**Bike Sharing Service Analysis**

**Karan Khanna, Siddhant Sanjiv Sapte**

**Northeastern University, College of Professional Studies**

**Course: Intermediate Analytics (ALY6015)  
Instructor: Prof. Tenglong Li  
Date: 12/5/2018**

**Abstract**

Bike sharing service analysis is a project based on the data that is gathered form the bike service based out in Washington. Apart from interesting real-world applications of bike sharing systems, the characteristics of data being generated by these systems make them attractive for the

research. In this project, there were two objectives,  
1. The environmental factors which affect the bike hiring count per day the most.  
2. Forecasting the bike hiring count per day.

**Bike Sharing Service Analysis**

## Background

### Bike sharing systems are new generation of traditional bike rentals where whole process from membership, rental and return has become automatic. Through these systems, user can easily rent a bike from a position and return at another position. Currently, there are about over 500 bike-sharing programs around the world which is composed of over 500 thousand bicycles. Today, there exists great interest in these systems due to their important role in traffic, environmental and health issues.

## Dataset

Bike-sharing rental process is highly correlated to the environmental and seasonal settings. For instance, weather conditions, precipitation, day of week, season, hour of the day, etc. can affect the rental behaviors. Data was extracted on two hourly and daily basis and then the corresponding weather and seasonal information were added. The core data set is related to the two-year historical log corresponding to years 2011 and 2012 from Capital Bikeshare system, Washington D.C., USA which is publicly available in <https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset>.

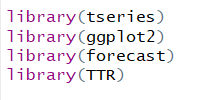
## Methods chosen

The objectives as described in the abstract, were to find out the factors on which the count of the bikes depend per day and the forecasting of the sales of the bike per day for the next year. We have used the linear model to predict the most significant environmental factors on which the bike hiring count depends and arima model to predict the forecast of the time series.

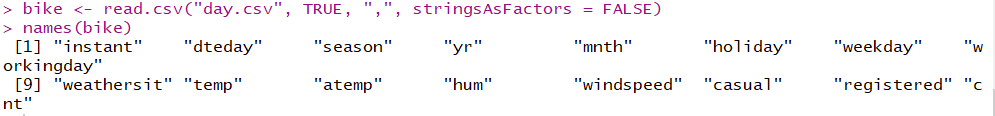
**Analysis**

The data was downloaded from the repository and was placed at the working directory.

The libraries that were used,



The data was loaded into R and the features of the dataset were,



The attribute information:

- instant: record index

- dteday: date

- season: season (1:spring, 2:summer, 3:fall, 4:winter)

- yr : year (0: 2011, 1:2012)

- mnth : month ( 1 to 12)

- hr : hour (0 to 23)

- holiday : weather day is holiday or not

- weekday : day of the week

- workingday: if day is neither weekend nor holiday is 1, otherwise is 0.

+ weathersit :

- 1: Clear, Few clouds, Partly cloudy, Partly cloudy

- 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

- 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds

- 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

- temp : Normalized temperature in Celsius.

- atemp: Normalized feeling temperature in Celsius.

- hum: Normalized humidity. The values are divided to 100 (max)

- windspeed: Normalized wind speed. The values are divided to 67 (max)

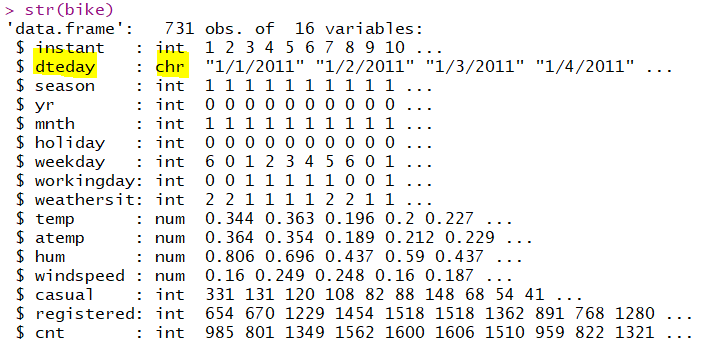
- casual: count of casual users

- registered: count of registered users

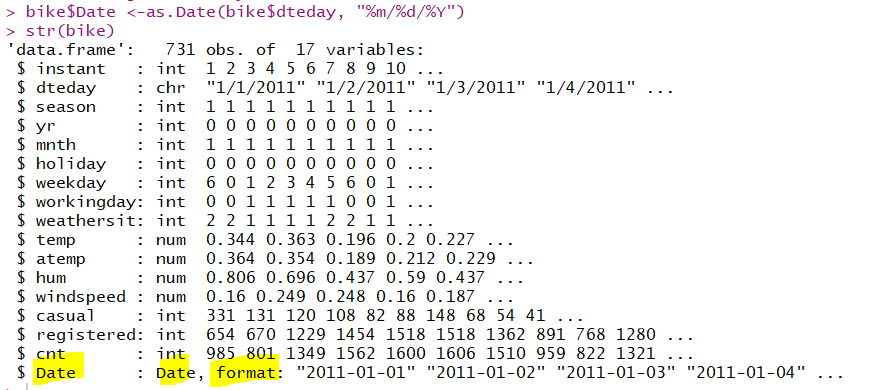
- cnt: count of total rental bikes including both casual and registered

**Data Conversions:**

The structures of the attributes in the data sets are very important, to see that we use the str() function.



In this the dteday variable is a character and we need to convert into date as we will be using the time series to forecast the predictions, the dteday variable need to be in day. Thus we convert that into date variable.

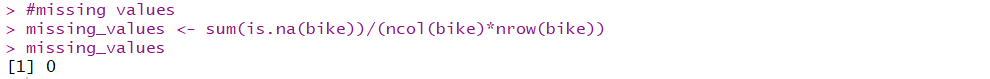


A new variable Date is added as a date variable.

