UFR
DE MATHÉMATIQUES
ET INFORMATIQUE

UNIVERSITÉ
PARIS
DESCARTES

Time Series Analysis Project

EL BAHAOUI OUSSAMA

Through this R Markdown Notebook, I'm answering the questions proposed by Prof Taoufik Ennajary for the Time Series Analysis Project. The goal of the project is to learn how to : plot, examine, and prepare series for modeling and forecasting via a process of evaluation and iteration. For this project, we used a dataset containing the hourly and daily count of rental bikes between years 2011 and 2012 in Capital bikeshare system in Washington, DC with the corresponding weather and seasonal information. dataset link.

0. Importing libraries

```
library(lubridate)

##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date

library(tseries)
library(forecast)
library("TTR")
```

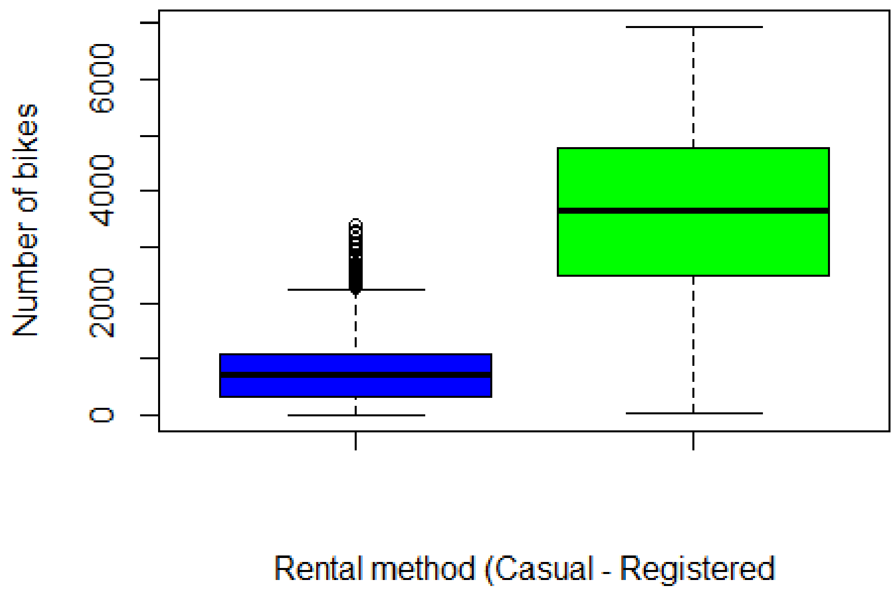
1. Examine your data

First, we download the zip folder contained in https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset and extract its files. For this project, we're mainly interested in the *day.csv* file. Then, we're reading the day data and store it into the **day.data** variable using the read.csv() function.

```
day.data <- read.csv(file = "data/day.csv")
```

Now that the data is loaded, we need to examine its content.

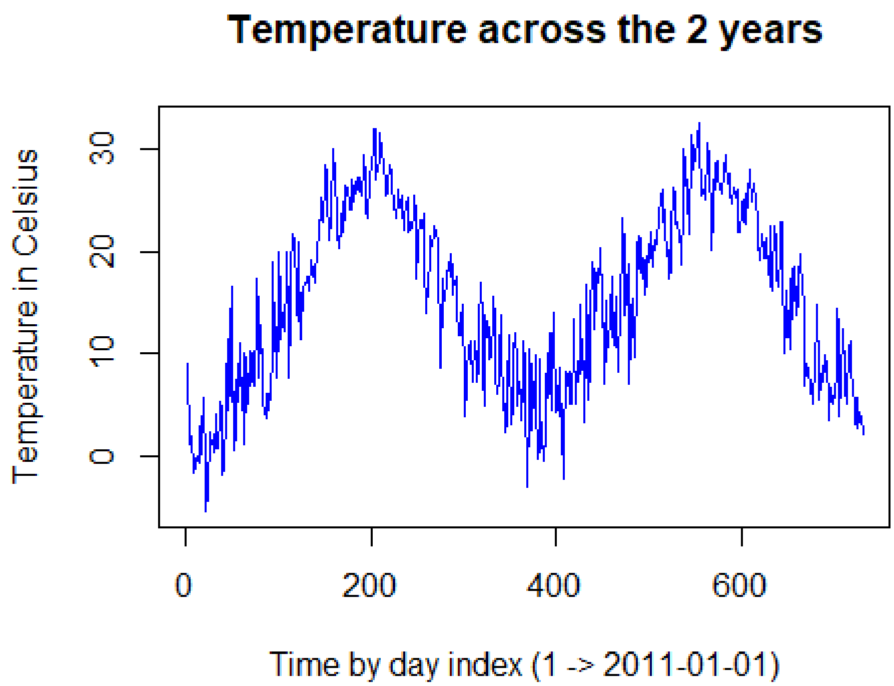
```
dim(day.data)

## [1] 731  16

boxplot(day.data$casual ,day.data$registered, col = c("blue", "green"), ylab
= "Number of bikes", xlab = "Rental method (Casual - Registered")
```



Q1 - How do the temperatures change across the seasons?

