Monthly Auto Sales in US - Time Series Analysis using SARIMA

Priyaranjan Pattnayak

December 19, 2017

Time Series Analysis to predict auto sales in 2018 using ARIMA/SARIMA

Pre-requisites

Previous knowledge of time-series forecasting is not required. However, basic working of R commands, use og ggplot and basic data analytics/statistics concepts are required.

We will only use ggplot2,forecast,grid,Amelia,tseries,scales,gridExtra and lmtest.

What is Time Series Analysis? And why is it required?

'Time' is the most important factor which ensures success in a business. It's difficult to keep up with the pace of time. But, technology has developed some powerful methods using which we can 'see things' ahead of time. One such method, which deals with time based data is Time Series Modeling. As the name suggests, it involves working on time (years, days, hours, minutes) based data, to derive hidden insights to make informed decision making.

Time series models are very useful models when you have serially correlated data. Most of business houses work on time series data to analyze sales number for the next year, website traffic, competition position and much more.

Scope of this Project

This project will provide a step-by-step guide for fitting an ARIMA model using R. ARIMA models are a popular and flexible class of forecasting model that utilize historical information to make predictions. This type of model is a basic forecasting technique that can be used as a foundation for more complex models. In this project, we walk through an example of examining time series for predicting domestic auto sales for 2018 in US, fitting an ARIMA model, and creating a basic forecast.

Data Source

U.S. Bureau of Economic Analysis, Motor Vehicle Retail Sales: Domestic Autos*, retrieved from FRED, Federal Reserve Bank of St. Louis (https://fred.stlouisfed.org/series/DAUTONSA)

Data can also be obtained from : Federal Reserve Bank of St. Louis (https://fred.stlouisfed.org/series/DAUTONSA?utm_source=series_page&utm_medium=related_content&utm_term=other_formats&utm_campaign=other_format)

Release: Supplemental Estimates, Motor Vehicles
Units: Thousands of Units, Not Seasonally Adjusted

Frequency: Monthly

Autos are all passenger cars, including station wagons. Domestic sales are all United States (U.S.) sales of vehicles assembled in the U.S., Canada, and Mexico.

Read the data from the csv file (Remember to replace the filepath)

```
monthly_sales <- read.csv("G:\\RProject\\Time Series Stock Returns\\Auto Sales\\DAUTONSA.csv",header = TRUE, stringsAsFactors = FALSE)
```

The dataset looks like:

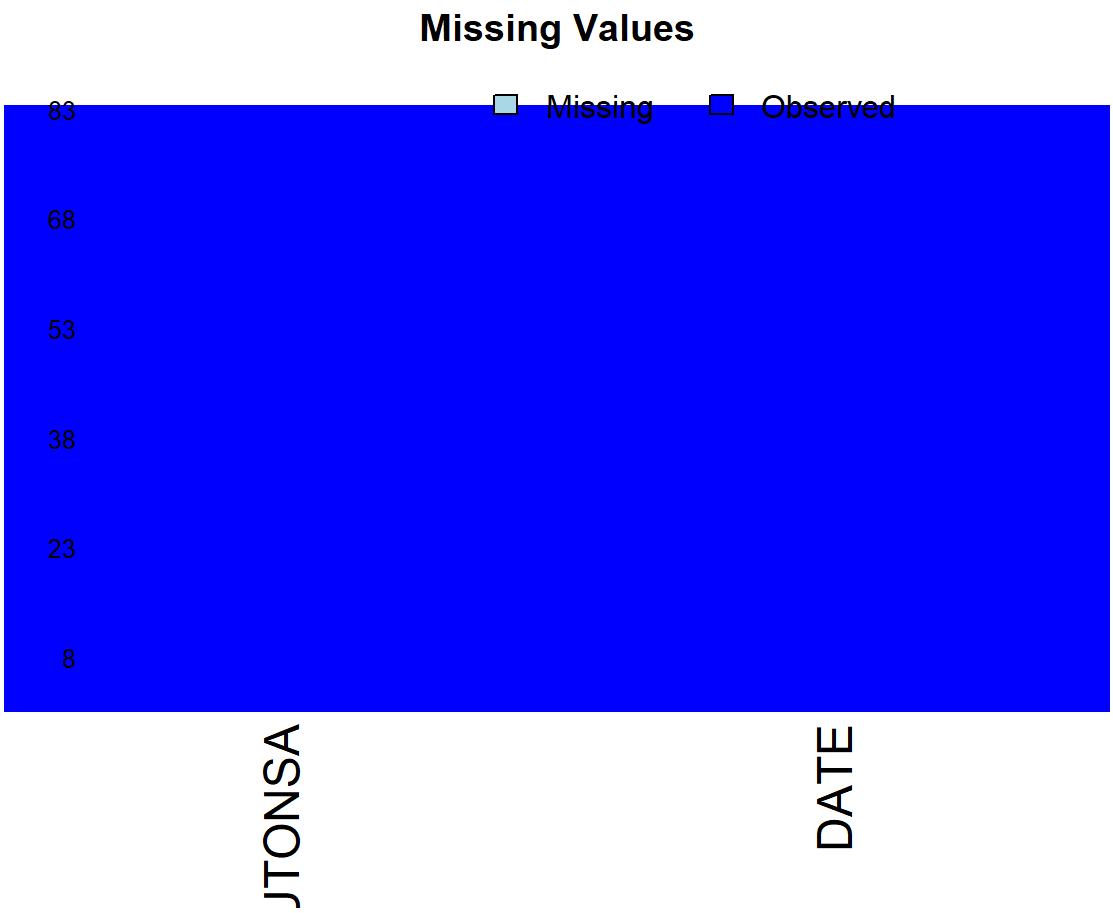
```
head(monthly_sales)
```

```
##       DATE DAUTONSA
## 1 1/1/2011    258.6
## 2 2/1/2011    338.6
## 3 3/1/2011    445.3
## 4 4/1/2011    405.4
## 5 5/1/2011    362.2
## 6 6/1/2011    349.0
```

We will be using the monthly domestic auto sales(in thousands) for building our time series

Before we proceed, let’s check if we have any missing data. Amelia library gives us a missmap function that shows the missing details in a visual map.

```
missmap(monthly_sales,main = "Missing Values",col = c('light blue','Blue'),x.cex = 1.5)
```



Since **no missing data** is found, we can go ahead with visually plotting the daily count to see if we can identify any trends, seasonality, cycle or outlier from the data.

Before we plot the count data, we need to convert the *dteday* field from character to date type.

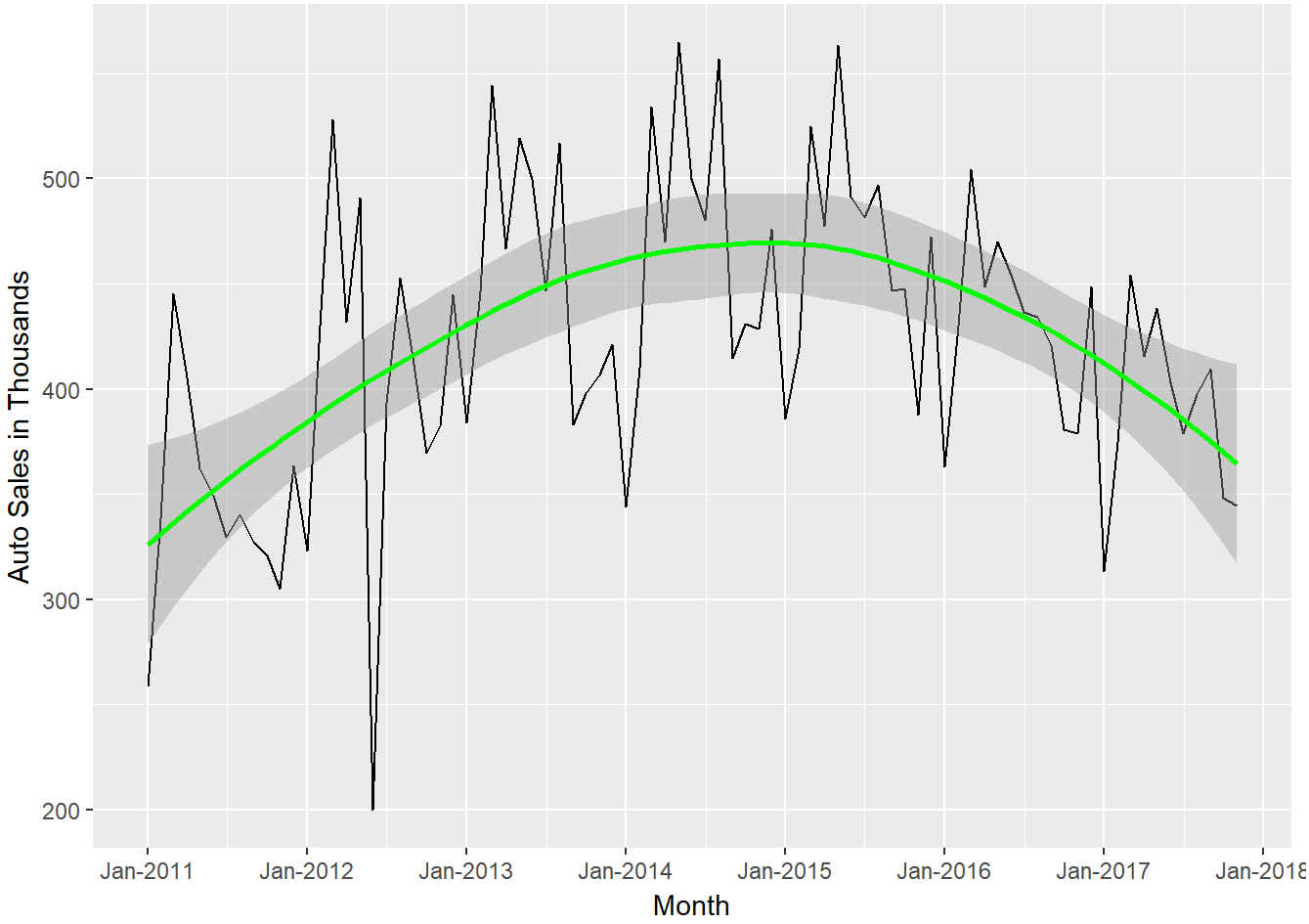
```
monthly_sales$DATE <- as.Date(monthly_sales$DATE, "%m/%d/%Y")
```

We plot the monthly sales data:

```
uc_ts_plot <- ggplot(monthly_sales, aes(DATE,DAUTONSA)) + geom_line(na.rm=TRUE) +
  xlab("Month") + ylab("Auto Sales in Thousands") +
  scale_x_date(labels = date_format(format= "%b-%Y"),breaks = date_breaks("1 year")) +
  stat_smooth(colour = "green")

uc_ts_plot
```

```
## `geom_smooth()` using method = 'loess'
```