**Data Cleaning**

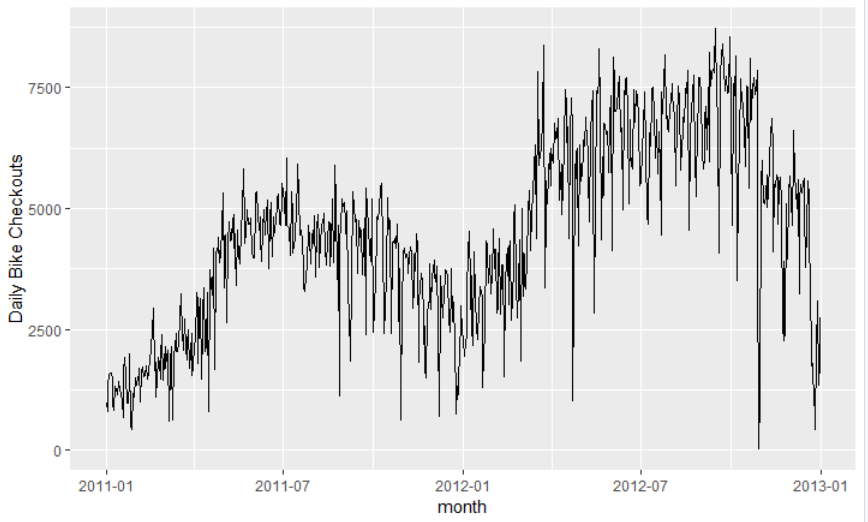
The next important step is to clean the data, to check for any missing values, for any outliers.

For checking the missing values,



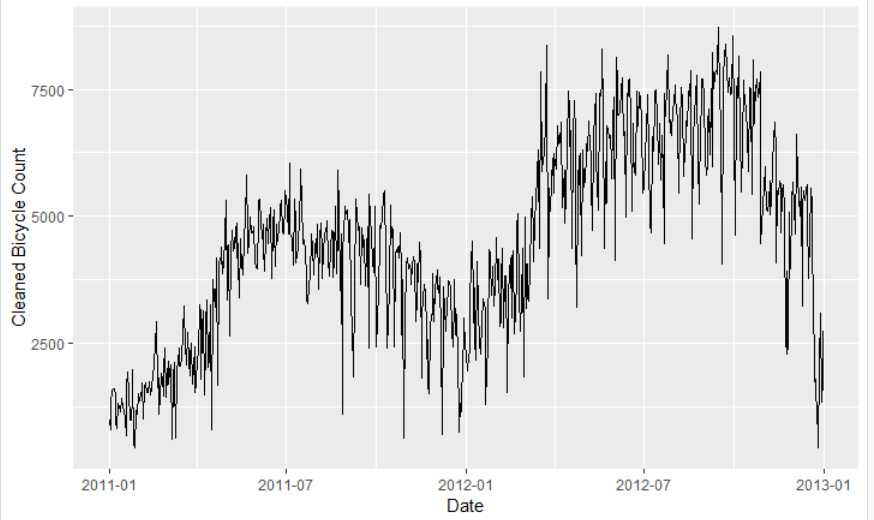
There are no missing values in the entire dataset.

For any outliers in the data, first we need to plot the graph of the data,



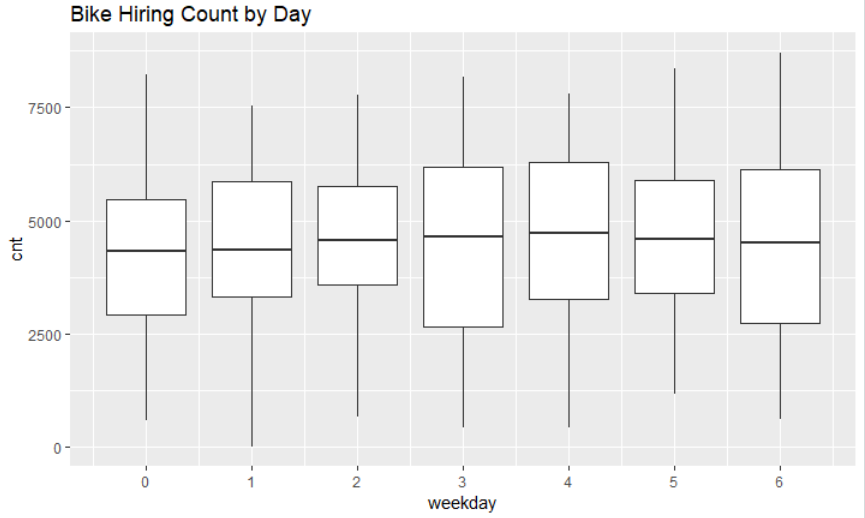
In this case, bicycle checkouts are showing a lot of fluctuations from one day to another. However, even with this volatility present, we already see some patterns emerge. For example, lower usage of bicycles occurs in the winter months and higher checkout numbers are observed in the summer months. In some cases, the number of bicycles checked out dropped below 100 on day and rose to over 4,000 the next day. These are suspected outliers that could bias the model by skewing statistical summaries. The outliers will be cleaned with the help of the tsclean(), this function identifies the outliers and replaces the outliers using series smoothing and decomposition.