We need to add a new variable containing the temperature in celsius. Then we create a

```
day.data$temp.cel <- day.data$temp*(39 + 8) - 8
day.temp <- ts(day.data$temp.cel)
plot(day.temp, main = "Temperature across the 2 years", ylab="Temperature in Celsius", col="blue", xlab="Time by day index (1 -> 2011-01-01)")
```



The temperatures is incressing at the beginning of every year (spring and summer), reaching it's peak at summer then keep decressing through (fall and winter) until the end of the year.

Q2 - What are the mean and median temperatures?

- Mean and Median temperature of Spring

```
spring.temp <- subset(day.data, season == 1)$temp.cel
print(mean(spring.temp))

## [1] 5.994135

print(median(spring.temp))

## [1] 5.434151
```

- Mean and Median temperature of Summer

```
summer.temp <- subset(day.data, season == 2)$temp.cel
print(mean(summer.temp))

## [1] 17.58704

print(median(summer.temp))

## [1] 18.41792
```

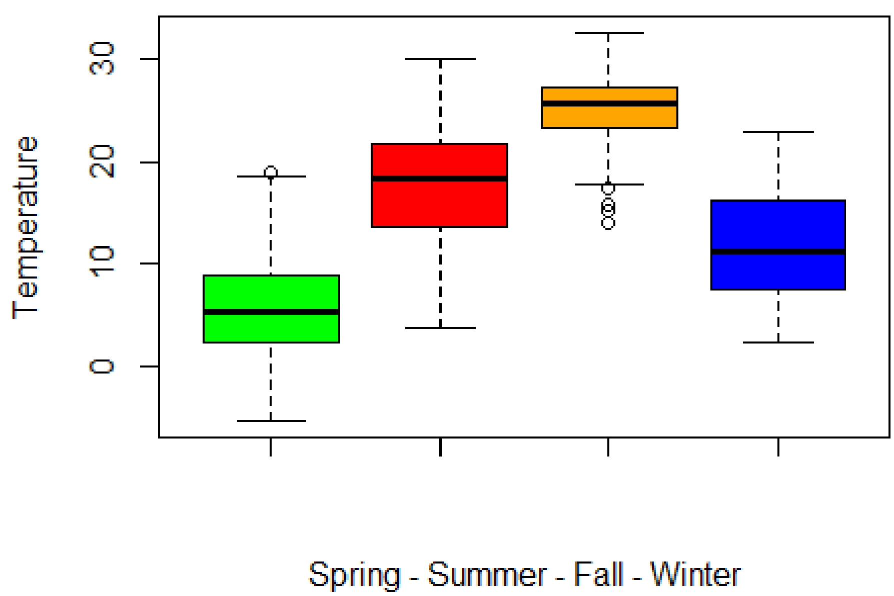
- Mean and Median temperature of Fall

```
fall.temp <- subset(day.data, season == 3)$temp.cel
print(mean(fall.temp))
```

```
## [1] 25.19654

print(median(fall.temp))

## [1] 25.5854
```

- Mean and Median temperature of Winter

```
winter.temp <- subset(day.data, season == 4)$temp.cel
print(mean(winter.temp))

## [1] 11.87658

print(median(winter.temp))

## [1] 11.23083

boxplot(subset(day.data, season == 1)$temp.cel, subset(day.data, season ==
2)$temp.cel, subset(day.data, season == 3)$temp.cel, subset(day.data, season
== 4)$temp.cel, col = c("green", "red", "orange", "blue"), xlab = "Spring -
Summer - Fall - Winter", ylab = "Temperature")
```



Q3 - Is there a correlation between the temp/atemp/mean.temp.atemp and the total count of bike rentals?

First we add two new columns to the dataset.

```
day.data$atemp.cel <- day.data$atemp*(50 + 16) - 16
day.data$mean.temp.atemp = (day.data$temp.cel + day.data$atemp.cel)/2
```

Then we calculate correlations: * Correlation between the temp and cnt

```
cor(day.data$temp.cel, day.data$cnt, method = c("pearson"))

## [1] 0.627494
```

- Correlation between the atemp and cnt

```
cor(day.data$atemp.cel, day.data$cnt, method = c("pearson"))

## [1] 0.6310657
```

- Correlation between the mean.temp.atemp and cnt

```
cor(day.data$mean.temp.atemp, day.data$cnt, method = c("pearson"))

## [1] 0.6307721
```

Correlation values are greater than **0.6** . We can affirme that there is a correlation between the 3 temperature variables and the total count of bike rentals.

