Summary: **Forecasting on Wal-Mart Store Data Set**

1. **Forecasting goal:** The goal is to provide a robust forecast model that Wal-Mart can use to manage its store wise inventory efficiently and avoid stock outs or excess inventory counts. Our goal here is predictive in nature and forecasting five months in future so that Wal-Mart can take decisions well and analyze well within time.
2. **Forecasts is used by:** This forecast can be used by multiple stores / departments of Wal-Mart to maintain inventory levels as well as know the upcoming demand trend within the market. The accuracy of prediction is critical in this case as under prediction in inventory forecasts would result in loss in revenue whereas any over prediction will result in inventory holding costs.
3. **Forecast horizon:** Forecast Horizon would be for a period of five months i.e. 23 weeks in future. Given the fact that we have weekly data for two complete years for each store of Wal-Mart, we are currently working out with one store of Wal-Mart, which can very easily be increased / extended to other stores, owing to the robustness of model.
4. **Needed data granularity:** Data Granularity shall be at department level, however we can easily extend the same model to same departments of other Wal-Mart stores owing to the fact that same dependent parameters are present in other stores as well, and the trend is mostly dependent on these dependent variables for forecasting.

* We have a dataset from period 2010-02-05 up to 2012-10-26.
* We have started off from the simplest of modelling techniques and then proceeded towards more complex techniques in search of better models fitting the data.
* The following modelling techniques have been used:
  + Naïve Model
  + Random walk (with and without drift)
  + Average Method
  + Seasonal Naïve Model
  + Simple Exponential Smoothing(Holts Method)
  + Holts Winter (with Seasonal Component)
  + Arima
  + RandomForest
* Let us focus on Store 1 of the data set.

> dataset <- read.csv("train.csv")

> dataset\_store1 = subset(dataset, Store == 1)

> head(dataset\_store1, 3)

Store Date Weekly\_Sales IsHoliday

1 2010-02-05 24924.50 FALSE

1 2010-02-12 46039.49 TRUE

1 2010-02-19 41595.55 FALSE

> tail(dataset\_store1, 3)

Store Date Weekly\_Sales IsHoliday

1 2012-10-12 1061.02 FALSE

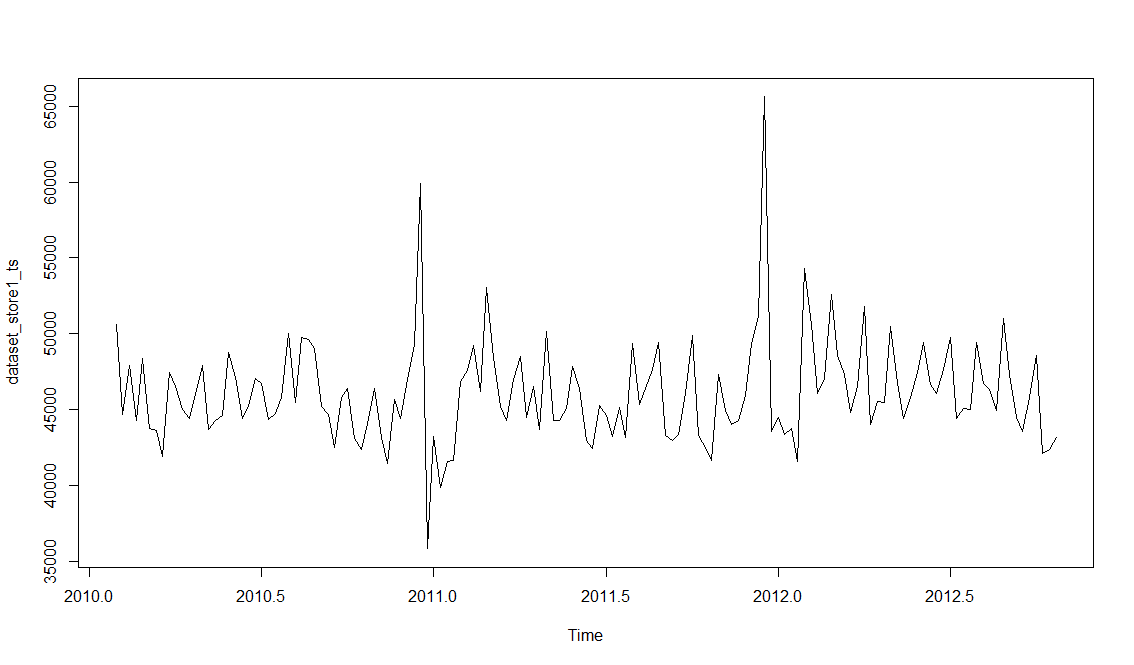
1 2012-10-19 760.01 FALSE

1 **2012-10-26** 1076.80 FALSE

* Plotting the Time Series
* Once we have read a time series into R, the next step is usually to make a plot of the time series data, which we can do by converting the dataset into timeseries object in R and plotting the same using plot() function

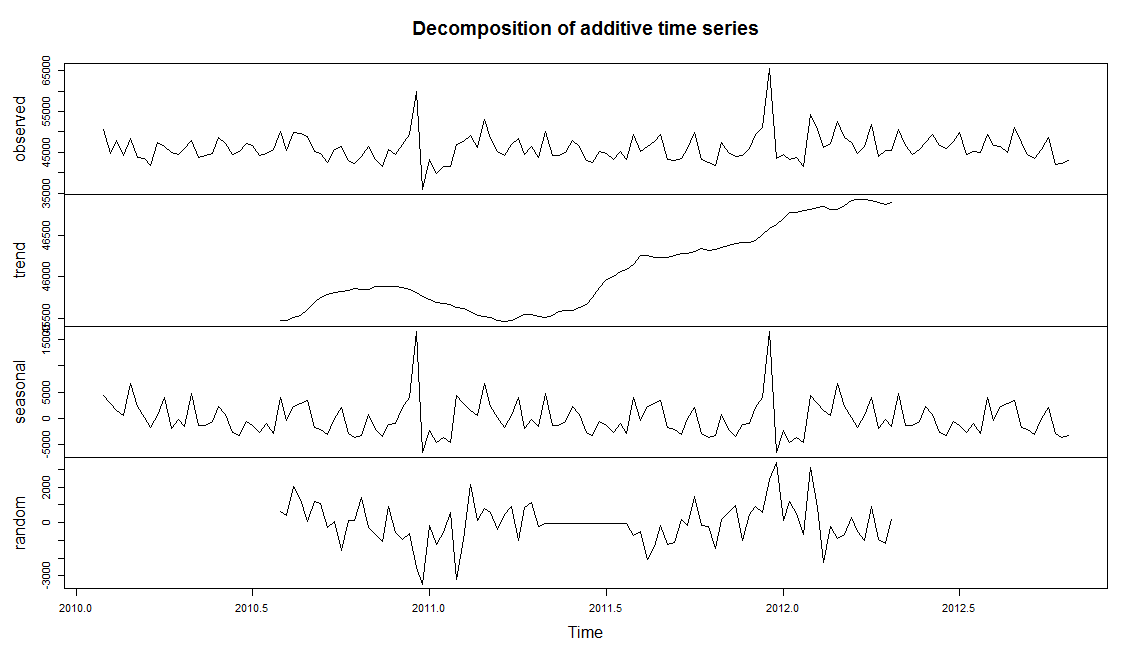
> dataset\_store1\_ts = ts(dataset\_store1$Weekly\_Sales, frequency=52, start = c(2010,5))

> plot(dataset\_store1\_ts)