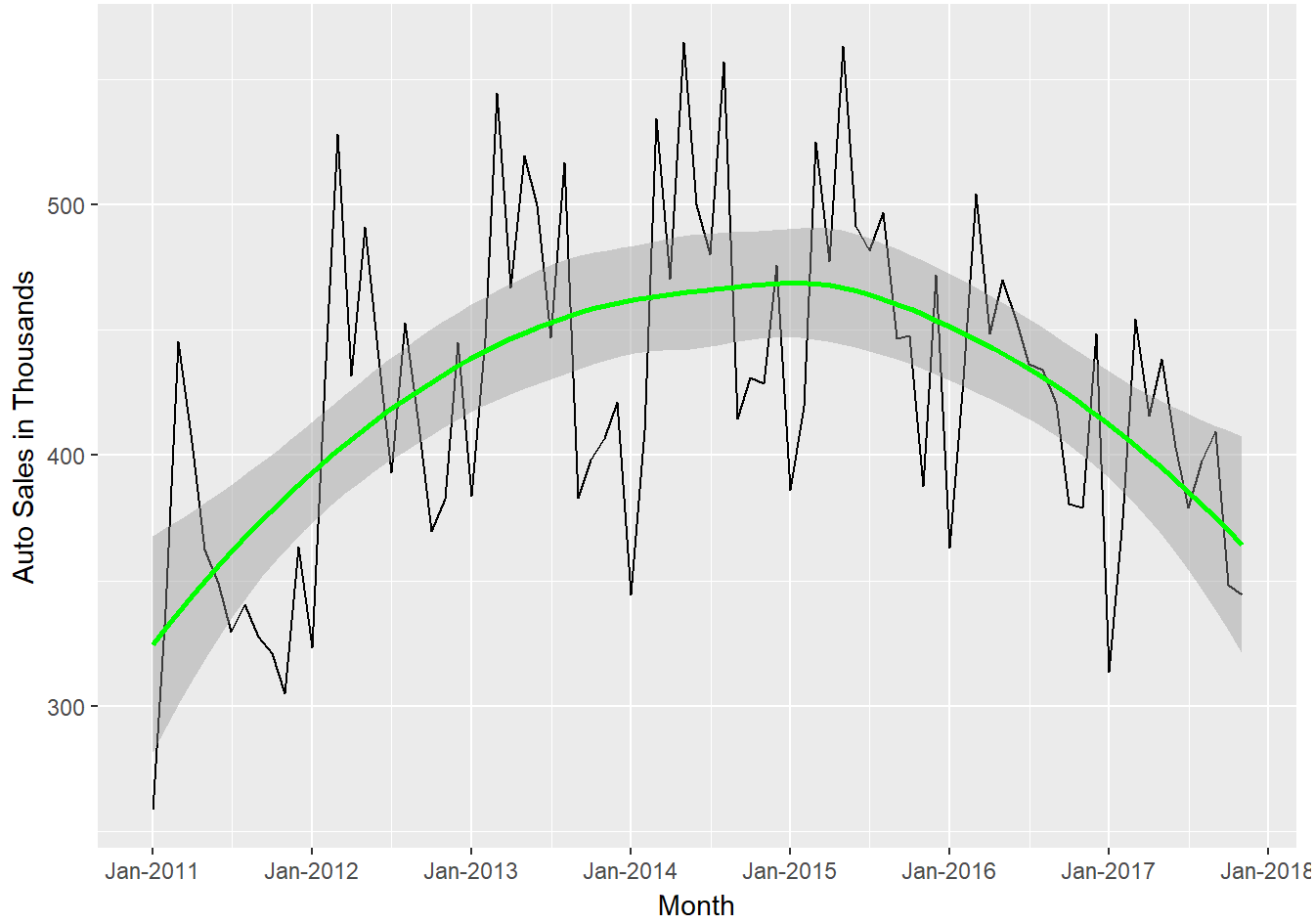
There seems to be an outlier that we could see from the plot. This suspected outlier can bias the model by skewing statistical summaries. R provides a convenient method for removing time series outliers: tsclean() as part of its forecast package. tsclean() identifies and replaces outliers using series smoothing and decomposition.

We need to remove the outlier before we proceed with stationarizing the series. tsclean() is also capable of inputing missing values in the series if there are any. We are using the ts() command to create a time series object to pass to tsclean():

```
monthly_ts <- ts(monthly_sales[,c('DAUTONSA')])
monthly_sales$csales <- tsclean(monthly_ts)
```

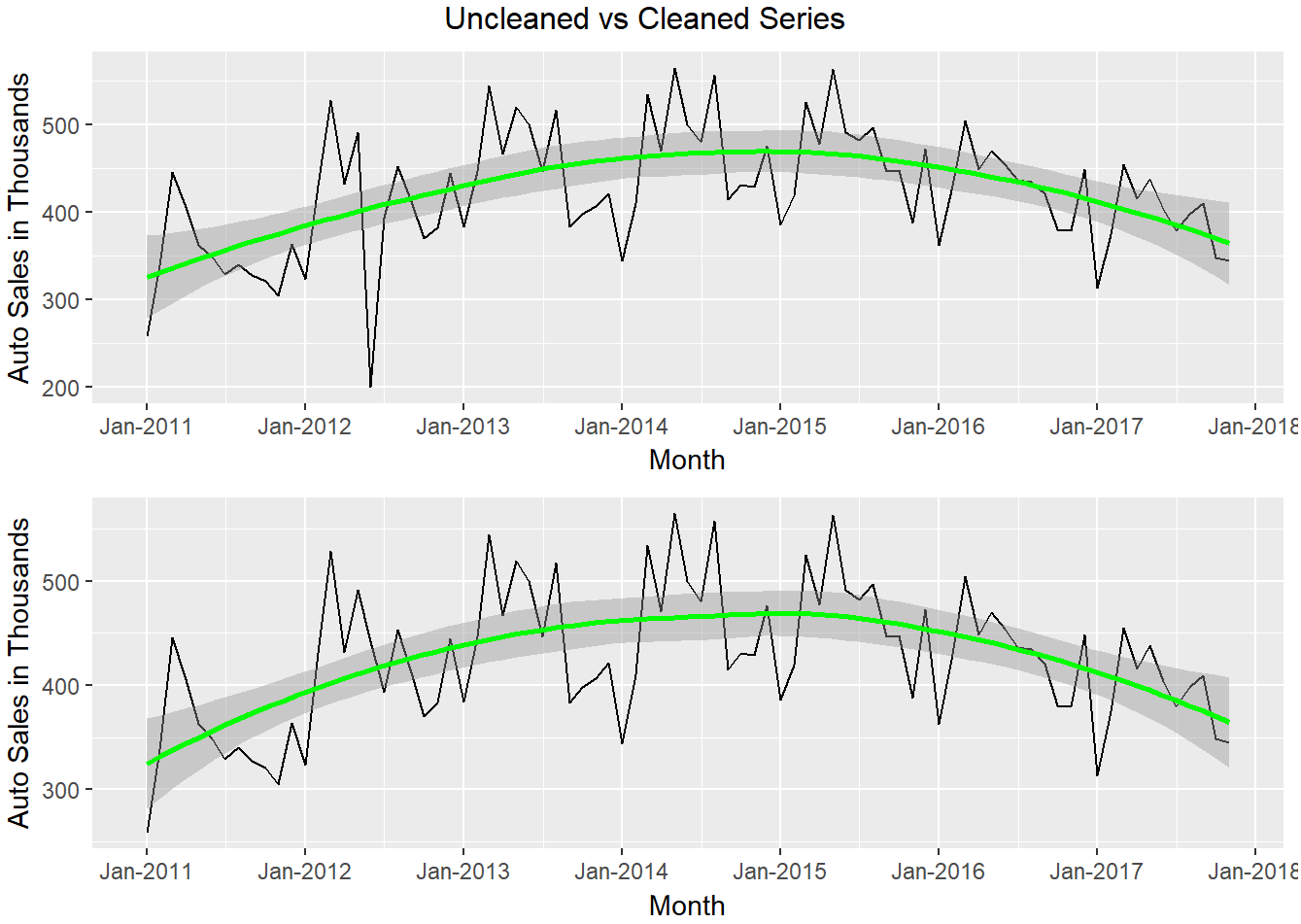
Plot the cleaned monthly sales data:

```
c_ts_plot <- ggplot(monthly_sales, aes(DATE,csales)) + geom_line(na.rm=TRUE) +
  xlab("Month") + ylab("Auto Sales in Thousands") +
  scale_x_date(labels = date_format(format= "%b-%Y"),breaks = date_breaks("1 year")) +
  stat_smooth(colour="green")
c_ts_plot
```



Now, let's compare both cleaned and uncleaned plots:

```
grid.arrange(uc_ts_plot,c_ts_plot,ncol=1, top = textGrob("Uncleaned vs Cleaned Series"))
```



Smoothing the series

If data points are still volatile, then we can apply smoothing smoothing. By applying smoothing, we can have a better idea about the series and it's components. It also makes the series more predictable. In this case, we could have used quarterly/biannualy moving average. If the data points are on a daily basis,many level of seasonality(daily, weekly, monthly or yearly) can be incorporated.

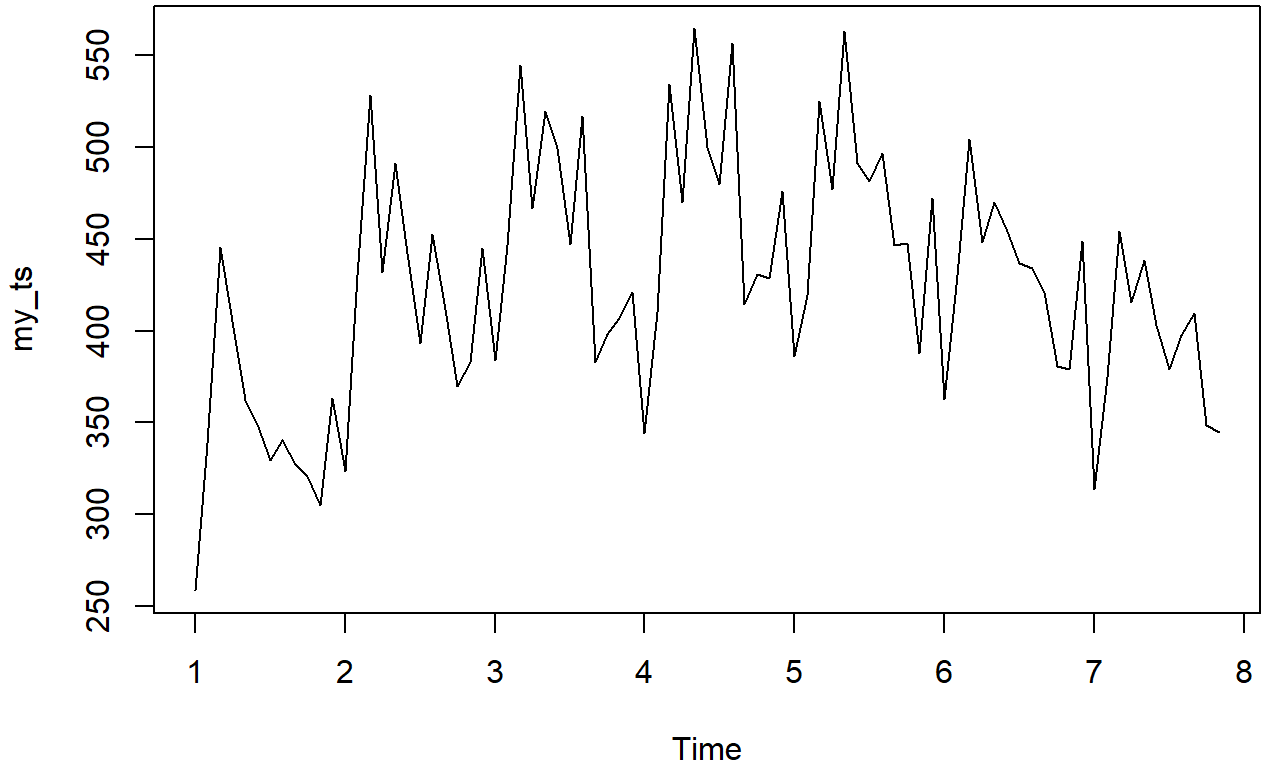
However, looking at the graph, our data does not require any smoothing. Therefore, We go ahead with the cleaned data.

```
my_ts <- ts(na.omit(monthly_sales$csales), frequency = 12)
```

As our data is monthly, we used *frequency =12* in above command. We also ignore the NA values.