Q4 - What are the mean temperature, humidity, windspeed and total rentals per months?

```
header <- c("Month", "Mean Temperature", "Mean Humidity", "Mean Windspeed",
"Total rentals")
per.months.df <- data.frame()
for (i in (1:12)) {
  sub.data <- subset(day.data, mnth == i)
  line <-  c(i, mean((sub.data)$temp.cel), mean((sub.data)$hum*100),
             mean((sub.data)$windspeed*67), sum((sub.data)$cnt))
  per.months.df = rbind(per.months.df, line)
}
colnames(per.months.df) <- header
per.months.df

##    Month Mean Temperature Mean Humidity Mean Windspeed Total rentals
## 1      1         3.112865      58.58283       13.82229        134933
## 2      2         6.063643      56.74647       14.45082        151352
## 3      3        10.355322      58.84750       14.92086        228920
## 4      4        14.089945      58.80631       15.71031        269094
## 5      5        19.955526      68.89583       12.26026        331686
## 6      6        24.152568      57.58055       12.42313        346342
## 7      7        27.507110      59.78763       11.12594        344948
## 8      8        25.303334      63.77301       11.58552        351194
## 9      9        20.974793      71.47144       11.11832        345991
## 10    10        14.795573      69.37609       11.73877        322352
## 11    11         9.353329      62.48765       12.31470        254831
## 12    12         7.229455      66.60405       11.83280        211036
```

Q5 - Is temperature associated with bike rentals (registered vs. casual)?

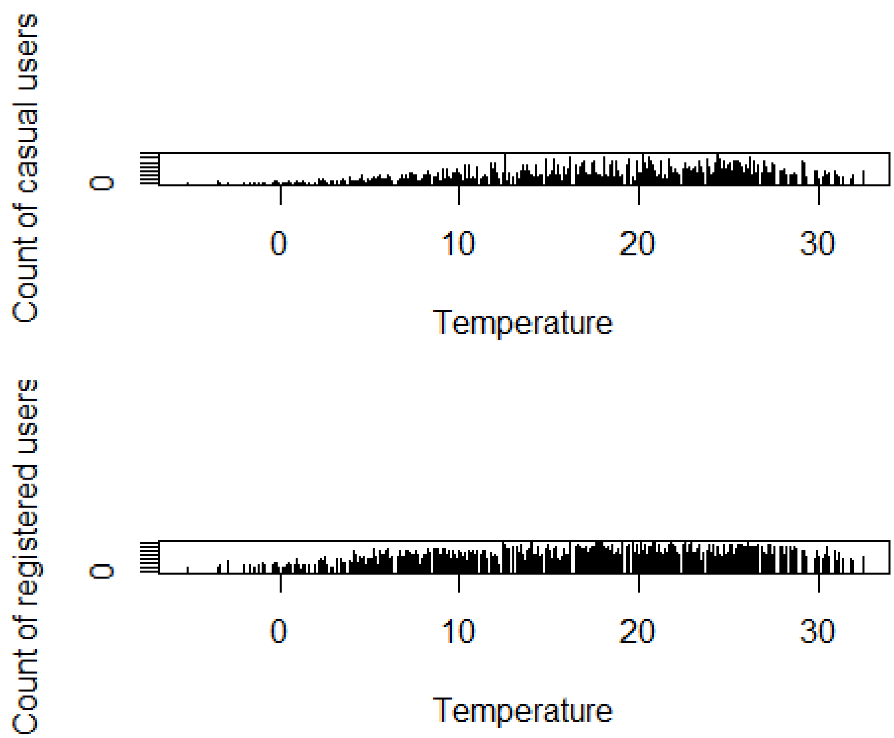
First, we calculate correlations…

```
cor(day.data$temp.cel, day.data$casual, method = c("pearson"))
```

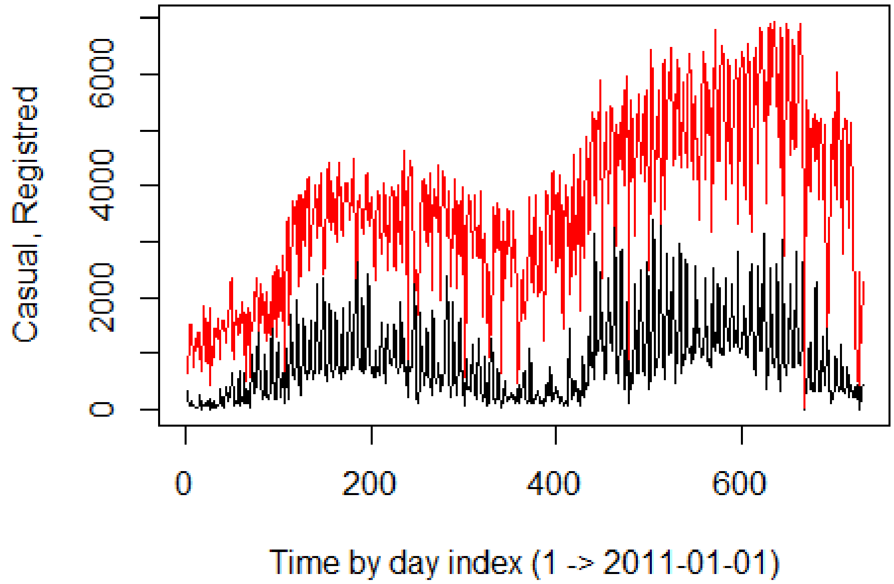
```
## [1] 0.5432847

cor(day.data$temp.cel, day.data$registered, method = c("pearson"))

## [1] 0.540012
```