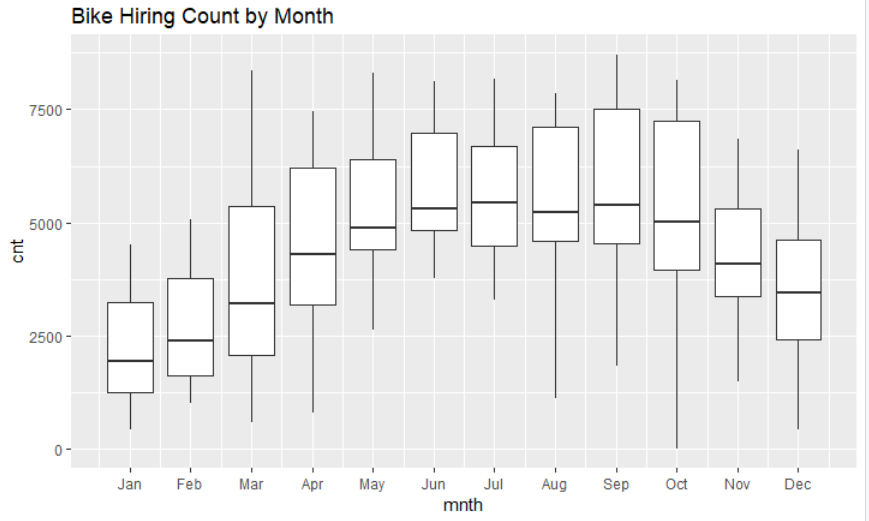




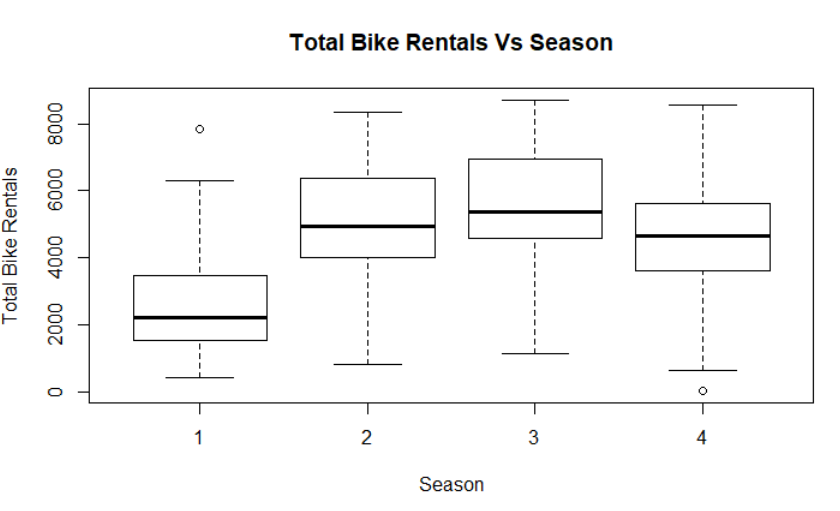
**Data Visualization**

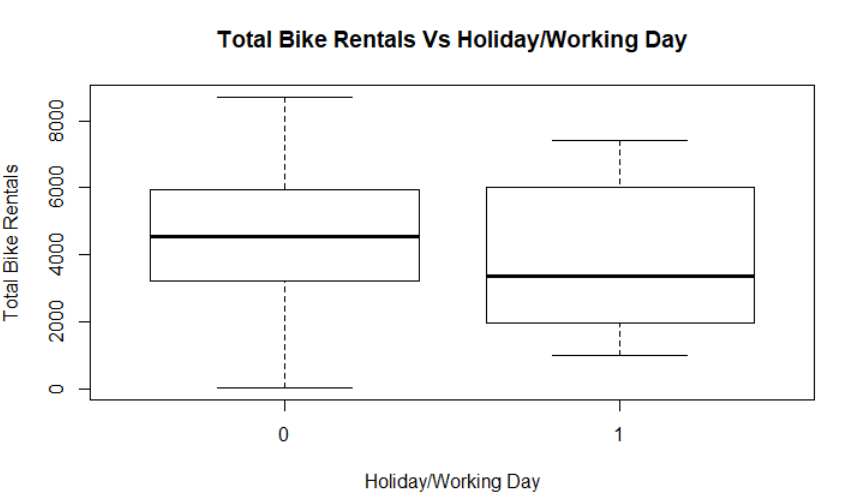
One of the most important steps is to visualize the data. The box plots gives quite a idea about the data. Here are the box plots that gives a clear picture regarding the data,