Next, we plot the cleaned series to infer visual cues from the graph.

```
plot(my_ts)
```



Identify Level of Differencing Required

Now that the series is cleaned, we need to remove trend by using appropriate order of difference and make the series stationary. We do this by looking at acf, Dickey-Fuller Test and standard deviation.

Dickey Fuller test:

X(t) = Rho * X(t-1) + Er(t)

=> X(t) - X(t-1) = (Rho - 1) X(t - 1) + Er(t)

We have to test if Rho - 1 is significantly different than zero or not. If the null hypothesis gets rejected, we'll get a stationary time series.

Stationary testing and converting a series into a stationary series are the most critical processes in a time series modelling. We need to memorize each and every detail of this concept to move on to the next step of time series modelling.

To confirm that the series is not stationary, we perform the augmented Dickey-Fuller Test.

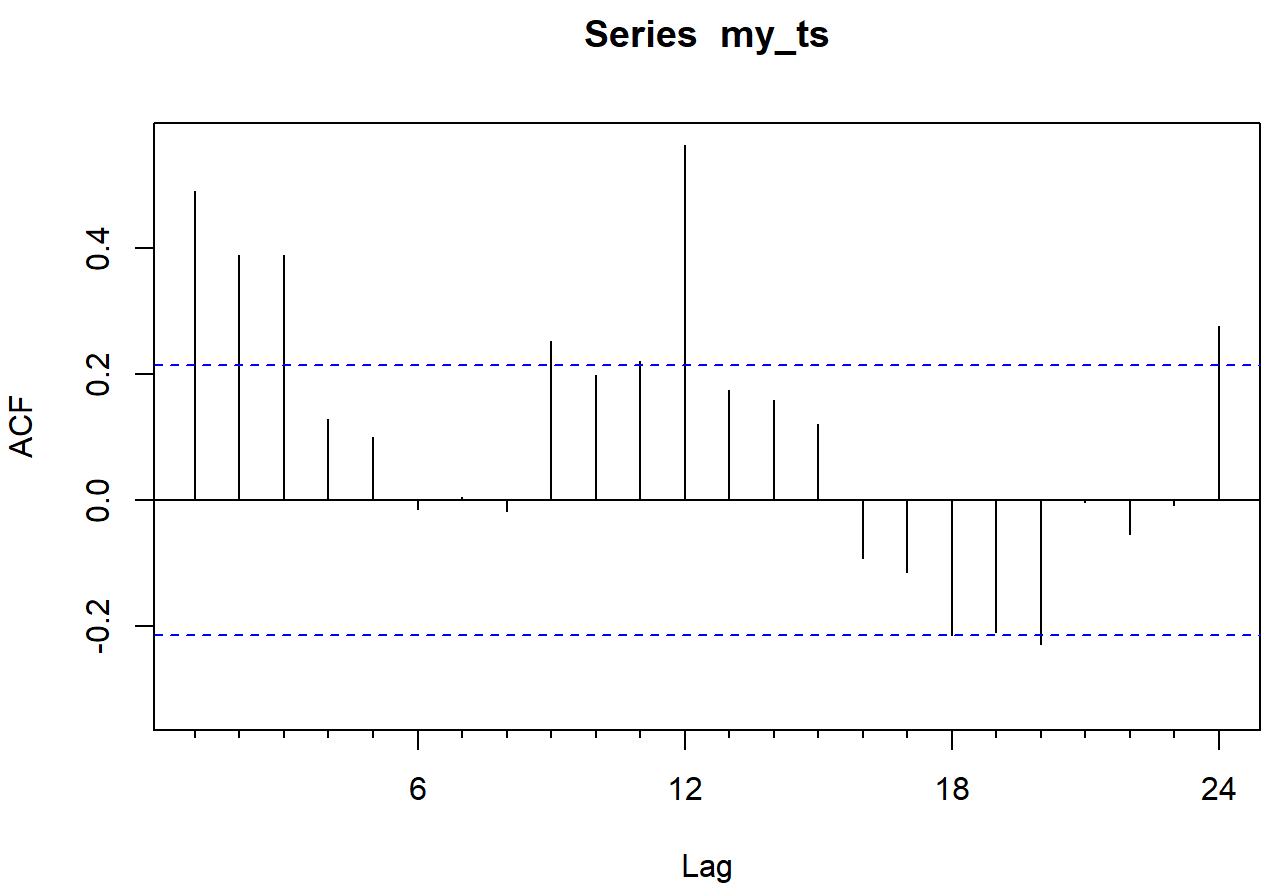
```
adf.test(my_ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  my_ts
## Dickey-Fuller = -2.9078, Lag order = 4, p-value = 0.2037
## alternative hypothesis: stationary
```

P value is 0.2037 indicating the null hypothesis 'series is non-stationary' is true i.e the series is not stationary

Plot the auto-correlation plot for the series to identify the order of differencing required.

```
Acf(my_ts)
```



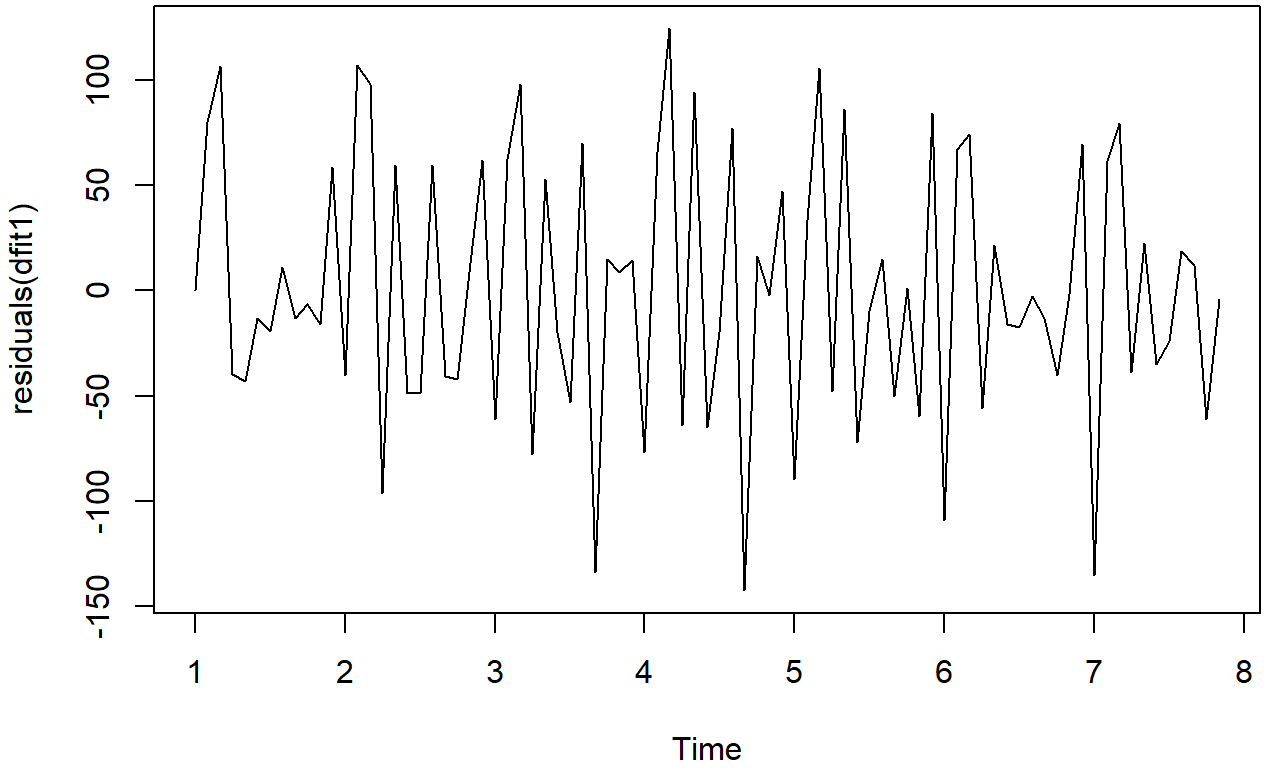
ACF plot shows positive correlation at higher lags. this indicates that we need differencing to make the series stationary.

Let's try order 1 difference

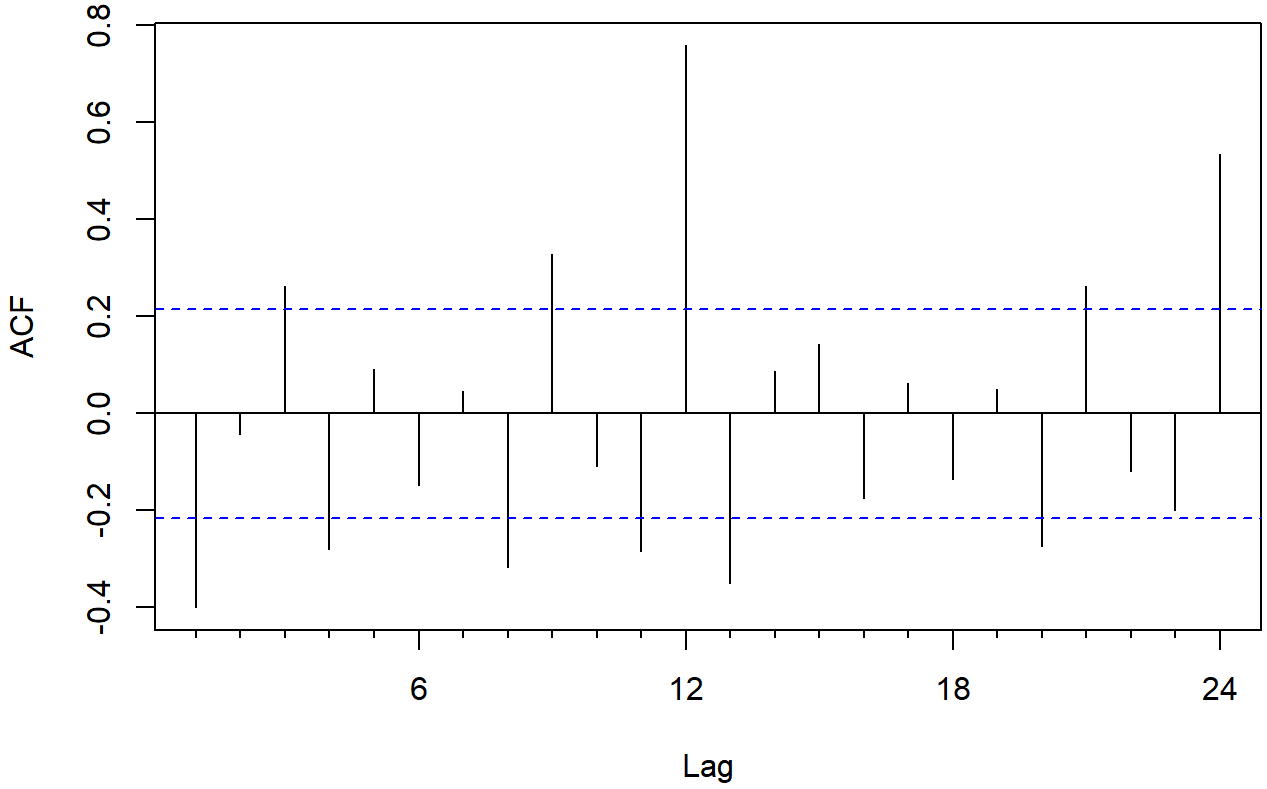
We will fit ARIMA(0,d,0)(0,D,0)[12] models and verify acf residuals to find which 'd' or 'D' order of differencing is appropriate in our case.