* We can see from the time plot that this time series could probably be described using an additive model, since the random fluctuations in the data are roughly constant in size over time.
* Decomposition
  + Decomposing a time series means separating it into its constituent components, which are usually a trend component and an irregular component, and if it is a seasonal time series, a seasonal component.
  + To estimate the trend component and seasonal component of a seasonal time series that can be described using an additive model, we can use the “decompose()” function in R

> plot(decompose(dataset\_store1\_ts))



* + The plot above shows the original time series (top), the estimated trend component (second from top), the estimated seasonal component (third from top), and the estimated irregular component (bottom).
  + Findings on the dataset
    - We see that the estimated trend component shows a small decrease from about 46000 in the last quarter of 2010 to about 4500 in the first quarter of 2011, followed by a steady increase from then on to about 47000 in the first quarter of 2012.
  + Breaking dataset into training and validation set
    - Let us break the dataset into training and validation such that forecasting for last 5 months is used for validation set

> train\_store1 = head(dataset\_store1 , 120)

> test\_store1 = tail(dataset\_store1 , nrow(dataset\_store1) - nrow(train\_store1))

|  |
| --- |
| > head(train\_store1,1)  Store Date Weekly\_Sales IsHoliday  1 2010-02-05 50605.27 FALSE  > tail(train\_store1,1)  Store Date Weekly\_Sales IsHoliday  1 2012-05-18 44411.23 FALSE  > head(test\_store1,1)  Store Date Weekly\_Sales IsHoliday  1 2012-05-25 45817.07 FALSE  > tail(test\_store1,1)  Store Date Weekly\_Sales IsHoliday  1 2012-10-26 43134.88 FALSE |
|  |
| |  | | --- | | > | |

* **Naïve Model**
  + All forecasts are simply set to be the value of the last observation.
  + This method works remarkably well for many economic and financial time series.
  + Let us create forecast model and check its accuracy

> train\_store1\_naive = naive(train\_store1\_ts, h = nrow(test\_store1))

> train\_store1\_naive\_forecast = forecast(train\_store1\_naive)

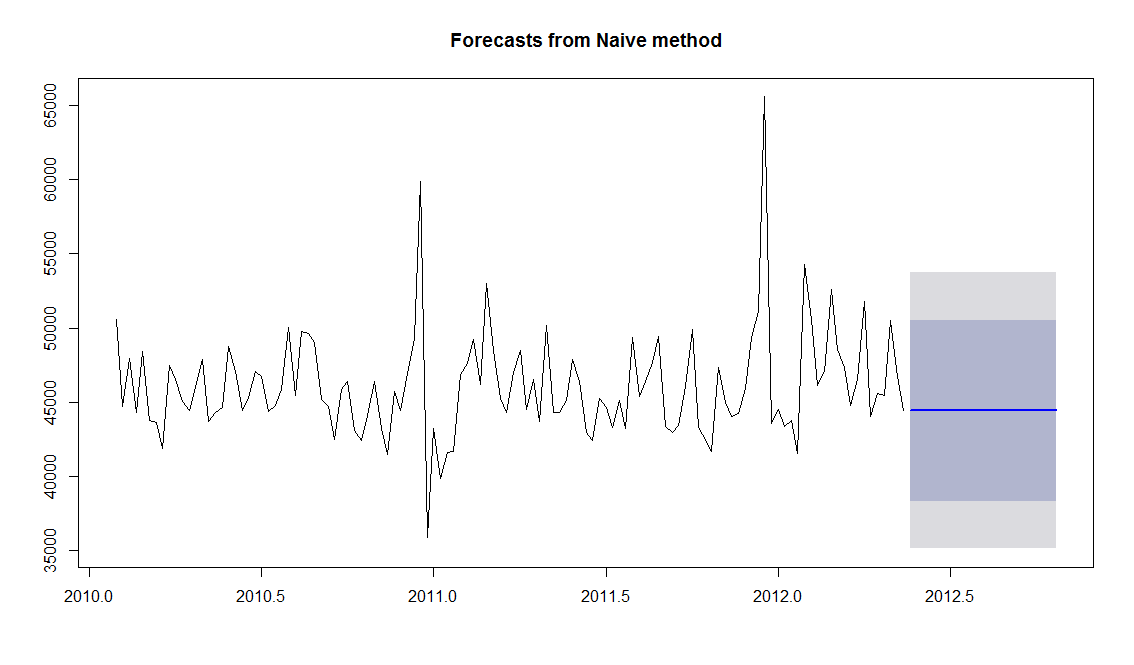
> accuracy(train\_store1\_naive\_forecast, x = test\_store1$Weekly\_Sales)

ME RMSE MAE MPE MAPE MASE ACF1

Training set -52.05076 4763.052 3161.331 -0.6156528 6.841246 1.0000000 -0.4401951

Test set 1812.25348 2961.267 2377.608 3.6744825 5.002397 0.7520909 NA

* + Let us forecast using Naïve forecast model
  + > plot(train\_store1\_naive\_forecast)



* + We can see the MAPE is showing **5% error** on test dataset
* **Random walk (with and without Drift)**
  + A variation on the naïve method is to allow the forecasts to increase or decrease over time, where the amount of change over time (called the drift) is set to be the average change seen in the historical data.
  + Let us create forecast model(**with drift**) and check its accuracy

> train\_store1\_rw\_forecast = rwf(train\_store1\_ts, h = 23, drift = T)

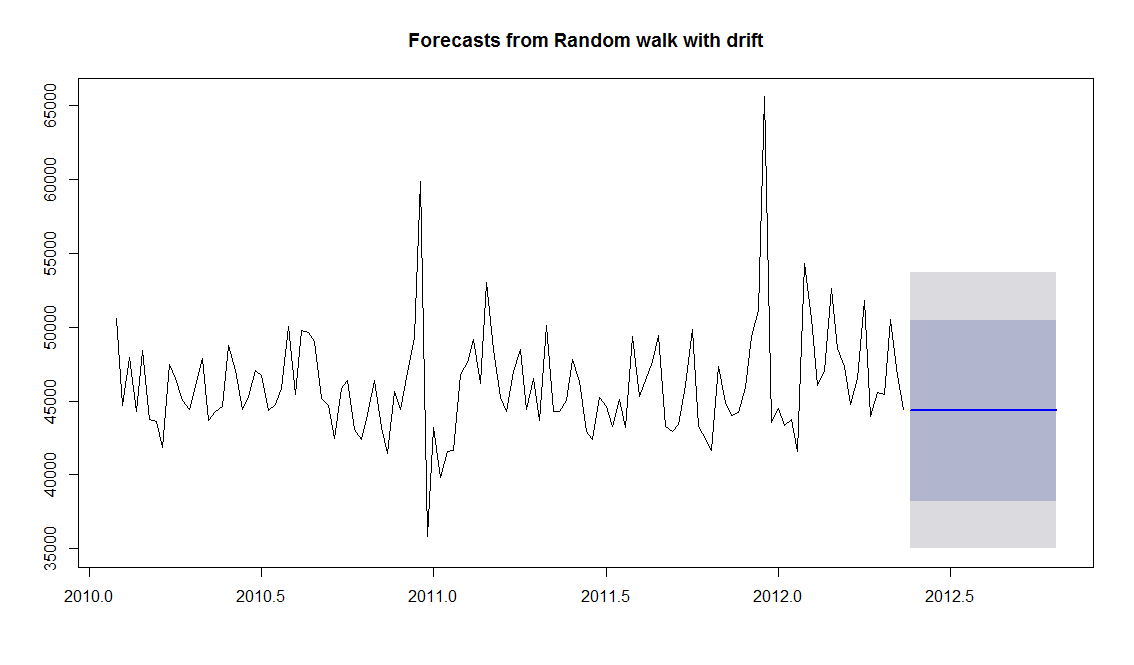
> accuracy(train\_store1\_rw\_forecast, x = test\_store1$Weekly\_Sales)