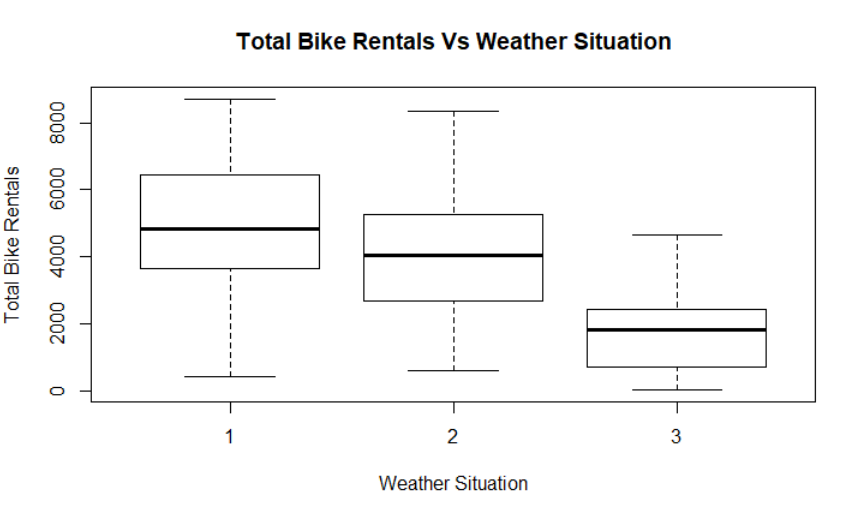


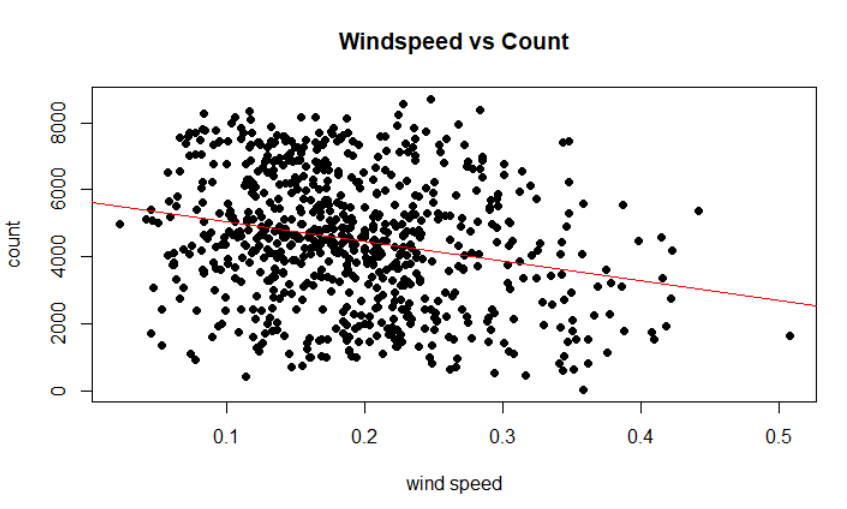


From the above box plots, the first one was the bike hiring by day and the next one is the bike hiring count by month. From both the graphs we can see that data is non-stationary and there is a trend in data. For using the time series, we need the data to be stationary; this is one thing we got to know after visualizing the data. Now, we will see the effect of the other features on the count,







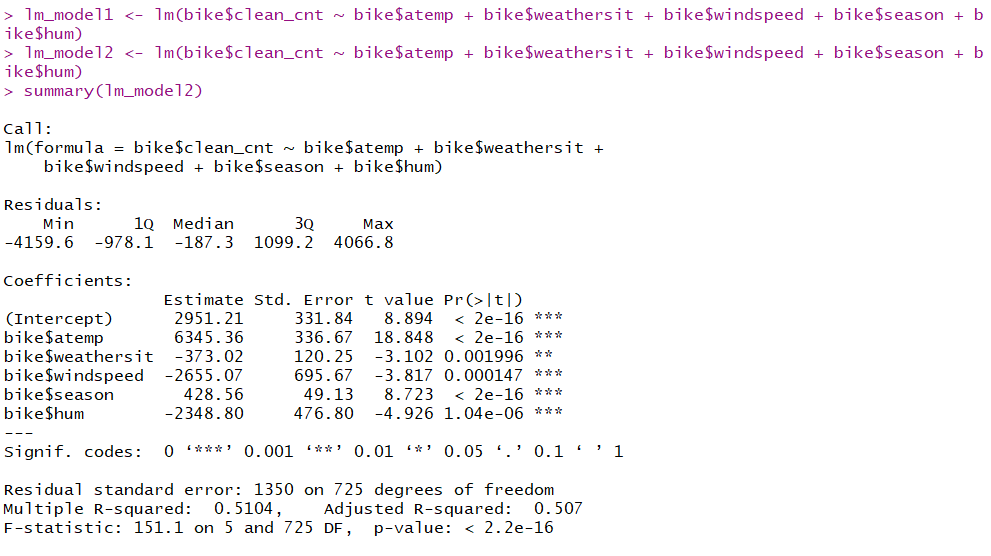


The boxplots show how the effect of seasons, holiday/weekday, weather on the bike hiring count. The bike hiring count is maximum in the summer and fall, weekends and holidays and in the clear and partly cloudy weather and for else it goes on decreasing. From the windspeed graph we can see that as the windspeed increase the count also decreases.

This all information about the features of the data give an idea about the significant factors that might affect the bike hiring count.

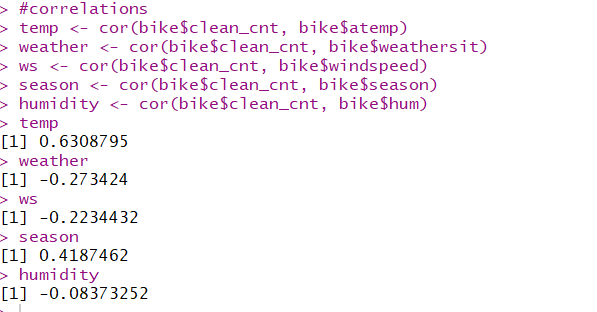
**Linear Model Analysis**

We have used the linear model to find out the most significant environmental factors on which the bike hiring count depends. After considering all the environmental factors in the data set, the final model shows all the significant environmental factors on which the bike hiring count depends.



lm\_model2 shows all the significant environmental factors which affect the bike hiring count; they are, atemp, weather, windspeed, season, humidity.

To find the most significant factor, we have calculated the correlations of each of the factors with the count.



The correlation for the temp which is the feeling temperature is the maximum, thus depending upon the feeling temperature the hiring count surely changes.

**Time Series Analysis**

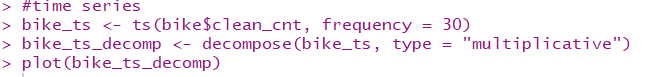
We have used the time series analysis to predict the forecast of the bike hiring count for the next coming year. The building blocks of a time series analysis are seasonality, trend, and cycle. These intuitive components capture the historical patterns in the series. Not every series will have all three (or any) of these components, but if they are present, deconstructing the series can help you understand its behavior and prepare a foundation for building a forecasting model.

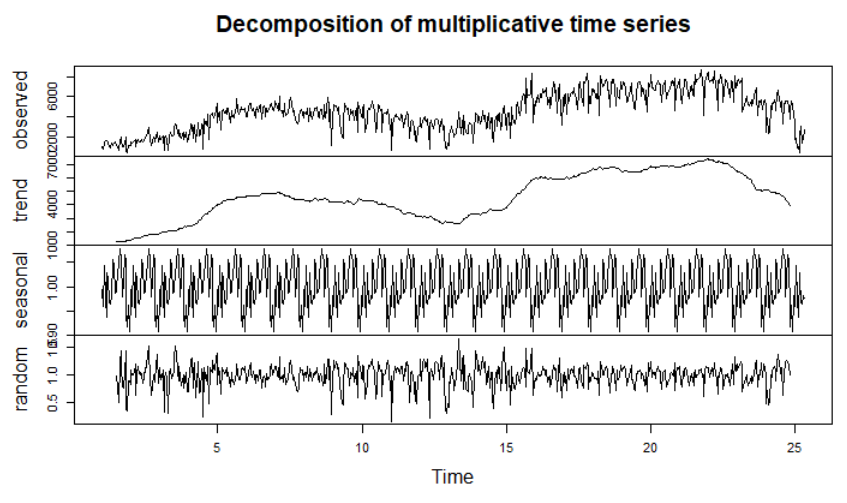
Seasonal component refers to fluctuations in the data related to calendar cycles. Trend component is the overall pattern of the series. Cycle component consists of decreasing or increasing patterns that are not seasonal. Usually, trend and cycle components are grouped together. Trend-cycle component is estimated using moving averages. Finally, part of the series that can't be attributed to seasonal, cycle, or trend components is referred to as residual or error.