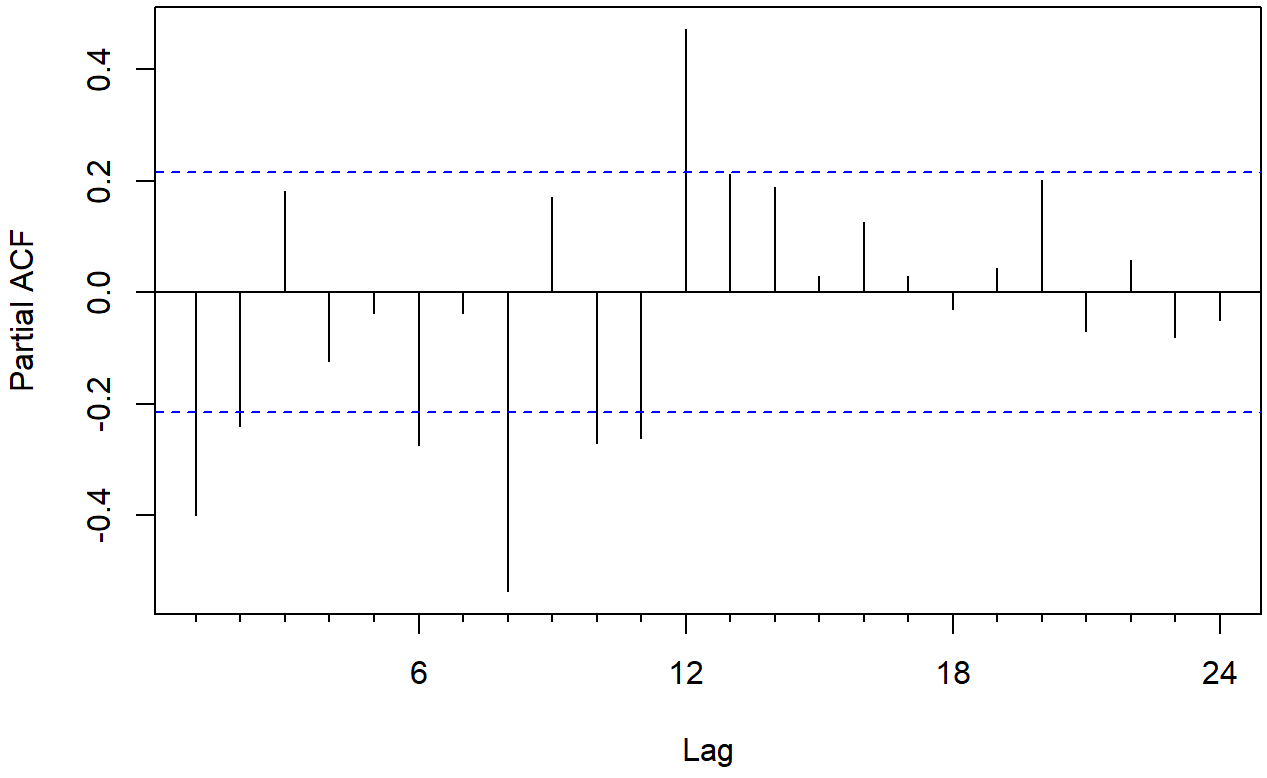
Applying only one order of difference i.e ARIMA(0,1,0)(0,0,0)

```
dfit1 <- arima(my_ts,order = c(0,1,0))
plot(residuals(dfit1))
```


residuals(dfit1)
100
50
0
-50
-100
-150
1
2
3
4
5
6
7
8
Time

Below is the acf and Pacf plot of residuals.

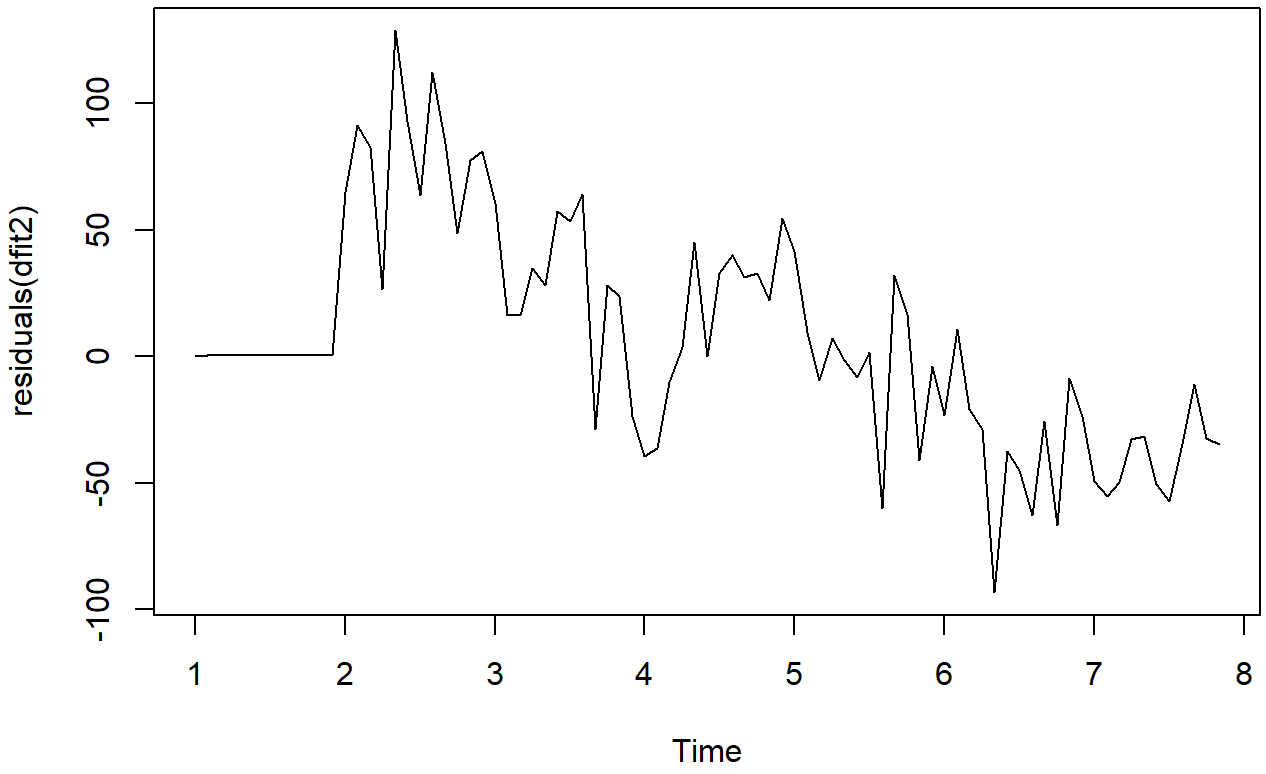




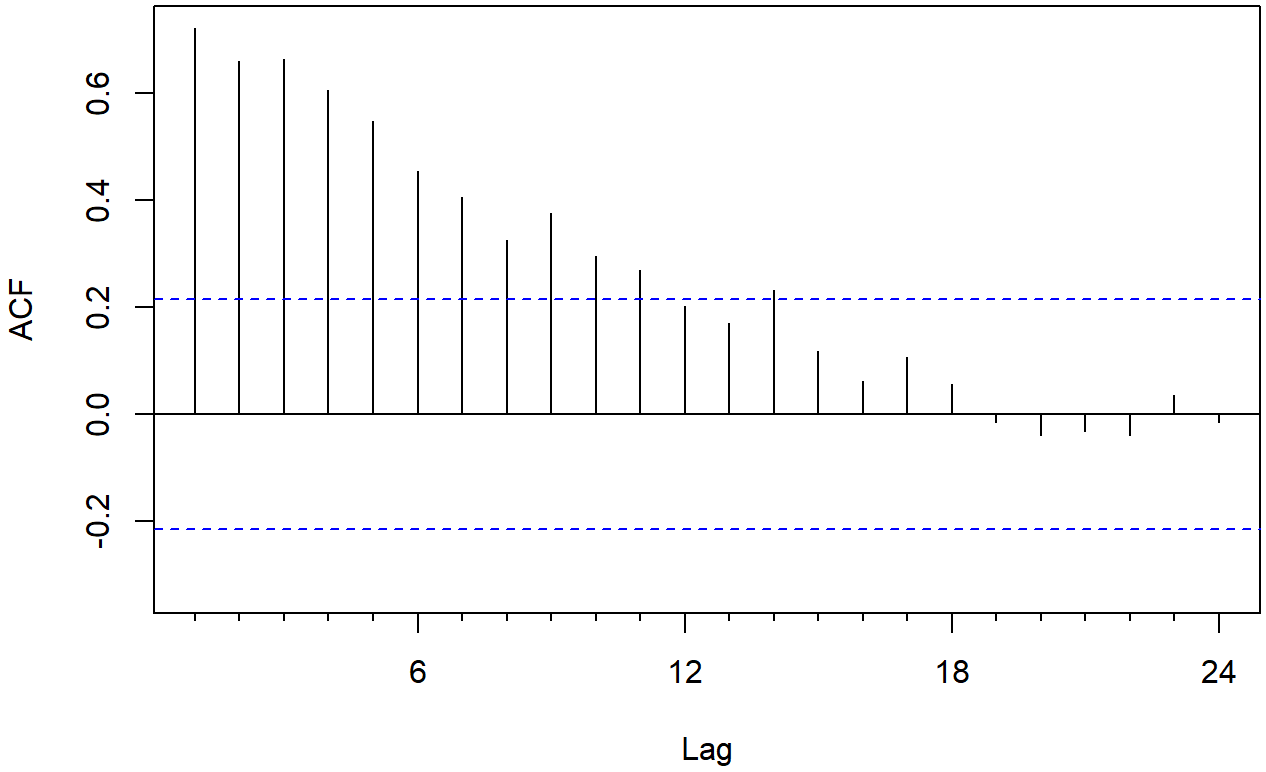
The differenced series still shows some strong autocorrelation at the seasonal period 12. Because the seasonal pattern is strong and stable, we know that we will want to use an order of seasonal differencing in the model.