ME RMSE MAE MPE MAPE MASE ACF1

Training set 2.934818e-12 4762.768 3161.768 -0.5019748 6.838406 1.0001384 -0.4401951

Test set 1.864304e+03 2993.404 2411.554 3.7873777 5.072970 0.7628288 NA

* + Let us plot the forecast



* + Let us create forecast model(**without drift**) and check its accuracy

> # without drift

> train\_store1\_rwd\_forecast = rwf(train\_store1\_ts, h = 23, drift = F)

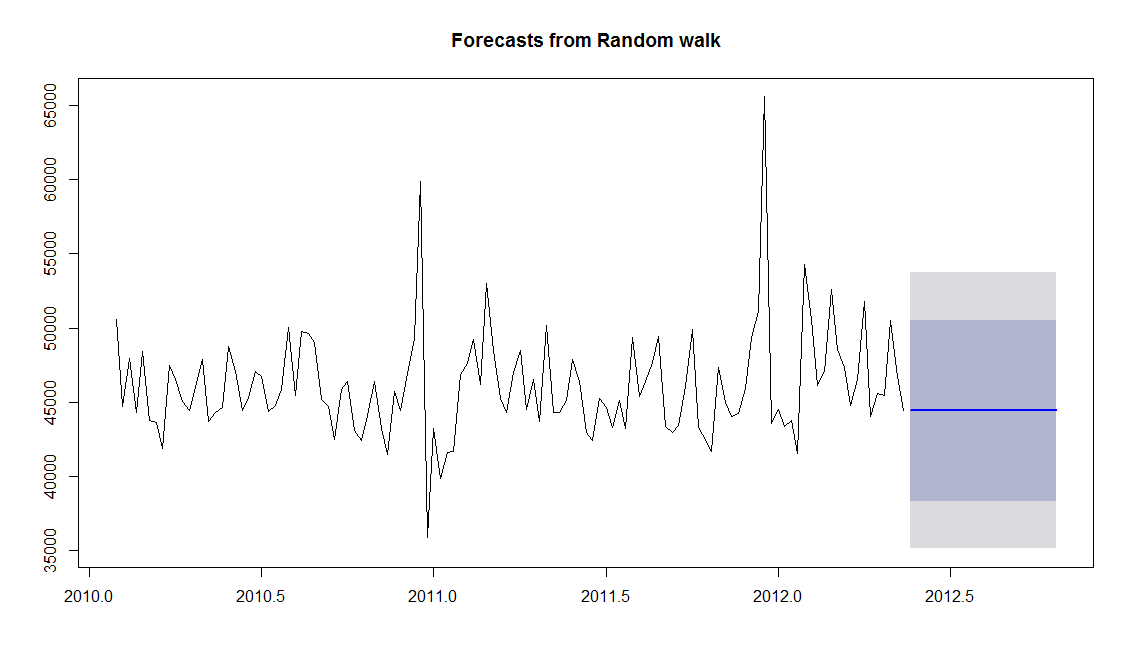
> accuracy(train\_store1\_rwd\_forecast, x = test\_store1$Weekly\_Sales)

ME RMSE MAE MPE MAPE MASE ACF1

Training set -52.05076 4763.052 3161.331 -0.6156528 6.841246 1.0000000 -0.4401951

Test set 1812.25348 2961.267 2377.608 3.6744825 5.002397 0.7520909 NA

* + Let us plot the forecast



* + We can see the MAPE for both the models is showing 5% error on test dataset
* **Average Method**
  + The forecasts of all future values are equal to the mean of the historical data.
  + Let us create forecast model and check its accuracy

> train\_store1\_sma\_forecast = meanf(train\_store1\_ts, h = 23)

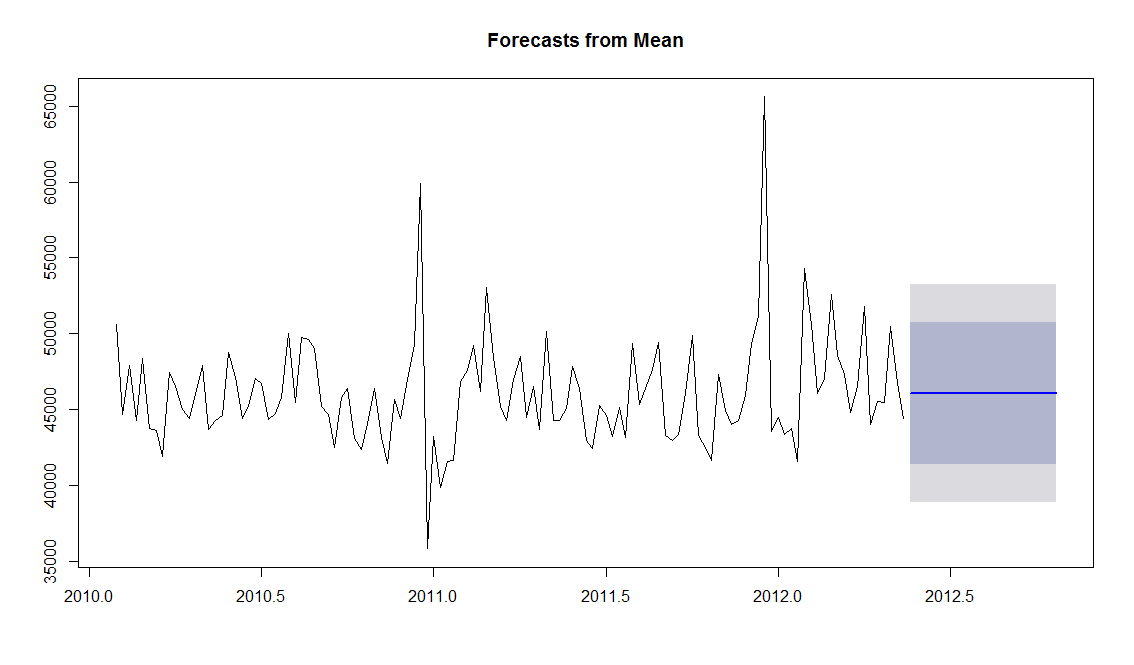
> accuracy(train\_store1\_sma\_forecast, x = test\_store1$Weekly\_Sales)

ME RMSE MAE MPE MAPE MASE ACF1

Training set 0.0000000000006063955 3599.159 2529.788 -0.55557374 5.387728 0.8002289 0.1241463

Test set 144.6600615942035119588 2346.436 1895.198 0.05756399 4.091671 0.5994938 NA

