(fit1)

fit2 <- lm(original~.-1, data =D447\_reg )

summary(fit2)

tsdisplay(fit1$residuals)

fit3 <- lm(log(original)~log(L1), data =D447\_reg )

summary(fit3)

#Regression Models

lag\_train <- Lag(train, -1)

trend <- c(1:daily\_ts\_length)

D447\_reg <- cbind(daily\_ts, trend)

colnames(D447\_reg)<- c("original", "trend")

reg\_train<- window(D447\_reg, start= start(D447\_reg), end= c(2009,263))

reg\_test<- window(D447\_reg, start= c(2009,264), end= end(D447\_reg))

View(reg\_test)

fit1 <- lm(original~., data =reg\_train )

summary(fit1)

fit2 <- lm(original~.-1, data =reg\_train )

summary(fit2)

tsdisplay(fit1$residuals)

fit3 <- lm(original~.-trend, data =reg\_train )

summary(fit3)

fit4 <-lm(original~1, data =reg\_train)

summary(fit4)

forecast<- predict(fit4, reg\_test)

plot(predict(fit2, reg\_test))

RMS\_error = sqrt(mean((reg\_test[,1]- forecast)^2))

RMS\_error

auto\_model <- step(fit4, formula(fit1), direction="forward")

auto\_model1 <- step(fit1, direction="both")

auto\_model

auto\_model1

#Extract Residuals

fit2\_resid <- residuals(fit2)

#We will also need fitted values for our analysis, which can be extracted using fitted(): #Extract Residuals

fit2\_fitted <- fitted(fit2)

fit2\_resid

#Plot Histogram

hist(fit2\_resid)

#QQ-Plot

qqnorm(fit2\_resid)

qqline(fit2\_resid)

#Calculating final forecast using ARIMA(1,1,0)

final\_fit <- Arima(daily\_ts, order=c(1,1,0))

final\_fit

final\_forecast <- forecast(final\_fit, h=14)$mean

final\_forecast

plot(forecast(final\_fit, h=14))