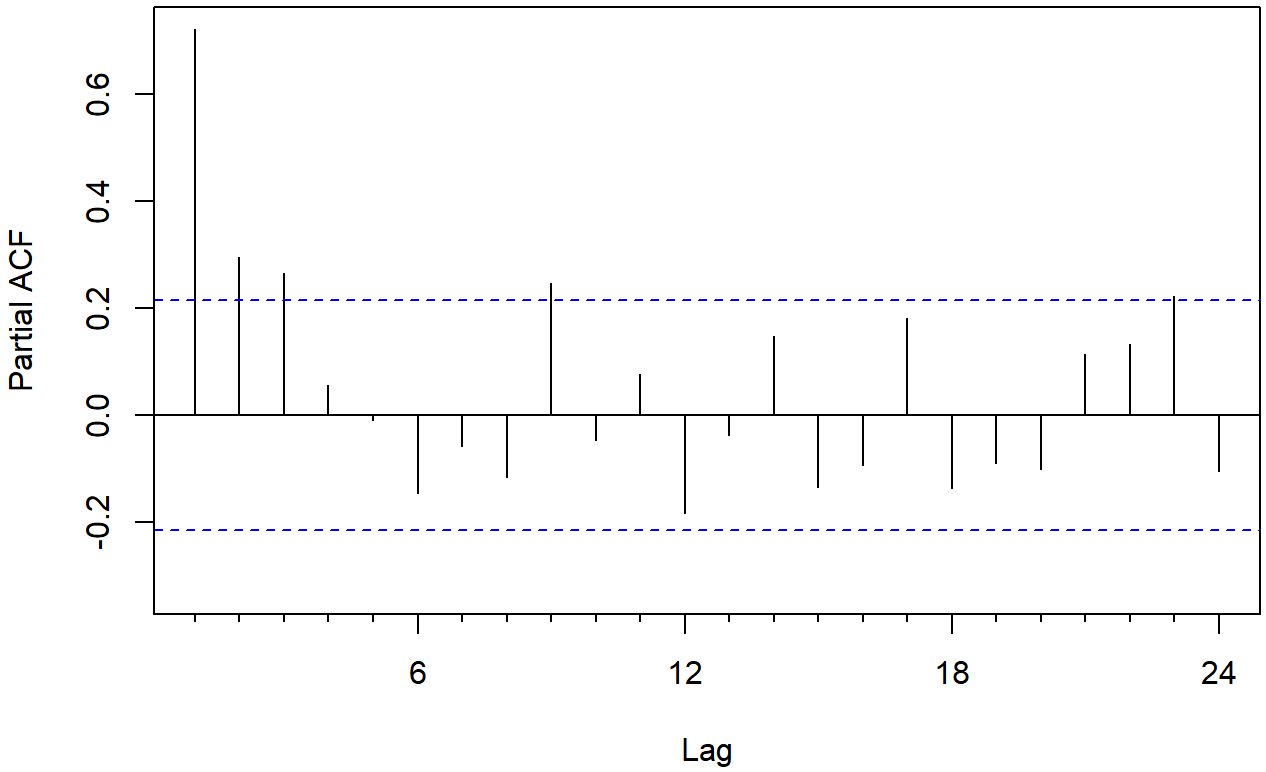
Before that let's try only with one seasonal difference i.e ARIMA(0,0,0)(0,1,0)

```
dfit2 <- arima(my_ts, order =c(0,0,0), seasonal = list(order = c(0,1,0), period = 12))
plot(residuals(dfit2))
```



Looking a the residual plot, residuals does not look like white noise. We also need to check the acf and pacf plots of residuals.


ACF
0.6
0.4
0.2
0.0
-0.2
6
12
18
24
Lag

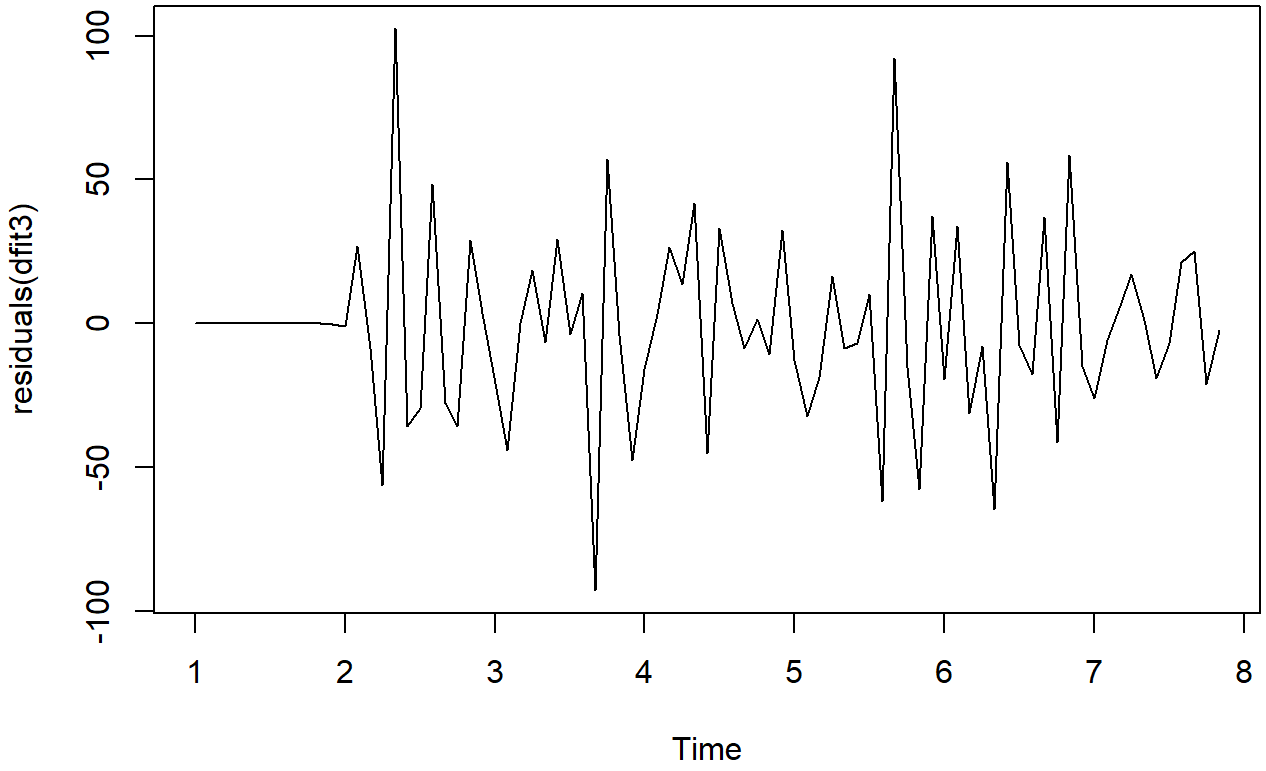

Partial ACF
0.6
0.4
0.2
0.0
-0.2
6
12
18
24
Lag

The seasonally differenced series shows a very strong pattern of positive autocorrelation and is similar to a seasonal random walk model. The correlation plots indicate an AR signature and/or incorporating another order of difference into the model.

Let's go ahead and apply **both seasonal and non-seasonal differencing** i,e ARIMA(0,1,0)(0,1,0)[12]

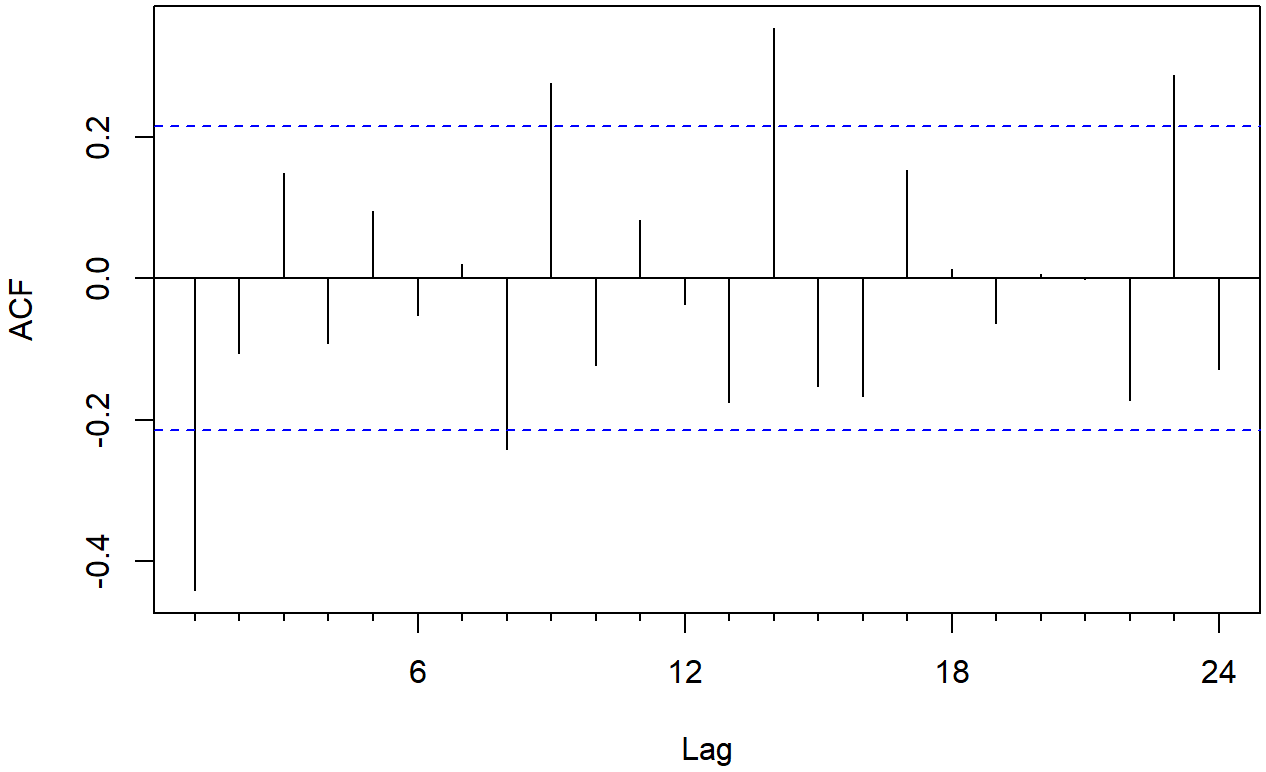
```
dfit3 <- arima(my_ts, order =c(0,1,0), seasonal = list(order = c(0,1,0), period = 12))
```

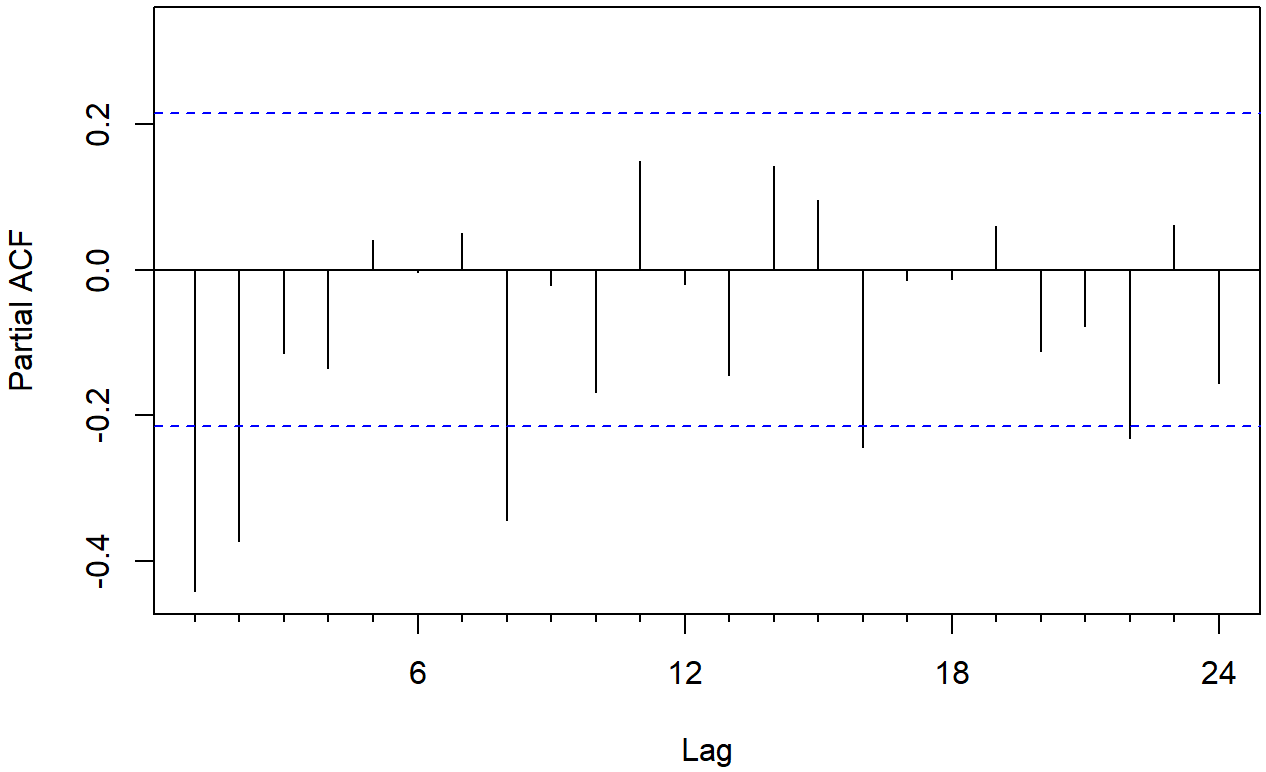
Next, we check the residuals:



Residuals seems to return to the mean and we don't see any pattern in the residuals.