* + Let us plot the forecast

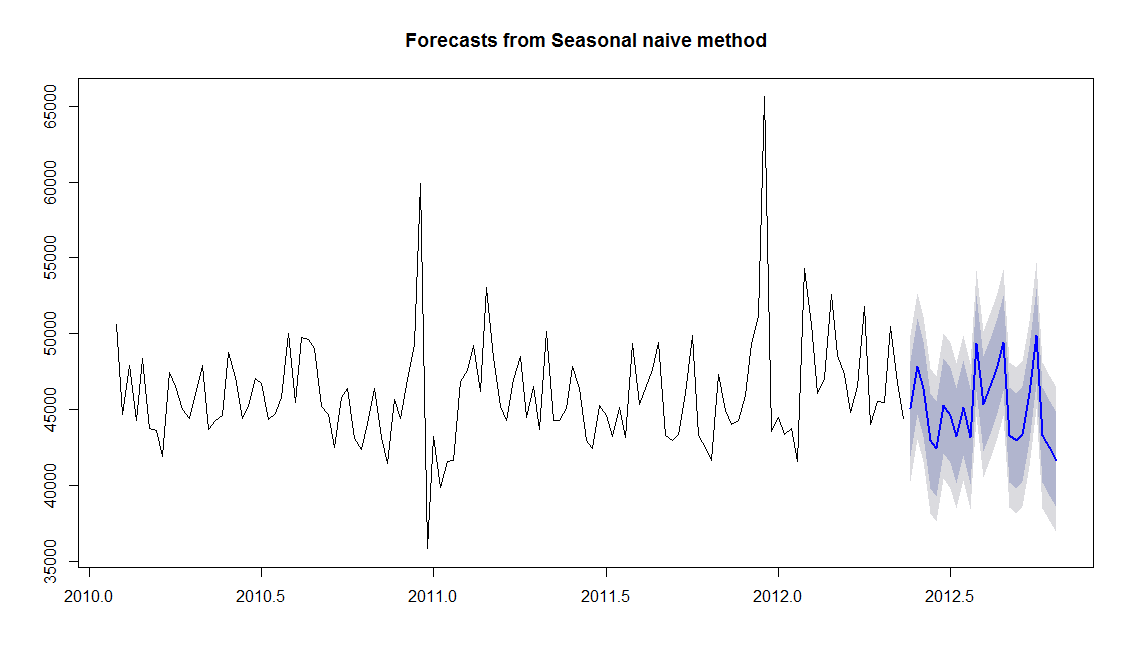


* + We can see the MAPE is showing **4% error** on test dataset
* **Seasonal Naïve Model**
  + A similar method to Naïve forecasting model is useful for highly seasonal data. In this case, we set each forecast to be equal to the last observed value from the same season of the year (e.g., the same month of the previous year)
  + Let us create forecast model and check its accuracy

|  |
| --- |
| > train\_store1\_snaive = snaive(train\_store1\_ts)  > train\_store1\_snaive\_forecast = forecast(train\_store1\_snaive, h = nrow(test\_store1))  > accuracy(train\_store1\_snaive\_forecast, x = test\_store1$Weekly\_Sales)  ME RMSE MAE MPE MAPE MASE ACF1  Training set 677.406 2442.65 1850.487 1.295784 3.915504 0.5853507 0.2834977  Test set 1122.713 2183.04 1648.147 2.355765 3.517350 0.5213458 NA |
|  |
| |  | | --- | | > | |

* + Let us forecast using Seasonal Naïve forecasting model

> plot(train\_store1\_snaive\_forecast)



* + We can see the MAPE is showing **3.5% error** on test dataset
* **Simple Exponential Smoothing(Holts Method)**
  + If we have a time series that can be described using an additive model with constant level and no seasonality, we can use simple exponential smoothing to make short-term forecasts.
  + The simple exponential smoothing method provides a way of estimating the level at the current time point.
  + Let us create forecast model and check its accuracy

> train\_store1\_ses\_model = HoltWinters(train\_store1\_ts, beta = F, gamma = F)

> train\_store1\_ses\_forecast = forecast(train\_store1\_ses\_model, h = nrow(test\_store1))

> accuracy(train\_store1\_ses\_forecast, x = test\_1\_1)

ME RMSE MAE MPE MAPE MASE ACF1

Training set -250.3464 3823.514 2758.757 -0.5707007 5.923347 0.8706616 0.1207606

Test set -772.6863 2466.147 2045.531 -1.9321100 4.497153 0.6470474 NA

* + Let us see the smoothing parameters of the model

> train\_store1\_ses\_model

Holt-Winters exponential smoothing without trend and without seasonal component.

Call:

HoltWinters(x = train\_store1\_ts, beta = F, gamma = F)

Smoothing parameters:

alpha: 0.1211464

beta : FALSE

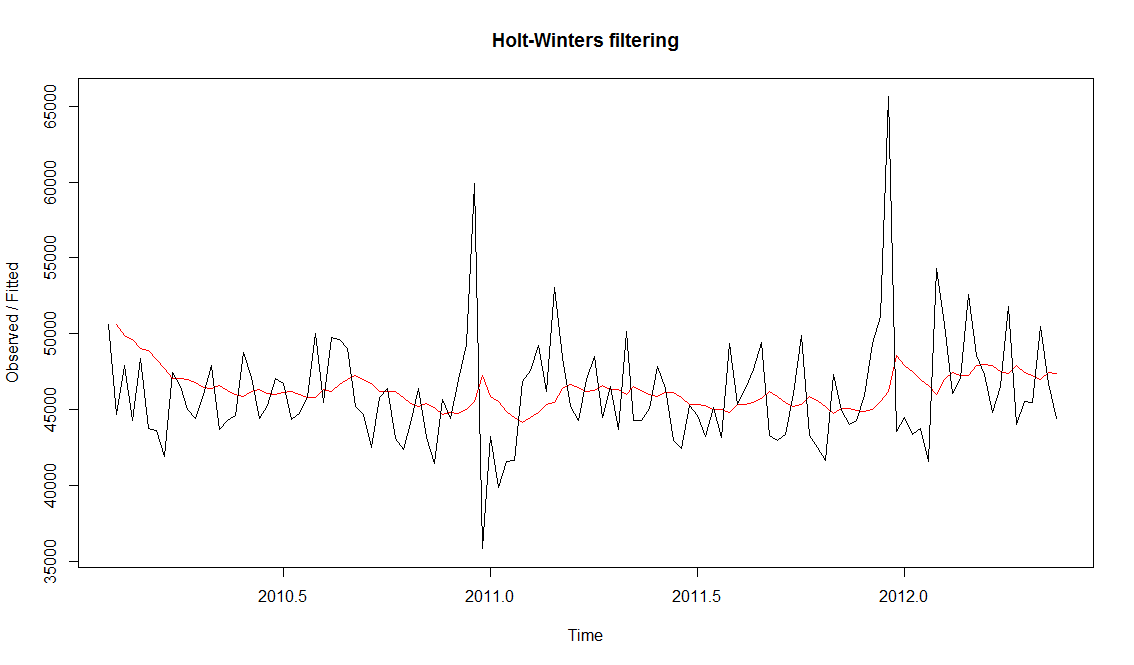
gamma: FALSE

Coefficients:

[,1]