Temperature and the count of bike users are not correlated!

```
day.casual <- ts(day.data$casual)
day.registered <- ts(day.data$registered)
par(mfrow=c(2,1))
plot(day.temp, day.casual, type="h", xlab="Temperature", ylab="Count of
casual users")
plot(day.temp, day.registered, type="h", xlab="Temperature", ylab="Count of
registered users")
```




```
seqplot.ts(day.casual, day.registered, ylab = "Casual, Registred", xlab="Time
by day index (1 -> 2011-01-01)")
```

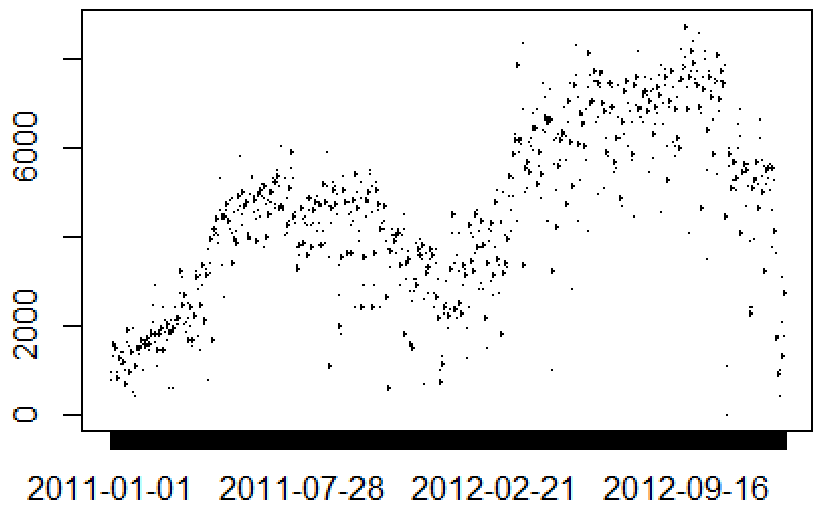


Both registered bike rentals and casual bike rentals get their highest values when when temperature is between 14 and 24 degrees. And for both rental methods, the number of rentals is incressing with temperature, reaching a peak at 20 degrees, then decreases slowly when temperature is high.

In the following, we you build a predictive model ff the number of bike sharing by day.

Q6 - Plot the cnt vs dteday and examine its patterns and irregularities

```
par(mfrow=c(1,1))
plot(day.data$dteday, day.data$cnt, type="h", col="blue")
```

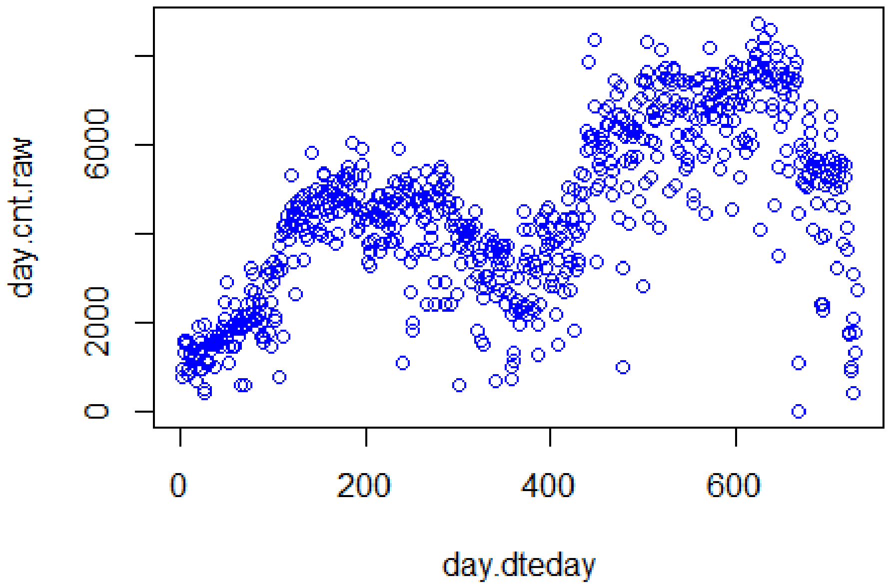


The count of total rental bikes plotted through time makes a double bell-like graph, showing the same overall shape, and an increase in bike rental in 2012.

Q7 - Clean up any outliers or missing values if needed

Now we create two new time series. day.cnt.raw contains the initial time series (before cleaning) and day.dteday for the time dimension.

```
day.cnt.raw <- ts(day.data$cnt)
day.dteday <- ts(day.data$dteday)
plot(day.dteday, day.cnt.raw, col="blue")
```