Below is the acf and Pacf plot of residuals.


ACF
0.2
0.0
-0.2
-0.4
6
12
18
24
Lag


Partial ACF
0.2
0.0
-0.2
-0.4
6
12
18
24
Lag

ACF at lag 1 is -ve and slightly smaller than -0.4. We know that if the lag 1 acf falls below -0.5, then the series is over differenced. Positive spikes in acf have become negative, another sign of possible over differencing. Therefore, this model might be suffering from slight over differencing. This overdifferencing can be compensated by adding a MA term.

To select the appropriate order of differencing, we have to consider the error statistics, the standard deviation in specific.

In below summary, SD is same as RMSE.

```
##
## Call:
## arima(x = my_ts, order = c(0, 1, 0))
##
##
## sigma^2 estimated as 3893:  log likelihood = -455.3,  aic = 912.6
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 1.038055 62.01923 50.83083 -0.6970486 11.85129 0.9880124
##                      ACF1
## Training set -0.3993846
```

```
##
## Call:
## arima(x = my_ts, order = c(0, 0, 0), seasonal = list(order = c(0, 1, 0), period = 1
2))
##
##
## sigma^2 estimated as 2379:  log likelihood = -376.74,  aic = 755.48
##
## Training set error measures:
##                      ME     RMSE      MAE      MPE     MAPE     MASE
## Training set 7.060795 45.11405 34.54513 1.361696 8.090092 0.671463
##                     ACF1
## Training set 0.7200909
```

```
##
## Call:
## arima(x = my_ts, order = c(0, 1, 0), seasonal = list(order = c(0, 1, 0), period = 1
2))
##
##
## sigma^2 estimated as 1233:  log likelihood = -348.43,  aic = 698.86
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -1.20419 32.24894 22.79239 -0.4643199 5.265625 0.4430218
##                      ACF1
## Training set -0.4410923
```

Selecting appropriate order of differencing:

The optimal order of differencing is often the order of differencing at which the standard deviation is lowest. (Not always, though. Slightly too much or slightly too little differencing can also be corrected with AR or MA terms.

Out of the above, dfit3 model i.e ARIMA(0,1,0)(0,1,0)12 has the lowest standard deviation(RMSE) and AIC. Therefore, it is the correct order of differencing.

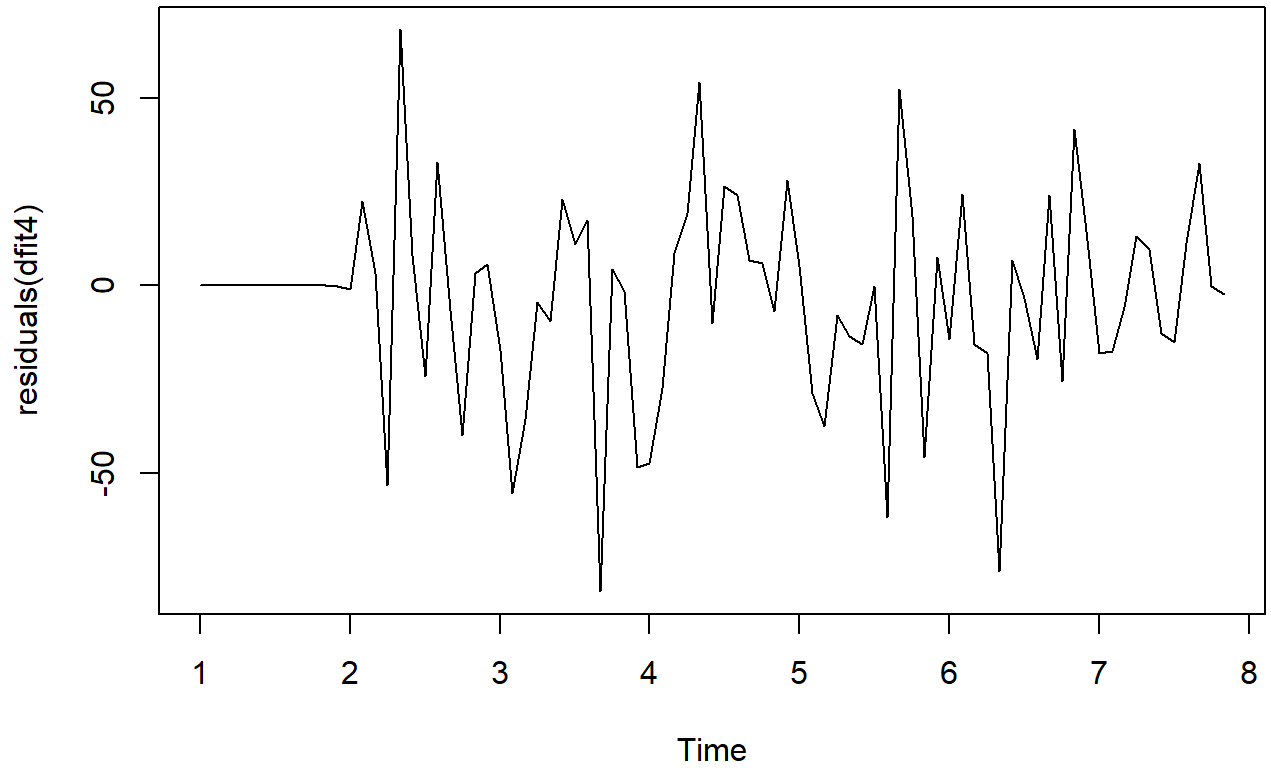
Therefore, the value of d=1 and D=1 is set. Now, we need to identify AR/MA and SAR/SMA values and fit the model.

Identifying the AR/MA(p/q) and SAR/SMA(P/Q) components.

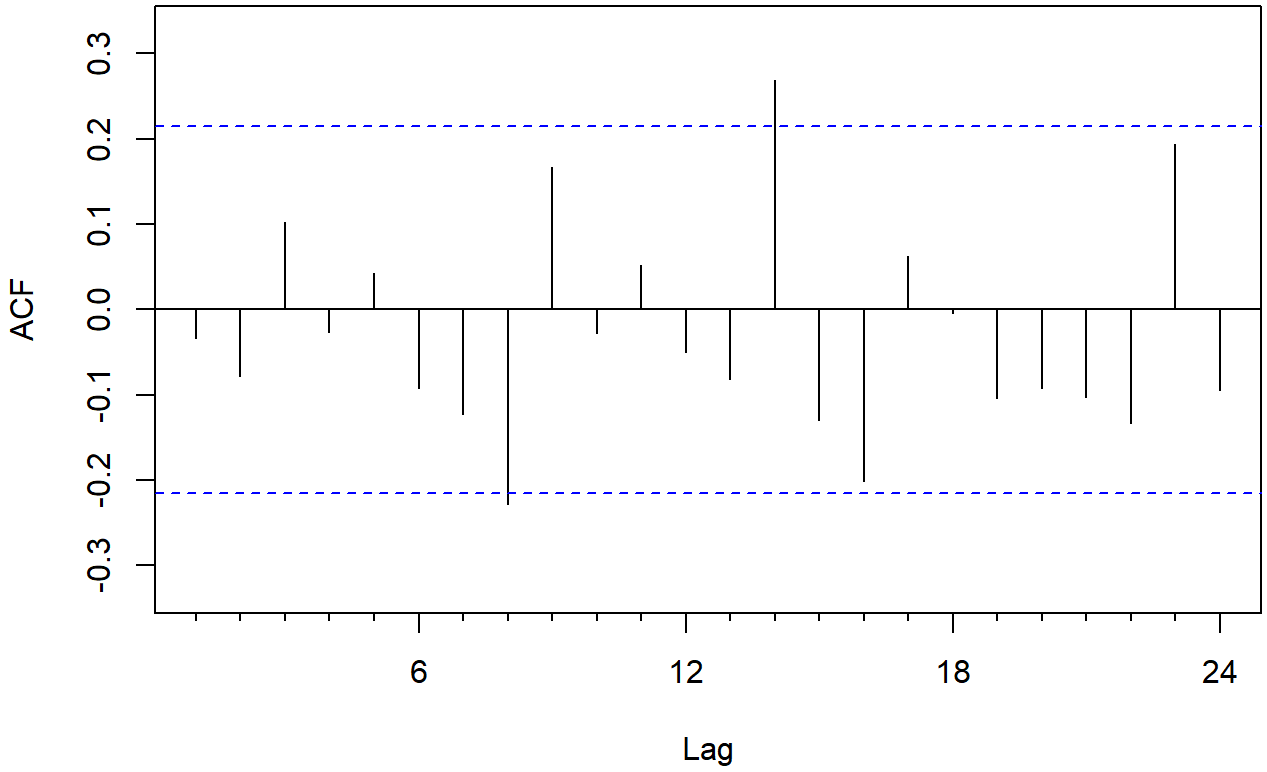
Looking back at the correlation plot of model dfit3, ACF is negative at lag 1 and shows sharp cut-off immediately after lag 1, we can add a MA to the model to compensate for the overdifferencing.

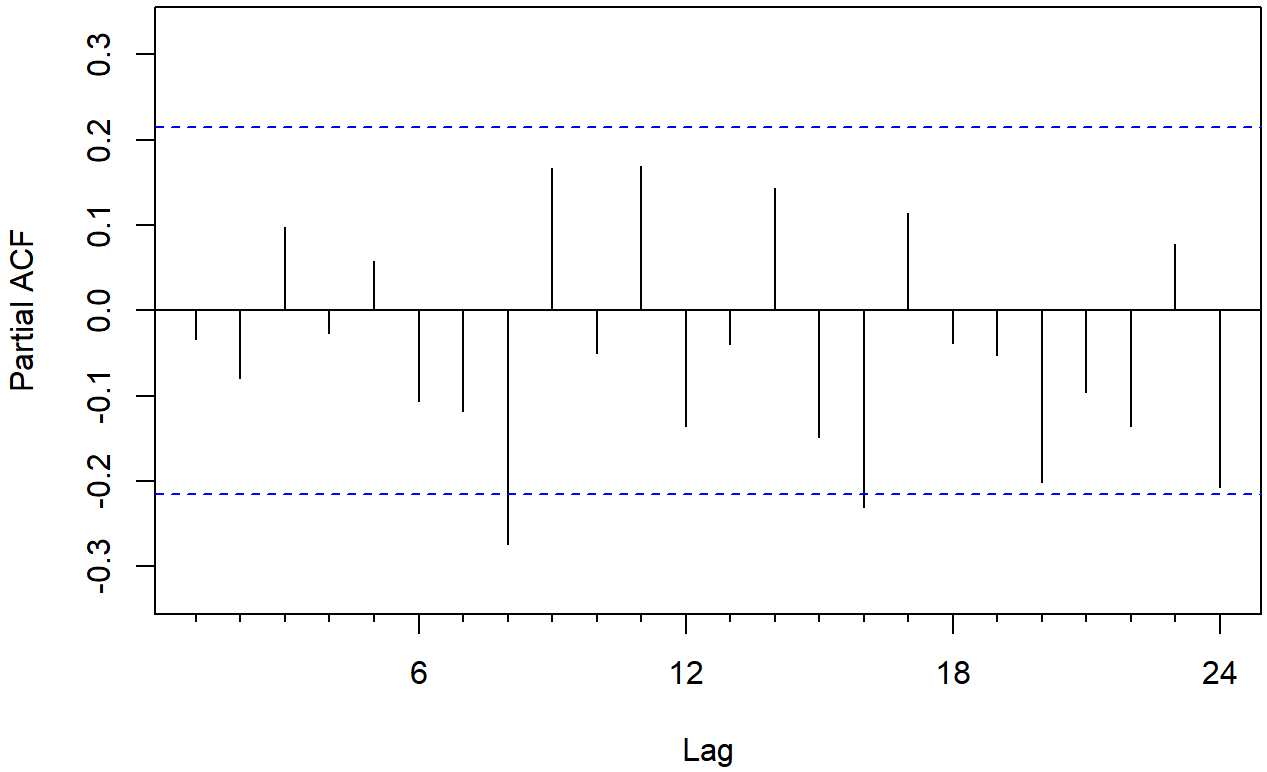
Since, we do not see any correlation at lag s,2s,3s etc i.e 12,24,36 etc, we do not need to add SAR/SMA to our model.

```
dfit4 <- arima(my_ts, order =c(0,1,1), seasonal = list(order = c(0,1,0), period = 12))
plot(residuals(dfit4))
```



In the above line, we added seasonal(and non-seasonal) difference along with a MA component that can compensate slight overdifferencing observed in acf plot of dfit3.