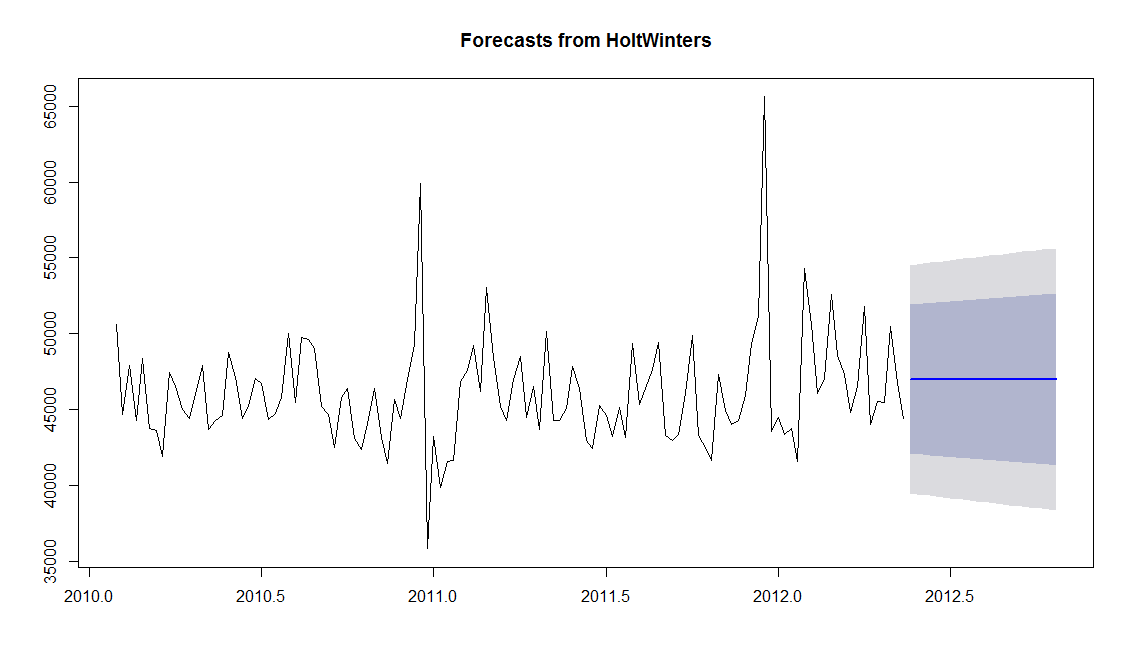
a 46996.17

* + Let us plot the model

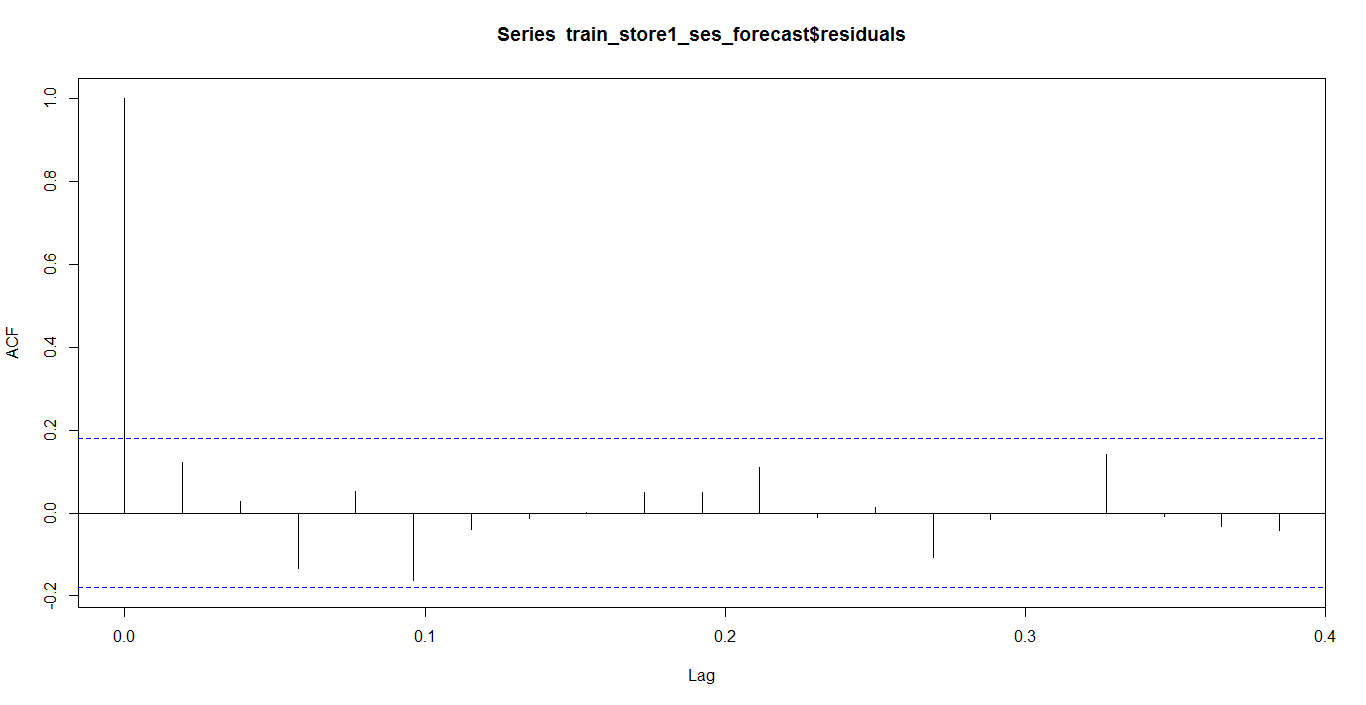


* + The plot shows the original time series in black, and the forecasts as a red line. The time series of forecasts is much smoother than the time series of the original data here.
  + Let us plot the forecast of the model

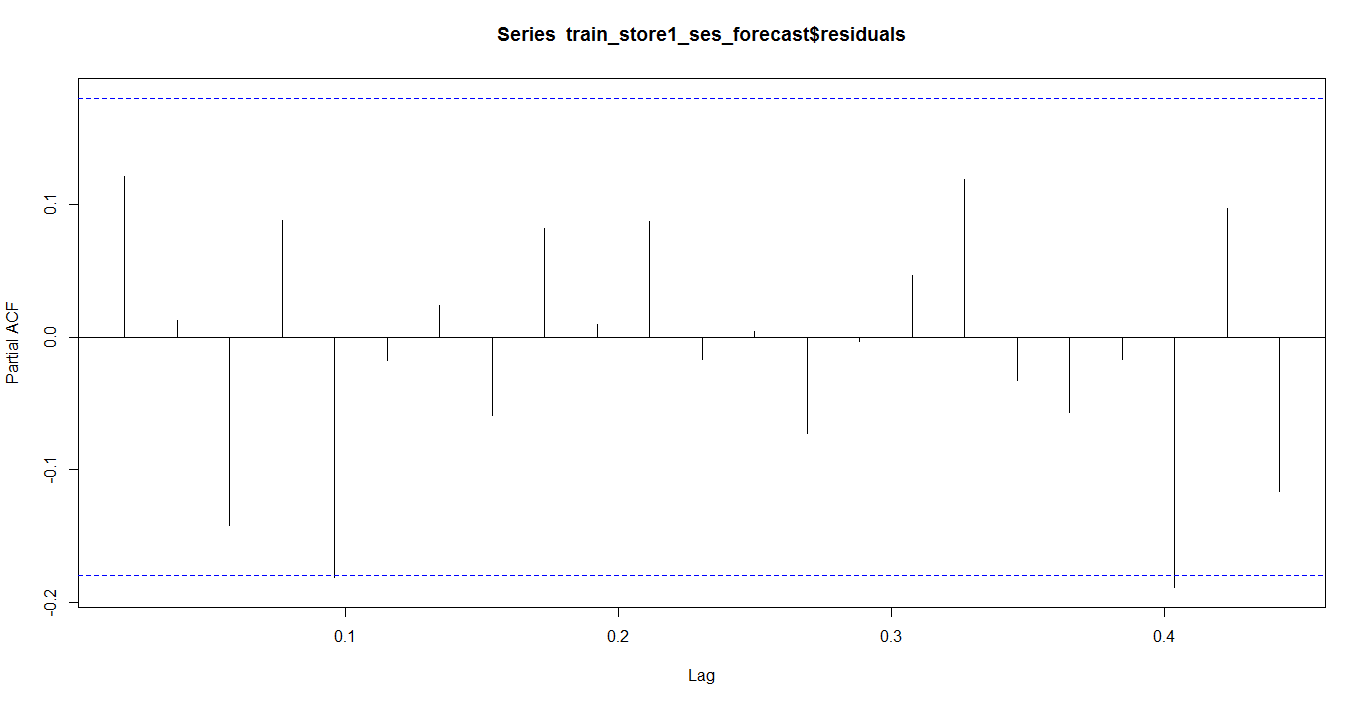
|  |
| --- |
| > plot(train\_store1\_ses\_forecast) |
|  |
| |  | | --- | |  | |