Formally, if Y is the number of bikes rented, we can decompose the series in two ways: by using either an additive or multiplicative model,

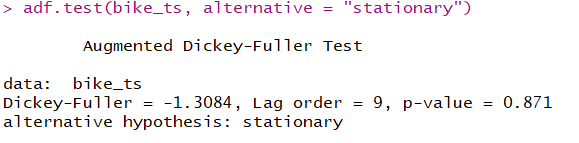
Y = S \* T \* E

where S is the seasonal component, T is trend and cycle, and E is the remaining error. An additive model is usually more appropriate when the seasonal or trend component is not proportional to the level of the series, as we can just overlay (i.e. add) components together to reconstruct the series. From the data and the plot that we have plot earlier to see the outliers, we can clearly see that the amplitude is increasing every year, thus, it becomes an multiplicative time series.





The data is clearly not stationary. Fitting an ARIMA model requires the series to be stationary. A series is said to be stationary when its mean, variance, and autocovariance are time invariant. The Augmented Dickey-Fuller (ADF) test is a formal statistical test for stationarity. The null hypothesis assumes that the series is non-stationary.

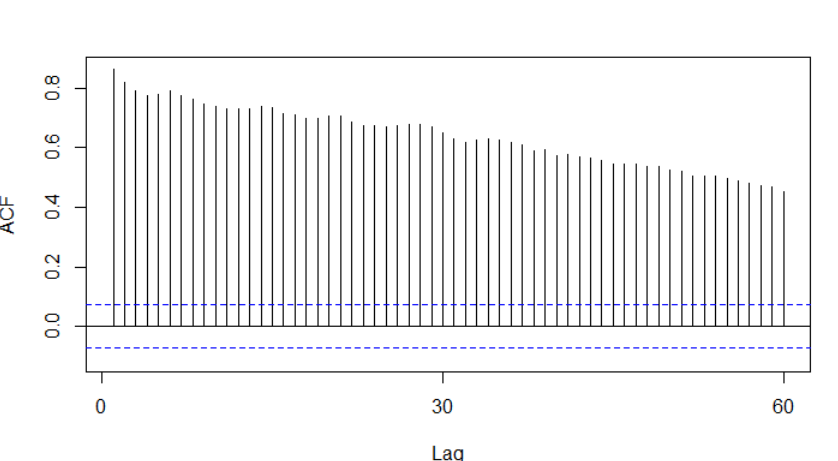


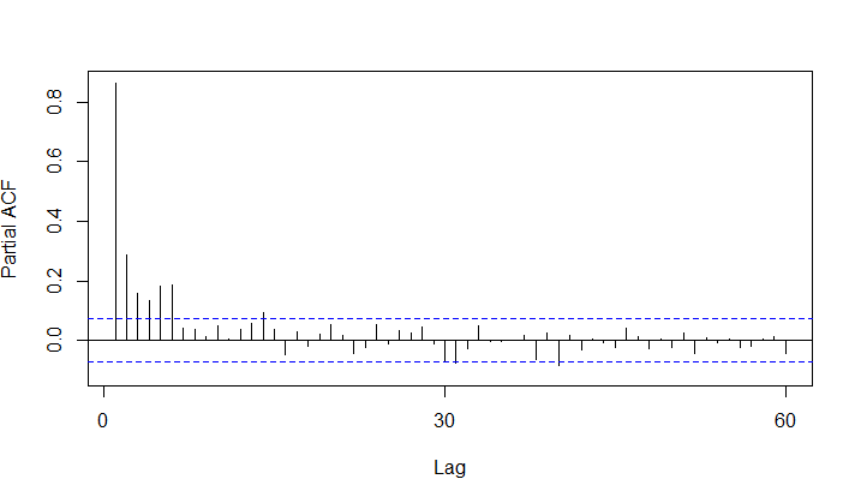
The p-value is not less than 0.05, which confirms that the data is non-stationary. Usually, non-stationary series can be corrected by a simple transformation such as differencing. Differencing the series can help in removing its trend or cycles. The idea behind differencing is that, if the original data series does not have constant properties over time, then the change from one period to another might. The number of differences performed is represented by the d component of ARIMA. Now, we move on to diagnostics that can help determine the order of differencing.

**Autocorrelations and Choosing Model Order**

Autocorrelation plots are a useful visual tool in determining whether a series is stationary. These plots can also help to choose the order parameters for ARIMA model. If the series is correlated with its lags then, generally, there are some trend or seasonal components and therefore its statistical properties are not constant over time. ACF plots display correlation between a series and its lags. In addition to suggesting the order of differencing, ACF plots can help in determining the order of the M A (q) model. Partial autocorrelation plots (PACF), as the name suggests, display correlation between a variable and its lags that is not explained by previous lags. PACF plots are useful when determining the order of the AR(p) model.







R plots 95% significance boundaries as blue dotted lines. There are significant autocorrelations with many lags in our bike series, as shown by the ACF plot below. However, this could be due to carry-over correlation from the first or early lags, since the PACF plot only shows a spike at lags 1 and 7. We can start with the order of d = 1 and re-evaluate whether further differencing is needed.