ACF
0.3
0.2
0.1
0.0
-0.1
-0.2
-0.3
6
12
18
24
Lag


Partial ACF
0.3
0.2
0.1
0.0
-0.1
-0.2
-0.3
6
12
18
24
Lag

```
##
## Call:
## arima(x = my_ts, order = c(0, 1, 1), seasonal = list(order = c(0, 1, 0), period = 1
2))
##
## Coefficients:
##           ma1
##       -0.6458
## s.e.   0.0906
##
## sigma^2 estimated as 856.1:  log likelihood = -335.93,  aic = 675.86
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -3.539959 26.87038 18.79392 -0.975538 4.307607 0.3653025
##                      ACF1
## Training set -0.03357659
```

Looking at he residual plot for above model, slight amount of correlation remains at lag 8, but the overall plots seem good.

Thus, this model seems like a good fit pending statistically significant MA co-efficient and low AIC.

Check for Statistical Significance

Let's check the model parameter's significance. The **coeftest()** function in *lmtest* package can help us in getting the p-values of coefficients.

```
coeftest(dfit4)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ma1 -0.645824   0.090609 -7.1276 1.021e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As we can see, P value is negligible and thus the test confirms that MA1 coefficient is statistically significant.

Check for Minimum AIC and Iterate

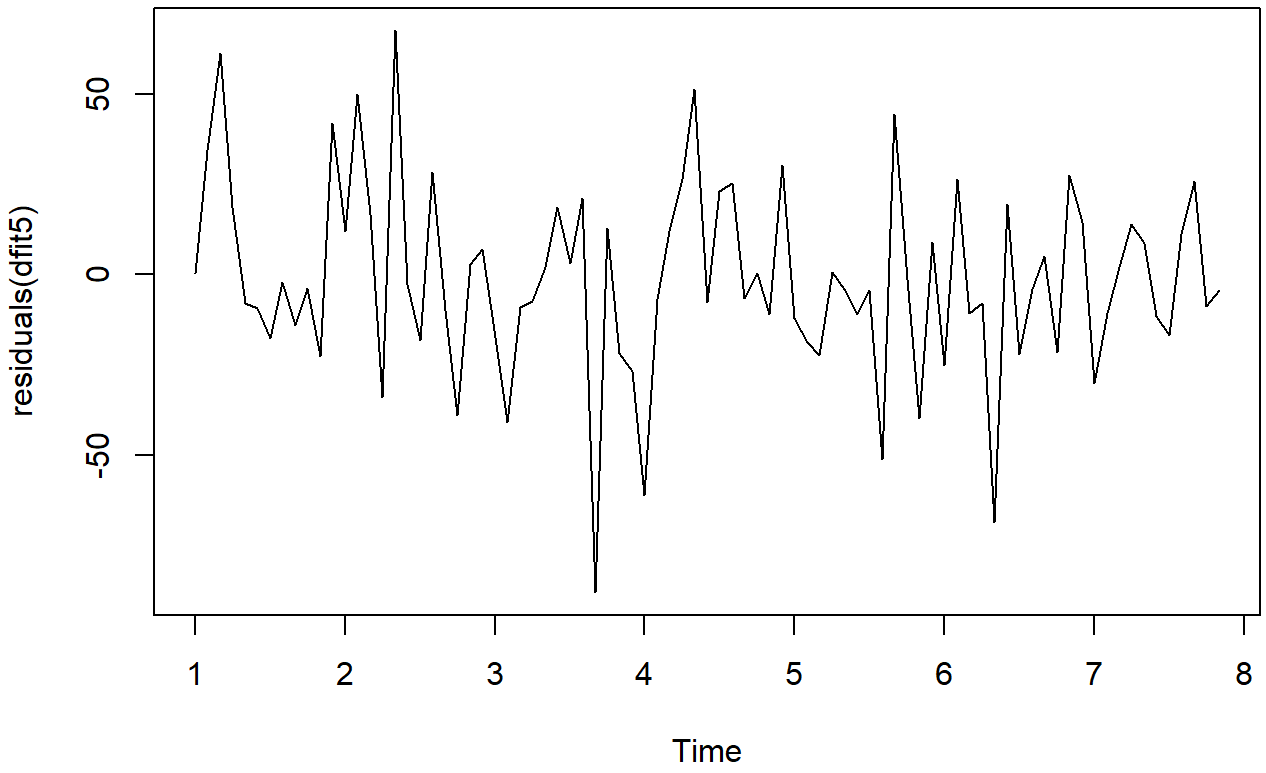
We can use the auto.arima() function to let R build a model with least AIC. This function searches through combinations of order parameters and picks the set that optimizes model fit criteria.

There exist a number of such criteria for comparing quality of fit across multiple models. Two of the most widely used are Akaike information criteria (AIC) and Baysian information criteria (BIC). These criteria are closely related and can be interpreted as an estimate of how much information would be lost if a given model is chosen. When comparing models, one wants to minimize AIC and BIC.

Next,we can compare that model to the one that we built above.

We run auto.Arima to find the model with lowest AIC and compare it to our model.

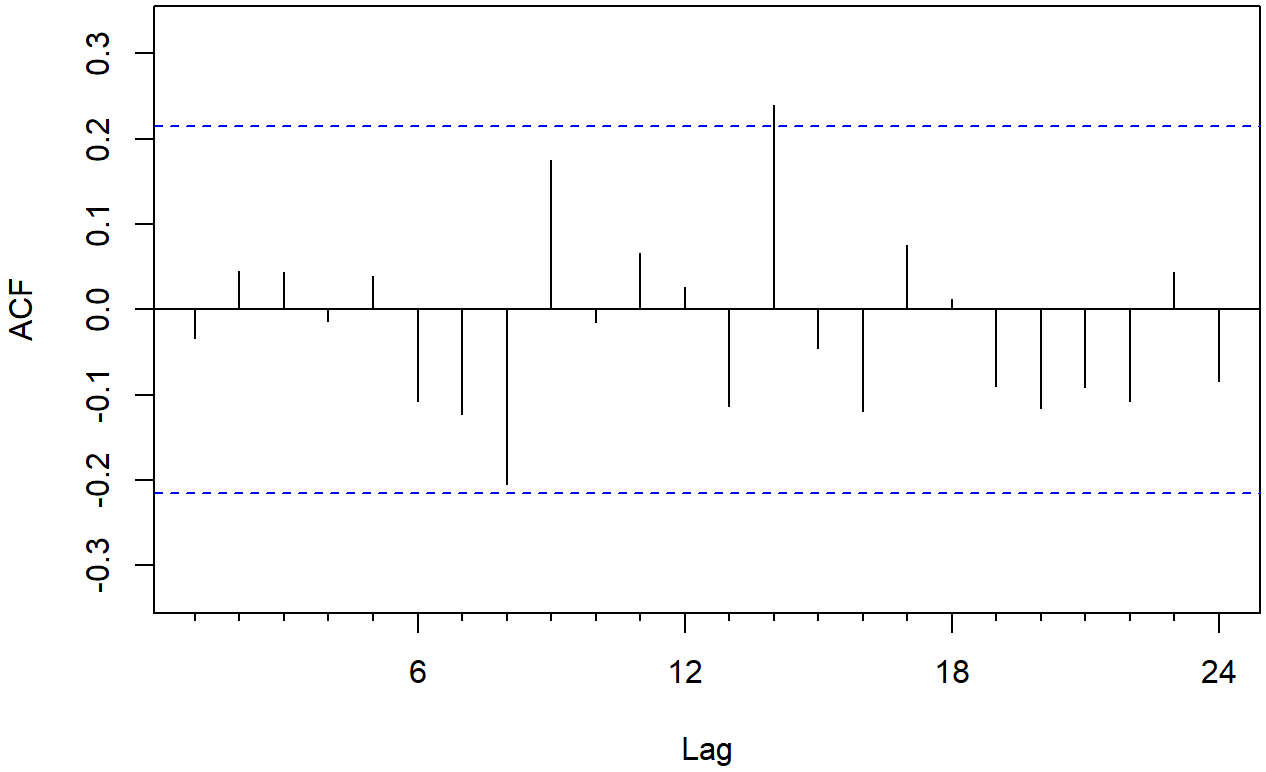
```
dfit5 <- auto.arima(my_ts, seasonal = TRUE)
plot(residuals(dfit5))
```

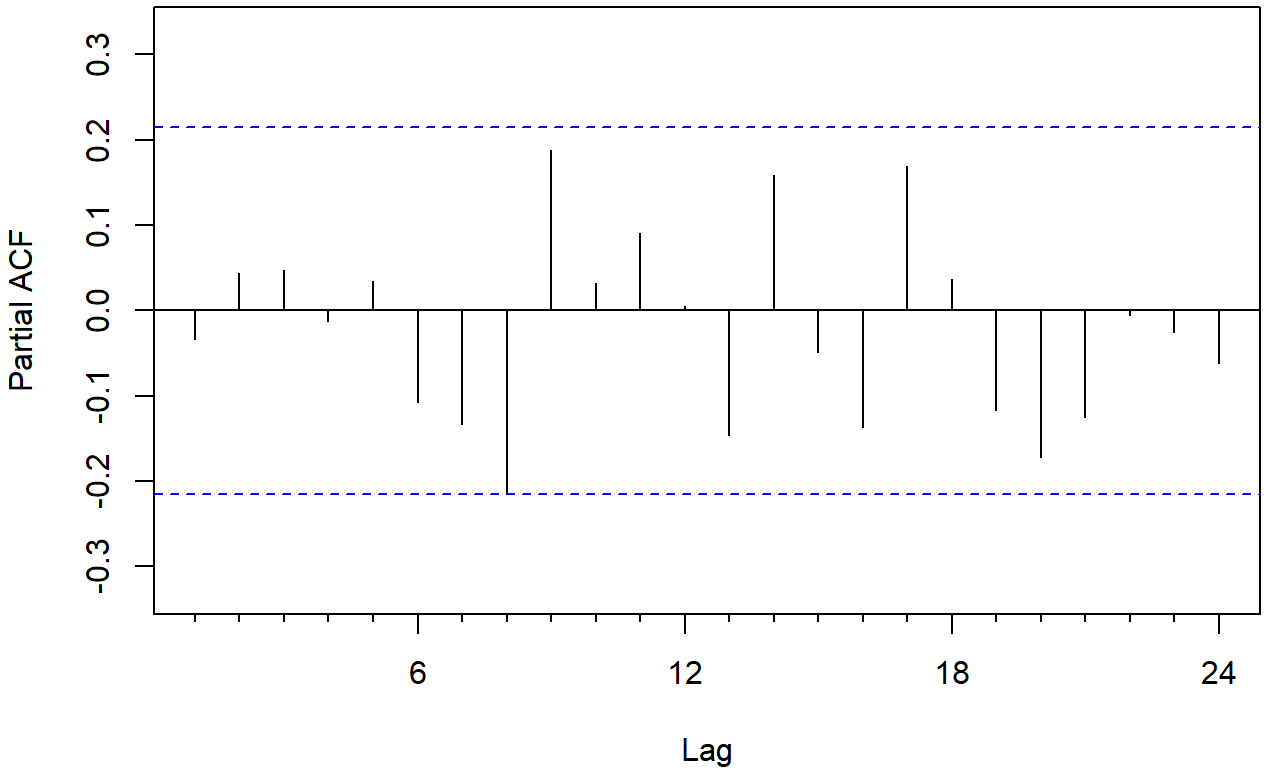


It is important that we mention **seasonal = TRUE** to let R know that this is a seasonal series. While auto.arima() can be very useful, it is still important to complete steps 1-5 in order to understand the series and interpret model results. Note that auto.arima() also allows the user to specify maximum order for (p, d, q), which is set to 5 by default

Residual plot is similar to that of the model we built above.

Next, we see the acf, pacf and summary of the auto built model.


ACF
0.3
0.2
0.1
0.0
-0.1
-0.2
-0.3
6
12
18
24
Lag


Partial ACF
0.3
0.2
0.1
0.0
-0.1
-0.2
-0.3
6
12
18
24
Lag