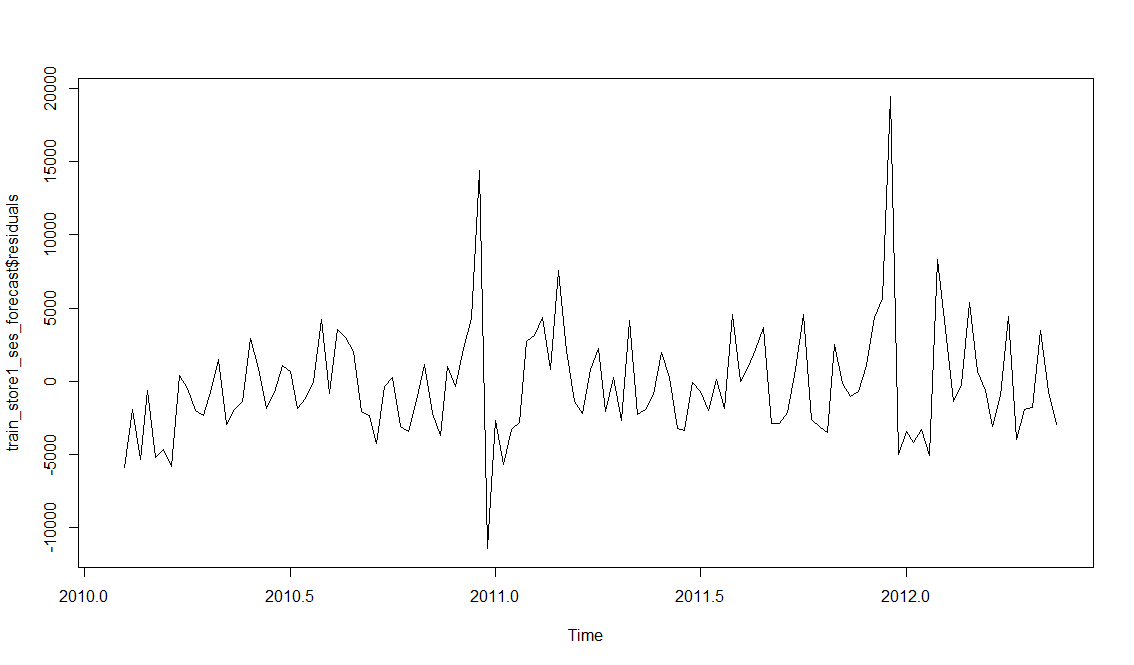
* + The in-sample forecast errors are stored in the named element “residuals” of the list variable returned by train\_store1\_ses\_forecast.HoltWinters(). If the predictive model cannot be improved upon, there should be no correlations between forecast errors for successive predictions. In other words, if there are correlations between forecast errors for successive predictions, it is likely that the simple exponential smoothing forecasts could be improved upon by another forecasting technique.
  + Let Plot the acf plot with lag 23



* + Let us plot the pacf plot with lag 23



* + - We can see from pacf plot that **there might be partial autocorrelation in lag 22**
    - To be sure that the predictive model cannot be improved upon, it is also a good idea to check whether the forecast errors are normally distributed with mean zero and constant variance.
    - The plot shows that the in-sample forecast errors seem to have roughly constant variance over time, although there seems to be a trend at the beginning of the residual set
    - Let us plot the forecast



* + - We can see the MAPE is showing **4.4% error** on test dataset
* **Holts Winter (with Seasonal Component)**
  + If we have a time series that can be described using an additive model with increasing or decreasing trend and no seasonality, we can use Holt’s exponential smoothing to make short-term forecasts.
  + The modelling technique takes care of seasonal component
  + Let us create forecast model and check its accuracy

|  |
| --- |
| > train\_store1\_ses\_model = HoltWinters(train\_store1\_ts, beta = F, gamma = T)  > train\_store1\_ses\_forecast = forecast(train\_store1\_ses\_model, h = nrow(test\_store1))  > accuracy(train\_store1\_ses\_forecast, x = test\_1\_1)  ME RMSE MAE MPE MAPE MASE ACF1  Training set 43.84422 1856.707 1191.085 0.1374687 2.604199 0.3940540 0.04096965  Test set 180.74562 1942.422 1514.367 0.3179742 3.255326 0.4790284 NA |
|  |
| |  | | --- | | > | |

* + Let us see the smoothing parameters

> train\_store1\_ses\_model

Holt-Winters exponential smoothing without trend and with additive seasonal component.

Call:

HoltWinters(x = train\_store1\_ts, beta = F, gamma = T)

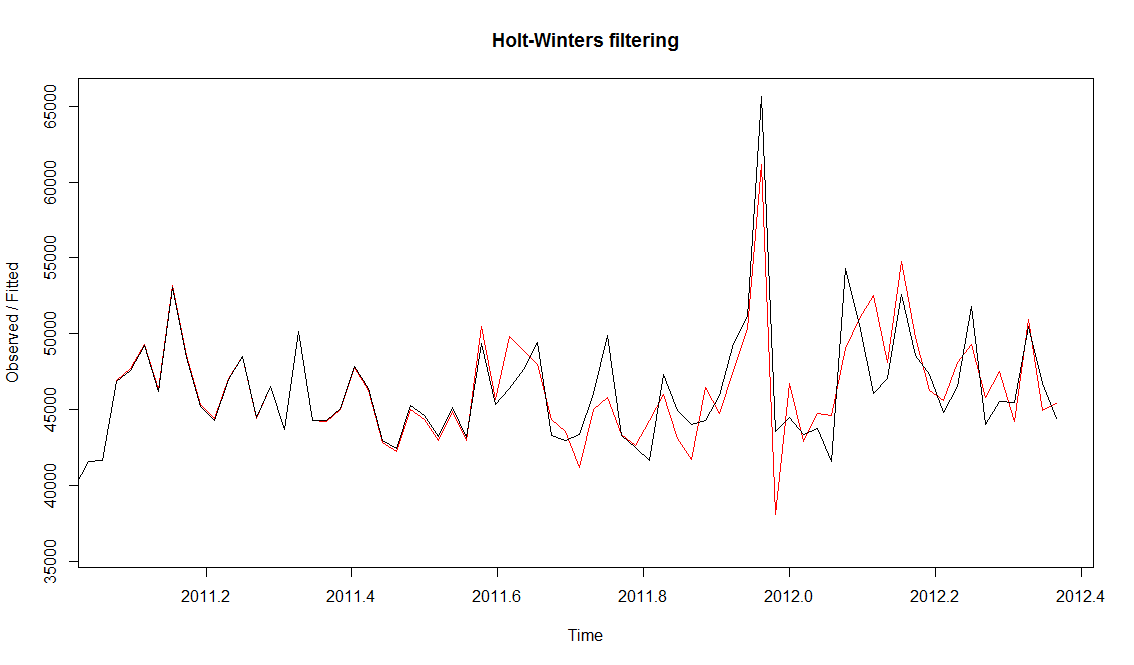
Smoothing parameters:

alpha: 0.220378

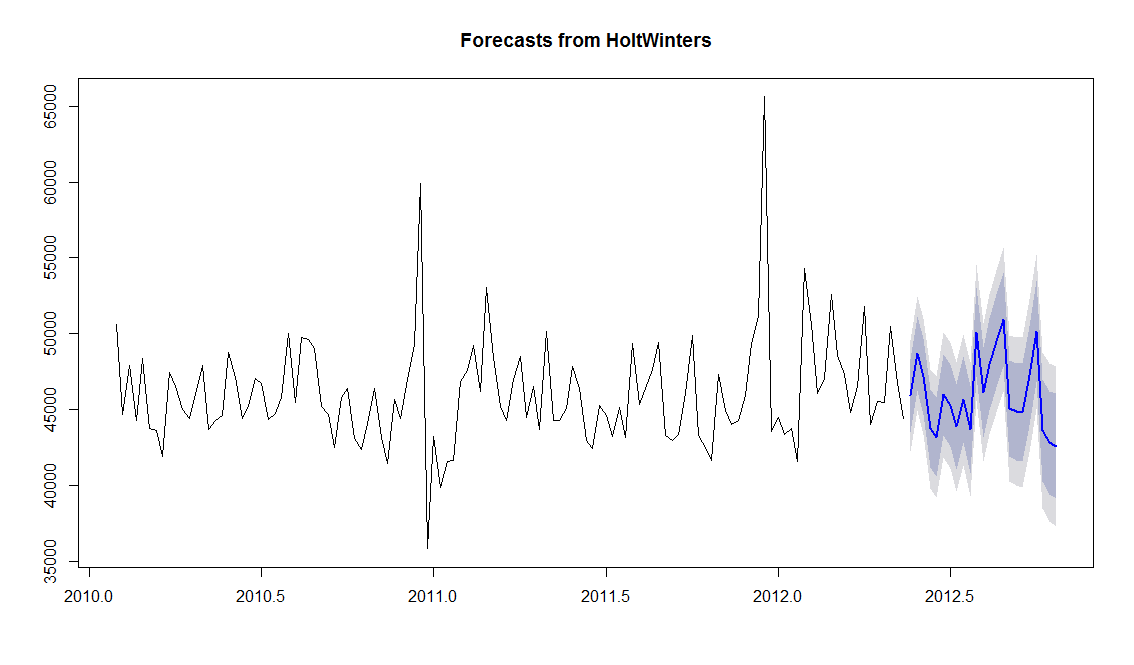
beta : FALSE

gamma: TRUE

* + Let us plot the model

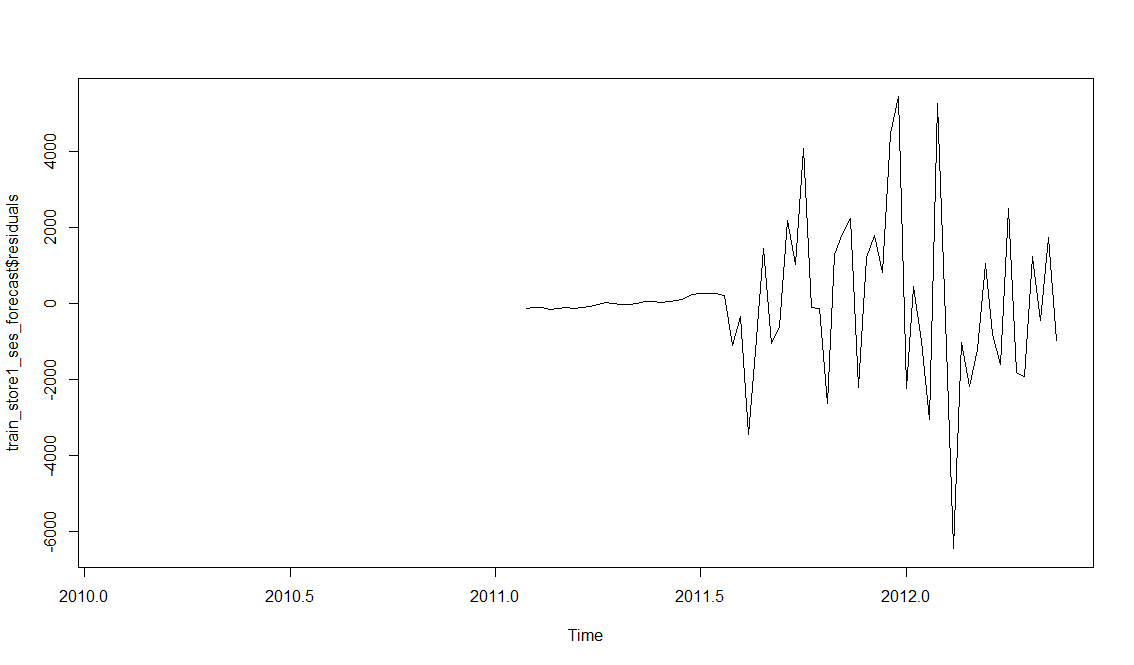


* + - Let us plot the forecast



* + We can plot the residual and see if it has constant variance over time

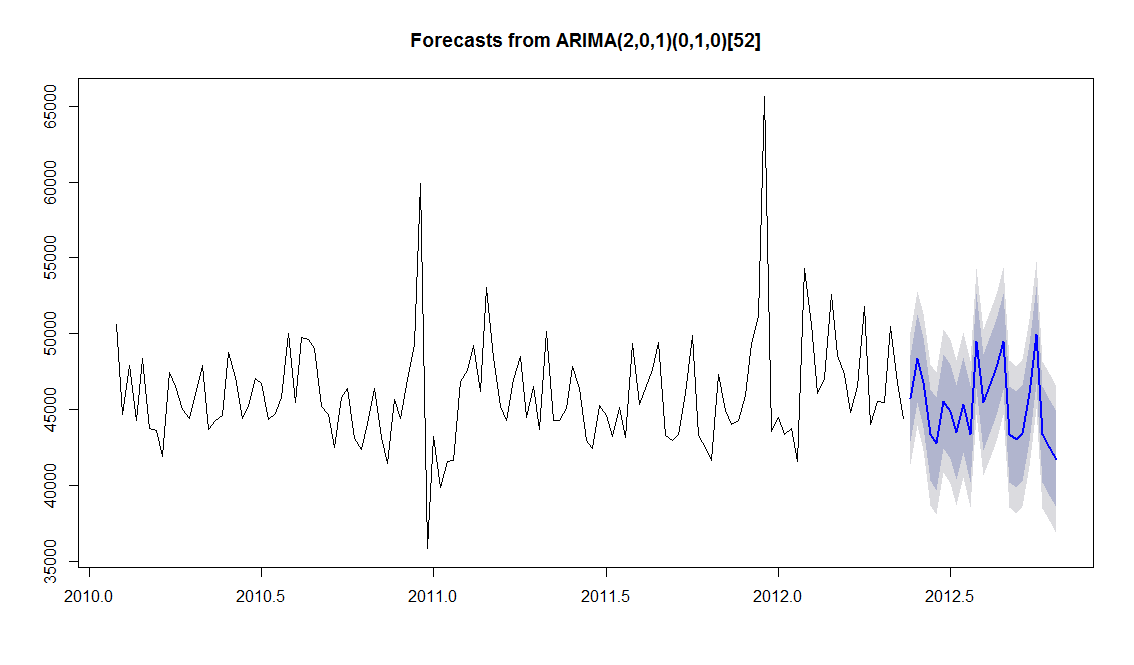
|  |
| --- |
| > plot.ts(train\_store1\_ses\_forecast$residuals) |
|  |
| |  | | --- | |  | |