> count\_d <- diff(bike$clean\_cnt, differences = 1)

The augmented Dickey-Fuller test on differenced data rejects the null hypotheses of non-stationarity. Plotting the differenced series, we see an oscillating pattern around 0 with no visible strong trend. This suggests that differencing of order 1 terms is sufficient and should be included in the model.

> adf.test(count\_d1, alternative = "stationary")

Augmented Dickey-Fuller Test

data: count\_d1

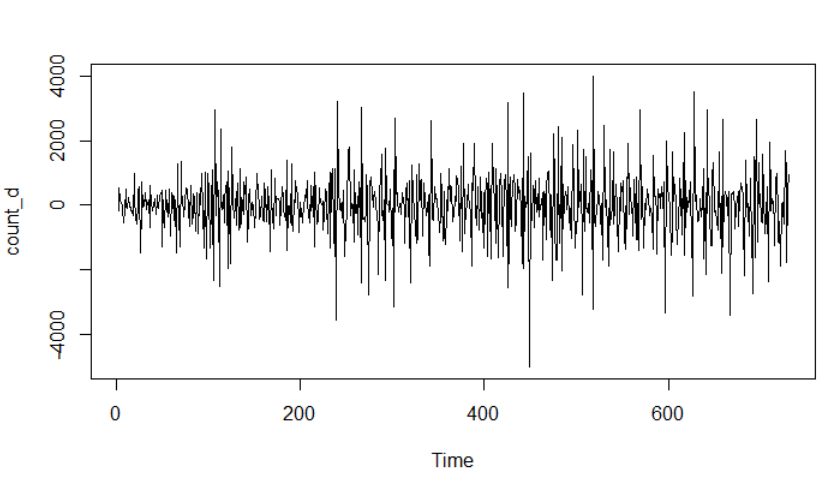
Dickey-Fuller = -13.499, Lag order = 8, p-value = 0.01

alternative hypothesis: stationary

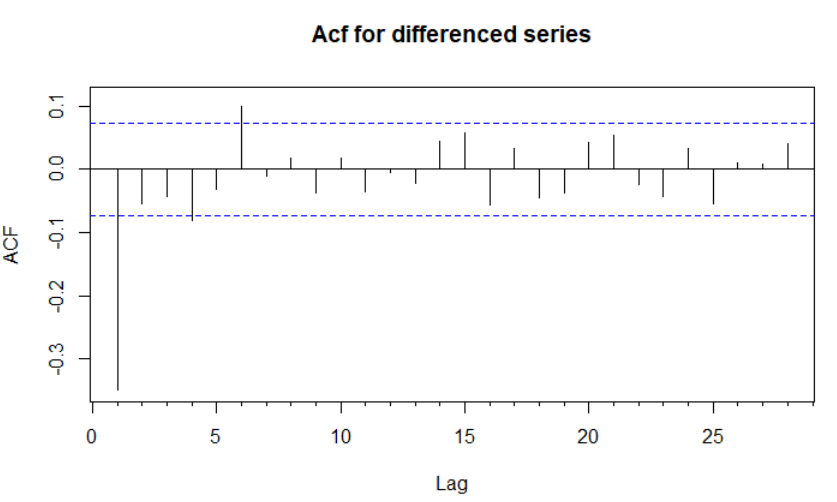
Warning message:

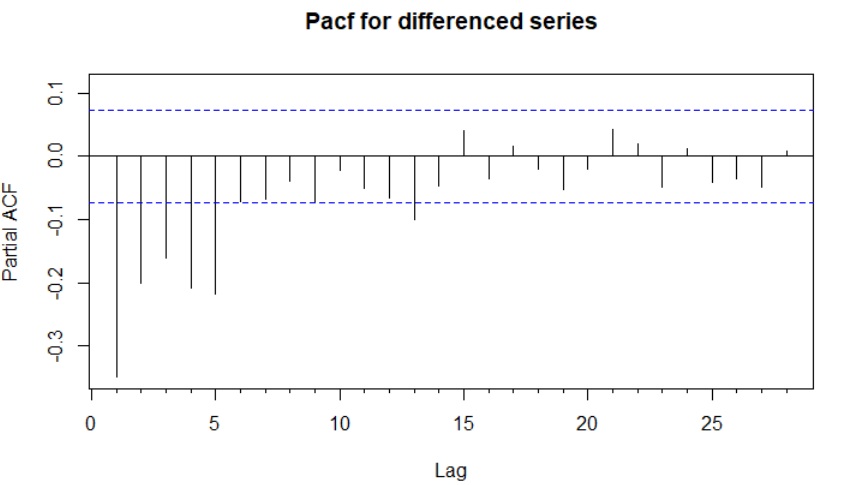
In adf.test(count\_d1, alternative = "stationary") :

p-value smaller than printed p-value



The acf and the pacf plots from the differenced count is,





The are significant lags at 1 for acf and 7 for the pacf. This suggests that we might want to test models with AR and MA components of order 1 and 7.

**Fitting an ARIMA model**

Now let's fit a model. The forecast package allows the user to explicitly specify the order of the model using the arima() function, or automatically generate a set of optimal (p, d, q) using auto.arima(). We will be first using the auto.arima() to test the model,

> fit\_arima <- auto.arima(bike$clean\_cnt, seasonal = F)

> fit\_arima

Series: bike$clean\_cnt

ARIMA(1,1,1)