```
## Series: my_ts
## ARIMA(1,1,2)(1,0,0)[12]
##
## Coefficients:
##           ar1     ma1      ma2    sar1
##       -0.7814  0.2867  -0.6342  0.8635
## s.e.   0.0933  0.0975   0.0857  0.0453
##
## sigma^2 estimated as 766.5:  log likelihood=-394.73
## AIC=799.46   AICc=800.25   BIC=811.49
##
## Training set error measures:
##                      ME     RMSE     MAE        MPE     MAPE      MASE
## Training set -1.408176 26.83876 20.0973 -0.5884551 4.799585 0.4983789
##                      ACF1
## Training set -0.03399492
```

Let's check the auto model parameter's significance.

```
coeftest(dfit5)
```

```
##
## z test of coefficients:
##
##       Estimate Std. Error z value  Pr(>|z|)
## ar1  -0.781409   0.093336 -8.3720 < 2.2e-16 ***
## ma1   0.286682   0.097520  2.9397  0.003285 **
## ma2  -0.634218   0.085662 -7.4037 1.324e-13 ***
## sar1  0.863497   0.045258 19.0794 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Evaluation and Model Selection

Auto arima gives us ARIMA(1,1,2)(1,0,0)[12]. All coefficients are significant.

Clearly this model performs worse than the model we built earlier as ARIMA(0,1,1)(0,1,0)[12] has a AIC of 675.86 As opposed to a higher AIC of 799.46 generated by auto-arima. RMSE for both models are about the same.

By rule of parsimony and/or minimum AIC, we can reject ARIMA(1,1,2)(1,0,0)[12] and accept ARIMA(0,1,1)(0,1,0)[12] as our model.

Looking back at the auto generated model, we can infer that an additional AR component, which underdifferences the series, is being compensated by addition of a MA component. As AR and MA components tend to cancel each other's effect, it's always a good idea to try a model with one fewer AR term and one fewer MA term.

Model Validation

To see how our model will perform in future, we can use n-fold holdout method.

```
hold <- window(ts(my_ts), start = 72)
```

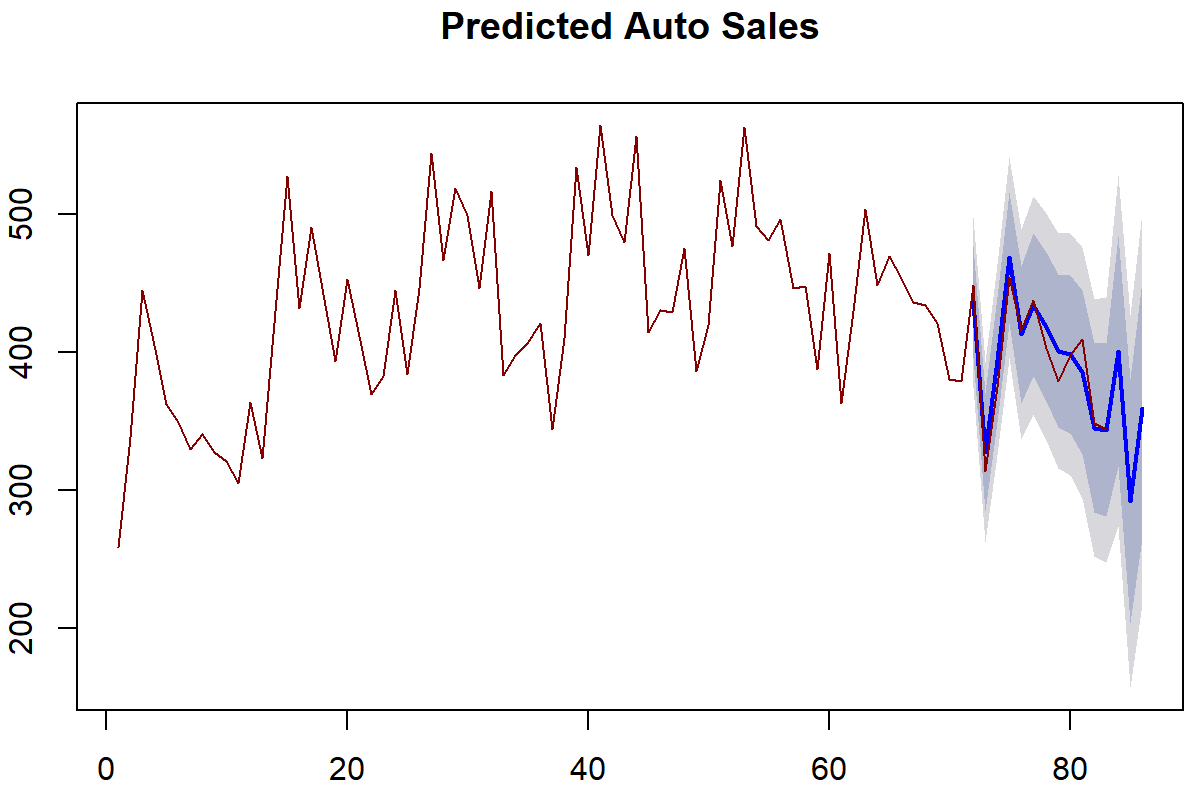
Fit the model to predict for observation(months) 72 through 83.

```
fit_predicted <- arima(ts(my_ts[-c(72:83)]), order =c(0,1,1), seasonal = list(order = c
(0,1,0), period = 12))
```

Use the above model to forecast values for last 10 months. Forecasting using a fitted model is straightforward in R. We can specify forecast horizon h periods ahead for predictions to be made, and use the fitted model to generate those predictions:

```
forecast_pred <- forecast(fit_predicted,h=15)
```

Next, we plot the predicted values and the validate against the actual data to see how our choosen model behaves in real world.



In the above graph, blue line is the predicted data and the confidence bands are in dark grey(80%) and light grey(95%).

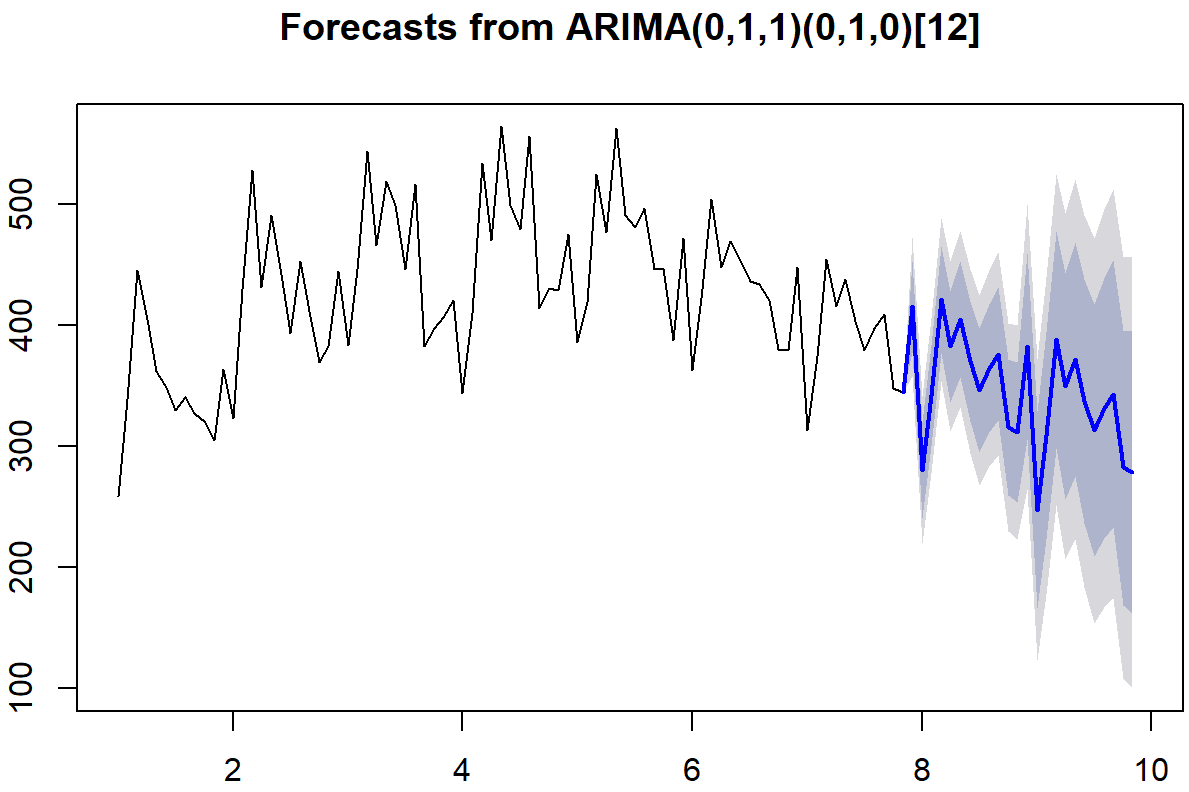
Model's prediction is pretty good and we can see predicted sales closely follow the actual data. This is an indication of a good model.

Sales Prediction for 2018/19.

Next step in our model is to forecast values i.e the monthly sales data. We have specified h=24 to predict for next 24 observations(months) i.e next 2 years - 2018 and 2019.

```
f_values <- forecast(dfit4, h=24)
```

Plot the predicted values.



Summary:

Confidence band for long term predictions keep on diverging. This is an indication that long term or distant futures predicted values are less certain. The reason for such uncertainty is that the model regresses future predictions on previously predicted values. Therefore, the Confidence bands increase in width.

Therefore, long-term forecasts using ARIMA/SARIMA should be avioded. For long-term prediction, it would be good to also draw on other sources of information during the model selection process and/or use an auxiliary model and take into account othe factors such as economic conditions, average income etc. We could try fitting time series models that allow for inclusion of other predictors using methods such ARMAX or dynamic regression. These more complex models allow for control of other factors in predicting the time series.