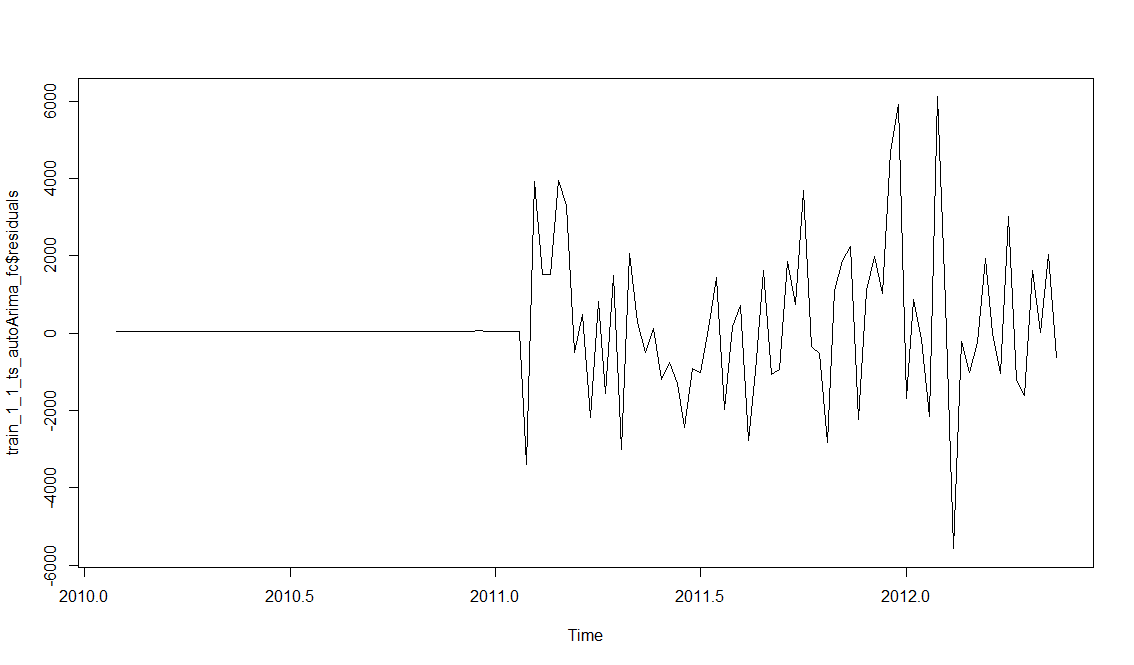
* + We can see that **the variance is almost but not completely constant over time**
  + Hence the model can be improved further.
  + We can see the MAPE is showing **3.25% error** on test dataset
* **Arima**
  + Autoregressive Integrated Moving Average (ARIMA) models include an explicit statistical model for the irregular component of a time series that allows for non-zero autocorrelations in the irregular component.
  + Let us create forecast model and check its accuracy

|  |
| --- |
| > accuracy(train\_1\_1\_ts\_autoArima\_fc, x = test\_store1$Weekly\_Sales)  ME RMSE MAE MPE MAPE MASE ACF1  Training set 171.7291 1636.507 970.5415 0.2789116 2.062491 0.3070041 -0.002996433  Test set 927.5453 2046.653 1555.3196 1.9385334 3.320364 0.4919825 NA |
|  |
| |  | | --- | | > | |

* + Let us plot the forecast
* plot(train\_1\_1\_ts\_autoArima\_fc)



* + Let us plot the residual plot



* + We can see that **the variance is almost constant over time**
  + We can see the MAPE is showing **3.3% error** on test dataset
* **RandomForest**
  + We wondered if we could apply Random Forest to do time series forecasting
  + One neat output of the RF regression is an analysis of which input features were important in the final decision tree used to predict
  + Random Forest is a general Machine Learning Algorithm and is not specific to TimeSeries
  + In this model, we have used created new features using the existing time data time features of dataset and additionally we have considered if there were any holidays during the week as an input feature.
  + Following new features were created
    - Year – the year of the recorded dataset
    - Month – the month of the recorded dataset
    - Day – the day of the recorded dataset
    - Days – the number of days passed in the year of the recorded dataset
    - Logsales – the log of sales for the recorded dataset

|  |
| --- |
| > # Model 7 : Random Forest  > # Random Forest is a general Machine Learning Algorithm and is not specific to TimeSeries  > dataset\_store1\_rf = dataset\_store1  >  > # creating new features from existing features of data set  > dataset\_store1\_rf$year = as.numeric(substr(dataset\_store1\_rf$Date,1,4))  > dataset\_store1\_rf$month = as.numeric(substr(dataset\_store1\_rf$Date,6,7))  > dataset\_store1\_rf$day = as.numeric(substr(dataset\_store1\_rf$Date,9,10))  > dataset\_store1\_rf$days = (dataset\_store1\_rf$month-1)\*30 + dataset\_store1\_rf$day  > dataset\_store1\_rf$IsHoliday[dataset\_store1\_rf$IsHoliday=="TRUE"]=1  > dataset\_store1\_rf$IsHoliday[dataset\_store1\_rf$IsHoliday=="FALSE"]=0  > dataset\_store1\_rf$dayHoliday = dataset\_store1\_rf$IsHoliday\*dataset\_store1\_rf$days  > dataset\_store1\_rf$logsales = log(dataset\_store1\_rf$Weekly\_Sales)  >  > train\_rf = head(dataset\_store1\_rf, 120)  > test\_rf = tail(dataset\_store1\_rf, nrow(dataset\_store1\_rf) - nrow(train\_rf)) |
|  |
| |  | | --- | | > | |

* + - Let us create forecast model and check its accuracy

|  |
| --- |
| > # build the model  > train\_store1\_rf = randomForest(logsales ~ year + month + day + days + dayHoliday ,  + ntree=4800, replace=TRUE, mtry=3, data=train\_rf)  >  >  > # validation of the model  > train\_store1\_rf\_prdt = exp(predict(train\_store1\_rf,test\_rf))  > accuracy(ts(train\_store1\_rf\_prdt), test\_rf$Weekly\_Sales)  ME RMSE MAE MPE MAPE  Test set 482.3968 1659.826 1154.938 0.9641694 2.451545 |
|  |
| |  | | --- | | > | |

* + We can see the MAPE is showing **2.4% error** on test dataset
* **Model Comparison**

|  |  |
| --- | --- |
| Modelling Technique | MAPE |
| Naïve Model | 5% |
| Random walk(with and without drift) | 5% |
| Average Method | 4% |
| Seasonal Naïve Model | 3.5% |
| Simple Exponential Smoothing (Holts Method) | 3.5% |
| Holts Winter (with Seasonal Component) | 3.25% |
| Arima (2,0,1)(0,1,0)[52] | 3.3% |
| RandomForest Model | 2.4% |

* + The MAPE marked in Red are the benchmarks models.
  + All the remaining models are intermediate models
  + The MAPE marked in yellow is yielding the lowest MAPE
* Conclusions
* The analysis done is for a single department / store considering 2 years data pattern to predict the next 5 months or 23 weeks dataset
* MAPE of Bench Mark Model : 5%

Lowest Model MAPE: 2.4%

* MAPE of Bench Mark Model : 5%
* I had tried using Neural Nets, but we feel NN needs more years of data set to be trained along with additional variables to include for exposure of new patterns from Wal-Mart data store.
* We can conclude that, for future modelling purpose, we need to have more years’ data rather than more number of stores data, to increase the accuracy of the model.