Coefficients:

ar1 ma1

0.3049 -0.8654

s.e. 0.0449 0.0227

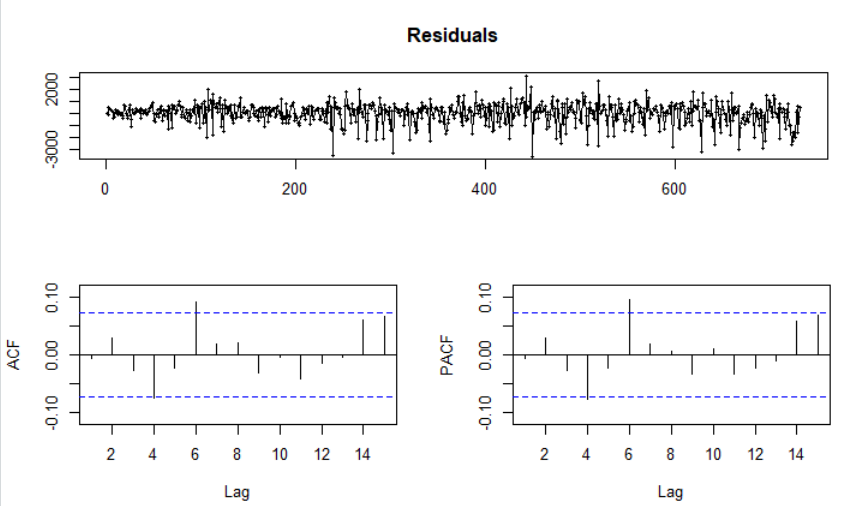
sigma^2 estimated as 734555: log likelihood=-5965.32

AIC=11936.64 AICc=11936.67 BIC=11950.42

The equation for this model would be,

Y\_t = 0.3049\*y\_t-1 – 0.8654\*e\_t-1 +E  
where E is some error and the original series is differenced with order 1. When we see the residuals,

> tsdisplay(residuals(fit\_arima),lag.max=15, main='Residuals')



From the residuals it is clear that the pattern present in ACF/PACF and model residuals plots repeating at lag 6. This suggests that our model may be better off with a different specification, such as p = 6 or q = 6. We will be using the p and the q values we got from the acf and the pacf graphs. We will be fitting another model with p and q as 1 and 7 respectively.

> fit1 <- arima(bike$clean\_cnt, order = c(1,1,7))

> fit1

Call:

arima(x = bike$clean\_cnt, order = c(1, 1, 7))

Coefficients:

ar1 ma1 ma2 ma3 ma4 ma5 ma6 ma7

0.3892 -0.9574 0.0959 -0.0388 -0.0623 0.0582 0.1110 -0.0835

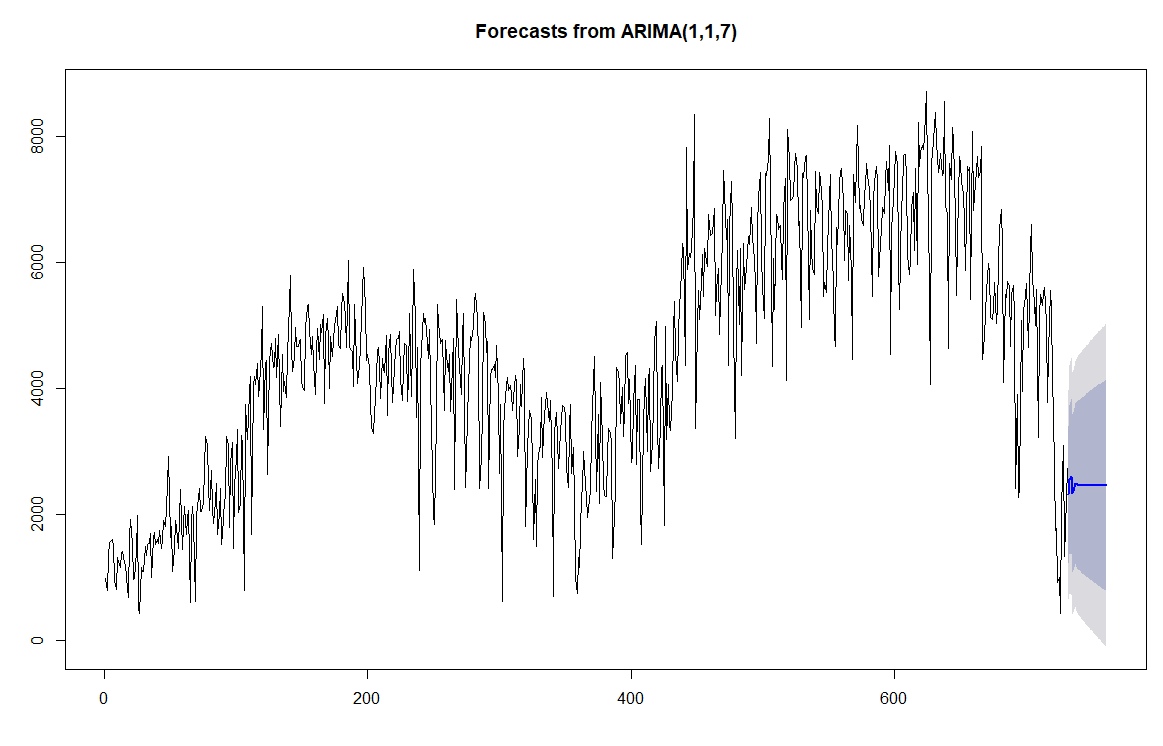
s.e. 0.3776 0.3768 0.2204 0.0678 0.0625 0.0607 0.0501 0.0451

sigma^2 estimated as 719483: log likelihood = -5958.79, aic = 11935.59

After the forecast with the above model,

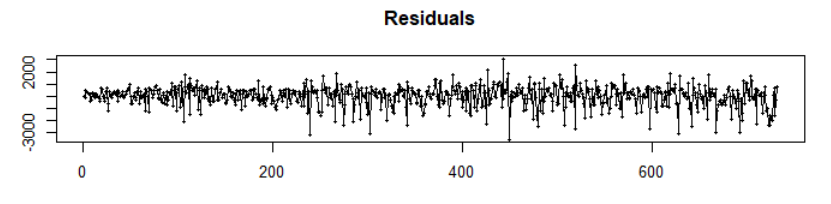
> fcast <- forecast(fit1, h = 30)

> plot(fcast)



From the forecast we can see that the there is a slight increase in the graph and then there is a decrease, it follows the same trend as the previous years. After plotting the residuals,

> tsdisplay(residuals(fit1),lag.max=15, main='Residuals')



The residuals are the white noise or the error in the model. This model can be further improved by various other techniques and using different time series model.