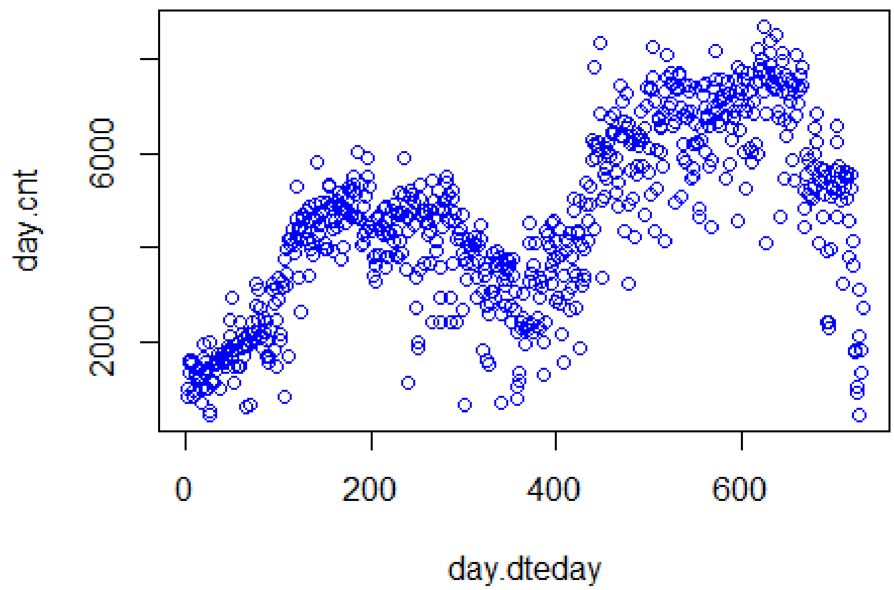


Then we remove outliners using tsclean() . We also extract the outliers.

```
day.cnt <- tsclean(day.cnt.raw)
outliners <- day.cnt.raw[day.cnt!=day.cnt.raw]
outliners

## [1] 1027 2843 3510   22 1096
```

Finaly, we can plot the cleaned time series.

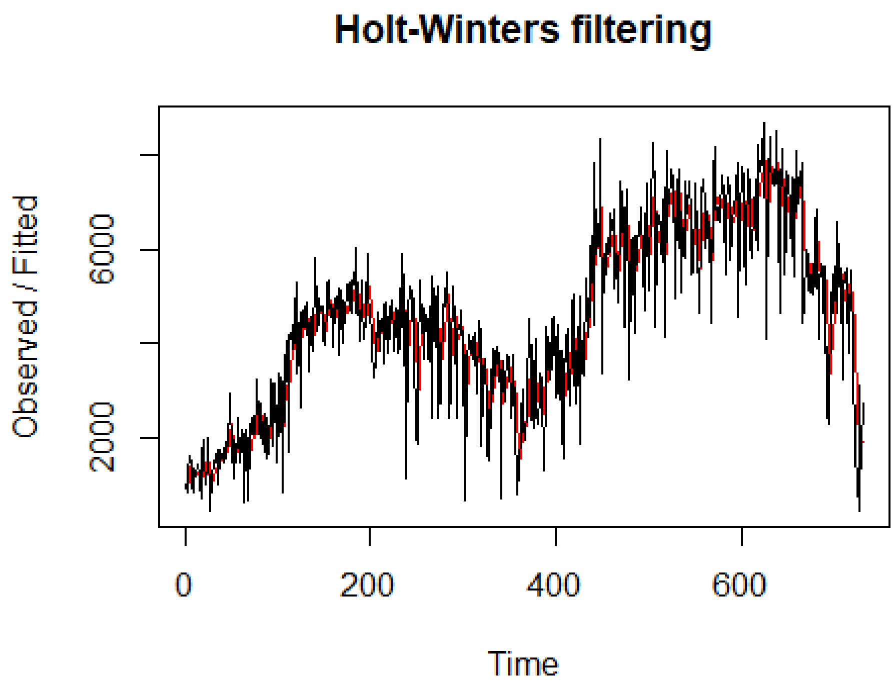
```
plot(day.dteday, day.cnt, col="blue")
```



Q8 - Smooth your time series and compare with the original

For this part, I'm using two types of smoothing : a Simple Exponential Smoothing and a Simple Moving Average with order 10 .

```
day.cnt.smoothed.se <- HoltWinters(day.cnt, beta=FALSE, gamma=FALSE)
day.cnt.smoothed.se

## Holt-Winters exponential smoothing without trend and without seasonal
component.
##
## Call:
## HoltWinters(x = day.cnt, beta = FALSE, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.2885039
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 2121.114

plot(day.cnt.smoothed.se)
```