**Future Scope**

After an initial naive model is built, it's natural to wonder how to improve on it. Other forecasting techniques, such as exponential smoothing, would help make the model more accurate using a weighted combination of seasonality, trend, and historical values to make predictions. Using methods such ARMAX or dynamic regression; these more complex models allow for control of other factors in predicting the time series.

**Conclusion**

We have successfully accomplished the objectives of finding the most significant factors which affect the bike hiring count and we have forecasted the bike hiring count and its following the trend like the past two years.

**References**

* Sapegina, S. (2018, March 29). Retrieved from <http://rstudio-pubs-static.s3.amazonaws.com/374830_ab4ac8951c94411aa9abfe5f561cef91.html>
* Srivastava, T. (2018, September 20). A Complete Tutorial on Time Series Modeling in R. Retrieved from <https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/>
* Martin, A. (n.d.). Data analysis and visualization in R - Final Paper: Bike Sharing Dataset Analysis. Retrieved from <http://rstudio-pubs-static.s3.amazonaws.com/158595_1f520fd8d8e34a5ab3a127376f2f6169.html>
* TDP4you. (n.d.). TDP4you/Bike-Sharing-Dataset-Analysis. Retrieved from <https://github.com/TDP4you/Bike-Sharing-Dataset-Analysis/blob/585e890ecb8f97f46ac53da11500065615d2c148/README.md>

**Appendix**

library(tseries)

library(ggplot2)

library(forecast)

library(TTR)

bike <- read.csv("day.csv", TRUE, ",", stringsAsFactors = FALSE)

names(bike)

View(bike)

#missing values

missing\_values <- sum(is.na(bike))/(ncol(bike)\*nrow(bike))

missing\_values

str(bike)

#converting the dteday char into date

bike$Date <-as.Date(bike$dteday, "%m/%d/%Y")

#visualizing the data

dev.off()

ggplot(bike, aes(Date, cnt)) + geom\_line() + scale\_x\_date('month') + ylab("Daily Bike Checkouts") + xlab("")

#cleaning of data

count\_ts = ts(bike[, c('cnt')])

bike$clean\_cnt = tsclean(count\_ts)

ggplot() + geom\_line(data = bike, aes(x = Date, y = clean\_cnt)) + ylab('Cleaned Bicycle Count')

#box plots

ggplot(bike, aes(x = weekday,y=cnt)) + geom\_boxplot(aes(group=weekday))+ scale\_x\_continuous(breaks=seq(0,7,1))+ ggtitle("Bike Hiring Count by Day")

ggplot(bike, aes(y=cnt, x = mnth)) + geom\_boxplot(aes(group=mnth)) + scale\_x\_continuous(breaks=seq(0,12,1), labels=c("","Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec")) + ggtitle("Bike Hiring Count by Month")

boxplot(bike$cnt ~ bike$season,

data = bike,

main = "Total Bike Rentals Vs Season",

xlab = "Season",ylab = "Total Bike Rentals")

boxplot(bike$cnt ~ bike$holiday,

data = bike,

main = "Total Bike Rentals Vs Holiday/Working Day",

xlab = "Holiday/Working Day",

ylab = "Total Bike Rentals")

boxplot(bike$cnt ~ bike$weathersit,

data = bike,

main = "Total Bike Rentals Vs Weather Situation",

xlab = "Weather Situation",

ylab = "Total Bike Rentals")

#windspeed

plot( bike$windspeed,bike$cnt,xlab="wind speed", ylab="count",main ="Windspeed vs Count" ,pch =19)

abline(lm(bike$cnt~bike$windspeed), col="red")

##############################################

#lm-model

lm\_model1 <- lm(bike$clean\_cnt ~ bike$atemp + bike$weathersit + bike$windspeed + bike$season + bike$hum +bike$temp)

summary(lm\_model1)

lm\_model2 <- lm(bike$clean\_cnt ~ bike$atemp + bike$weathersit + bike$windspeed + bike$season + bike$hum)

summary(lm\_model2)

#correlations

temp <- cor(bike$clean\_cnt, bike$atemp)

weather <- cor(bike$clean\_cnt, bike$weathersit)

ws <- cor(bike$clean\_cnt, bike$windspeed)

season <- cor(bike$clean\_cnt, bike$season)

humidity <- cor(bike$clean\_cnt, bike$hum)

temp

weather

ws

season

humidity

##############################################

#time series

bike\_ts <- ts(bike$clean\_cnt, frequency = 30)

bike\_ts\_decomp <- decompose(bike\_ts, type = "multiplicative")

plot(bike\_ts\_decomp)

#stationarity test

adf.test(bike\_ts, alternative = "stationary")

#deseasonal

acf\_bike <- Acf(bike\_ts, main="")

pacf\_bike <- Pacf(bike\_ts, main="")

count\_d <- diff(bike$clean\_cnt, differences = 1)

plot(count\_d)

adf.test(count\_d1, alternative = "stationary")

acf\_bike\_d1 <- Acf(count\_d1, main = "Acf for differenced series")

pacf\_bike\_d1 <- Pacf(count\_d1, main = "Pacf for differenced series")

#arima

fit\_arima <- auto.arima(bike$clean\_cnt, seasonal = F)

fit\_arima

tsdisplay(residuals(fit\_arima),lag.max=15, main='Residuals')

fcast\_arima <- forecast(fit\_arima, h=30)

plot(fcast\_arima)

fit1 <- arima(bike$clean\_cnt, order = c(1,1,7))

fit1

fcast <- forecast(fit1, h = 30)

plot(fcast)

tsdisplay(residuals(fit1),lag.max=15, main='Residuals')

**Contributions**

The entire project was done by both of us, Siddhant and Karan, hand in hand. The regression and time series part were done by both by us with proper input from both of us.