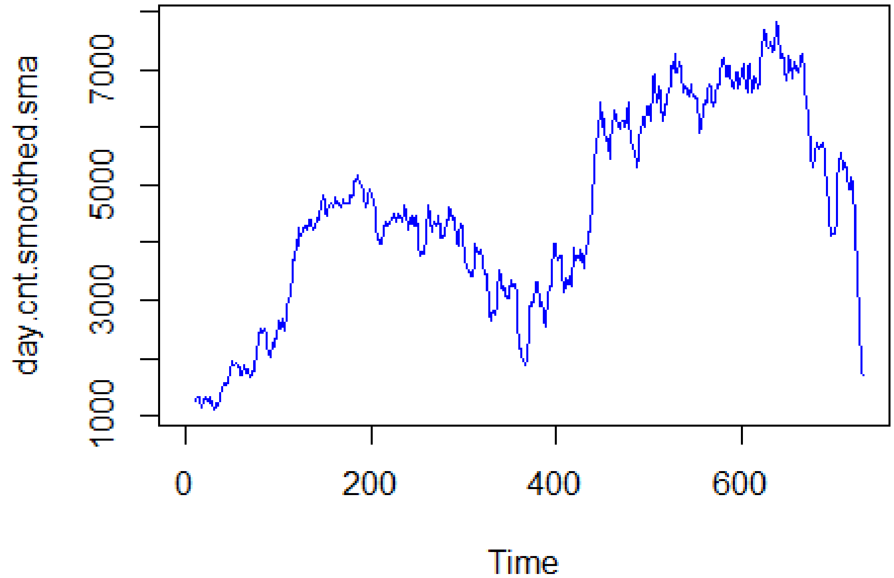

```
day.cnt.smoothed.sma <- SMA(day.cnt,n=10)
day.cnt.smoothed.sma

## Time Series:
## Start = 1
## End = 731
## Frequency = 1
##   [1]      NA      NA      NA      NA      NA      NA      NA      NA
NA
##  [10] 1251.50 1279.30 1315.40 1321.10 1307.00 1271.80 1231.60 1180.60
1153.00
##  [19] 1235.80 1296.40 1324.40 1306.30 1264.30 1263.80 1337.50 1267.70
1210.80
##  [28] 1259.20 1204.00 1120.90 1116.70 1154.60 1208.60 1222.00 1194.30
1244.20
##  [37] 1363.40 1417.90 1461.10 1512.00 1515.70 1554.30 1548.90 1552.80
1573.30
##  [46] 1654.30 1703.50 1779.80 1919.50 1922.50 1949.90 1886.00 1883.80
1916.60
##  [55] 1906.00 1870.60 1856.00 1848.70 1700.60 1722.20 1754.40 1812.20
1861.60
##  [64] 1877.60 1757.40 1798.50 1814.90 1763.80 1681.50 1694.10 1693.90
1767.10
##  [73] 1777.30 1775.20 1933.90 2021.10 2131.70 2254.30 2439.10 2449.10
2506.20
##  [82] 2476.60 2458.50 2473.90 2504.30 2399.20 2278.10 2208.90 2115.40
2076.20
```

```
##  [91] 2028.60 2041.70 2180.10 2270.60 2200.50 2312.00 2423.30 2327.90
2419.80
## [100] 2540.80 2652.90 2631.10 2522.40 2537.60 2670.70 2469.40 2529.70
2725.50
## [109] 2800.40 2905.30 2989.40 2954.30 3141.70 3234.10 3328.80 3689.30
3702.10
## [118] 3765.00 3904.10 4040.90 3957.10 4228.90 4270.40 4114.60 4150.60
4171.40
## [127] 4255.60 4283.10 4259.80 4208.90 4292.00 4338.30 4303.70 4381.30
4393.30
## [136] 4328.30 4269.20 4221.40 4242.70 4254.10 4416.40 4396.00 4412.90
4521.20
## [145] 4563.70 4635.60 4691.20 4781.50 4802.80 4720.90 4538.60 4470.00
4539.40
## [154] 4621.40 4657.80 4680.70 4667.60 4675.10 4636.40 4618.10 4678.50
4777.70
## [163] 4726.90 4697.70 4652.60 4680.00 4601.90 4603.00 4674.80 4757.70
4700.10
## [172] 4687.00 4691.70 4668.70 4678.70 4680.90 4834.70 4821.10 4774.00
4822.10
## [181] 4972.60 5025.30 5086.50 5072.40 5177.60 5123.90 5056.30 5044.70
4983.90
## [190] 4995.00 4931.60 4804.00 4717.90 4687.20 4591.30 4678.60 4808.00
4879.00
## [199] 4920.80 4841.30 4786.40 4756.20 4669.10 4563.40 4415.60 4245.80
4112.50
## [208] 4047.90 4041.10 3971.60 3985.90 4037.70 4125.60 4281.60 4278.40
4352.00
## [217] 4379.60 4343.40 4282.90 4330.90 4343.60 4391.40 4444.00 4450.00
4507.60
## [226] 4432.00 4379.20 4422.30 4513.20 4461.10 4416.20 4457.30 4365.40
4350.70
## [235] 4525.20 4656.20 4576.60 4570.20 4212.30 4265.20 4313.30 4314.60
4433.10
## [244] 4468.80 4352.00 4287.40 4427.20 4296.20 4455.70 4221.90 3942.70
3776.70
## [253] 3805.40 3798.50 3797.10 3825.00 3809.50 3840.30 4045.30 4296.80
4540.00
## [262] 4639.50 4469.10 4399.70 4407.90 4171.10 4234.90 4370.00 4357.00
4317.90
## [271] 4281.20 4311.20 4467.30 4275.00 4087.30 4204.80 4108.10 4089.70
4103.20
## [280] 4189.70 4339.90 4407.10 4398.60 4612.00 4561.80 4496.10 4414.90
4454.00
## [289] 4481.60 4440.10 4374.00 4065.30 3973.10 3947.20 4136.40 4283.20
4337.50
## [298] 4284.50 4169.80 3978.70 3878.60 3698.90 3612.50 3549.00 3525.00
3505.50
## [307] 3484.20 3420.10 3423.30 3522.30 3551.10 3908.90 3986.70 3913.10
3843.10
```

```
## [316] 3831.20 3805.50 3849.50 3876.40 3693.20 3595.00 3513.70 3469.10
3527.80
## [325] 3467.50 3221.50 3106.40 2807.30 2667.00 2792.10 2793.90 2841.40
2766.50
## [334] 2775.80 2872.00 3105.30 3210.10 3409.10 3511.00 3463.60 3227.00
3172.50
## [343] 3243.10 3200.80 3102.40 3039.40 3030.30 3055.80 3045.60 3143.90
3347.30
## [352] 3258.20 3236.50 3292.50 3284.20 3260.00 3128.60 2855.70 2560.20
2334.20
## [361] 2176.50 2163.60 2065.60 1990.50 1973.00 1895.60 1869.80 1992.30
2153.70
## [370] 2349.20 2642.80 2864.70 2964.90 2902.60 3013.90 3002.20 3216.80
3314.60
## [379] 3327.10 3231.00 3051.00 2892.40 2887.50 2979.10 2935.60 2848.00
2636.00
## [388] 2557.80 2742.40 2938.30 3116.00 3168.10 3232.80 3227.90 3274.00
3594.80
## [397] 3855.00 3987.90 3969.10 3825.30 3712.50 3745.30 3780.50 3736.40
3757.00
## [406] 3689.20 3448.20 3225.00 3152.10 3261.10 3383.30 3305.40 3283.30
3434.90
## [415] 3320.80 3250.60 3411.40 3735.80 3899.80 3856.30 3712.60 3751.00
3767.80
## [424] 3772.30 3686.80 3872.90 3814.60 3743.90 3580.00 3564.60 3687.00
3839.70
## [433] 3945.70 3966.30 4194.70 4186.80 4397.20 4575.30 4864.20 5150.10
5192.30
## [442] 5484.30 5535.30 5693.70 5891.20 6023.10 6180.40 6431.90 6137.90
6018.30
## [451] 6136.30 5862.90 5843.50 5841.50 5778.10 5778.60 5695.60 5453.00
5793.00
## [460] 5937.00 6026.90 6162.70 6278.60 6182.20 6194.80 6163.10 6045.20
5992.50
## [469] 5955.10 6057.50 6125.00 6116.00 6099.40 6019.20 6117.20 6254.40
6430.60
## [478] 6381.60 6063.20 5880.50 5786.90 5652.50 5606.70 5592.00 5565.90
5394.10
## [487] 5305.70 5430.70 5751.40 5817.70 5886.40 6019.70 6023.70 6174.50
6015.80
## [496] 6115.80 6244.80 6370.80 6340.50 6272.55 6095.75 6202.25 6313.35
6504.45
## [505] 6862.15 6917.85 6650.75 6515.15 6429.35 6544.70 6706.60 6617.80
6538.50
## [514] 6378.90 6123.80 6096.40 6394.30 6199.70 6485.70 6572.80 6599.20
6645.70
## [523] 6692.10 6837.20 7036.50 7100.80 7026.80 7280.50 6965.70 6943.70
6980.20
## [532] 7046.60 7111.30 7059.70 6796.00 6728.70 6690.00 6614.10 6699.20
6702.90
```

```
## [541] 6655.70 6567.10 6541.10 6576.80 6754.80 6618.60 6566.20 6528.80
6569.20
## [550] 6489.40 6540.60 6486.80 6363.30 6113.80 5893.10 6003.70 6064.00
6237.30
## [559] 6359.20 6443.10 6399.70 6378.70 6441.00 6635.60 6739.70 6741.90
6699.90
## [568] 6419.40 6415.80 6362.50 6424.80 6639.00 6642.10 6653.90 6751.10
6751.70
## [577] 6875.20 7150.90 7167.90 7197.40 7155.70 7020.80 6881.10 6892.00
6950.80
## [586] 7044.50 7062.60 6919.60 6791.50 6719.80 6690.60 6686.60 6874.90
6934.10
## [595] 6921.60 6954.70 6681.00 6755.40 6826.10 6909.20 6997.40 7077.20
6947.80
## [604] 6712.80 6689.70 6607.20 6922.00 7040.30 7074.70 6951.20 6755.70
6600.90
## [613] 6682.00 6867.70 6796.30 6842.70 6670.60 6722.00 6739.50 6902.20
7108.20
## [622] 7285.20 7399.70 7559.90 7672.90 7609.40 7419.10 7355.50 7375.00
7415.00
## [631] 7467.50 7477.80 7420.50 7302.90 7342.90 7395.30 7729.50 7825.90
7742.80
## [640] 7603.90 7228.30 7194.80 7184.00 7245.80 7269.00 7201.85 7008.15
6791.85
## [649] 6872.05 6951.25 7215.55 7169.25 7100.35 6872.25 6829.15 6903.10
7106.20
## [658] 7009.40 7049.30 6974.70 6952.30 6988.00 7093.40 7241.80 7232.80
7271.90
## [667] 6966.90 6907.30 6618.00 6492.20 6385.00 6223.10 5967.60 5742.40
5523.90
## [676] 5307.30 5364.90 5413.60 5493.10 5590.10 5676.70 5718.90 5614.50
5653.30
## [685] 5671.90 5673.10 5732.50 5667.90 5618.60 5528.40 5357.80 4973.40
4955.00
## [694] 4633.20 4331.10 4270.00 4103.00 4162.10 4144.50 4147.90 4152.40
4374.80
## [703] 4607.20 5040.10 5370.60 5399.40 5504.30 5536.50 5327.00 5277.20
5308.20
## [712] 5375.20 5305.00 5205.50 5137.30 4978.40 4936.10 4933.60 5137.50
5033.30
## [721] 4845.50 4488.50 4114.00 3644.90 3241.50 2907.00 2659.90 2413.70
2021.10
## [730] 1787.90 1698.50
```

```
plot(day.cnt.smoothed.sma, col="blue")
```

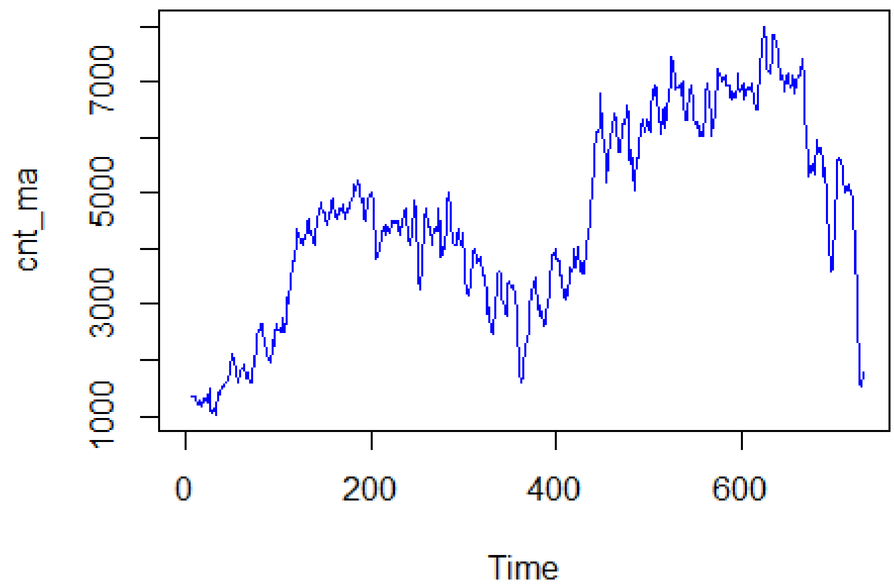


2. Decompose your data

Now we will be using the smoothed time series with order 7, that we will name hereafter cnt_ma.

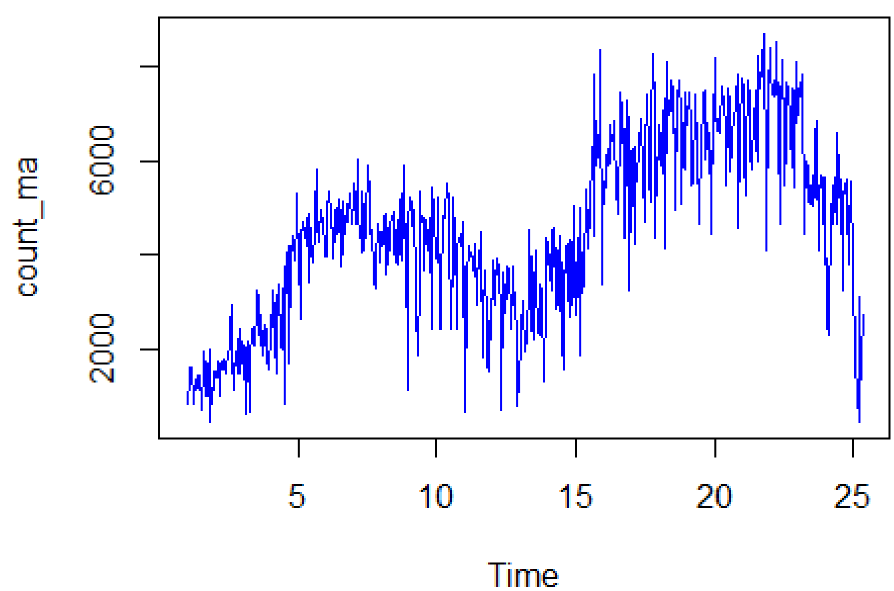
First we create the smoothed time series with order 7.

```
cnt_ma <- SMA(day.cnt,n=7)
plot(cnt_ma, col="blue")
```



Q1 - Transform cnt_ma into a time series with frequency 30 named count_ma.

```
count_ma <- ts(day.cnt, frequency = 30)
plot(count_ma, col="blue")
```

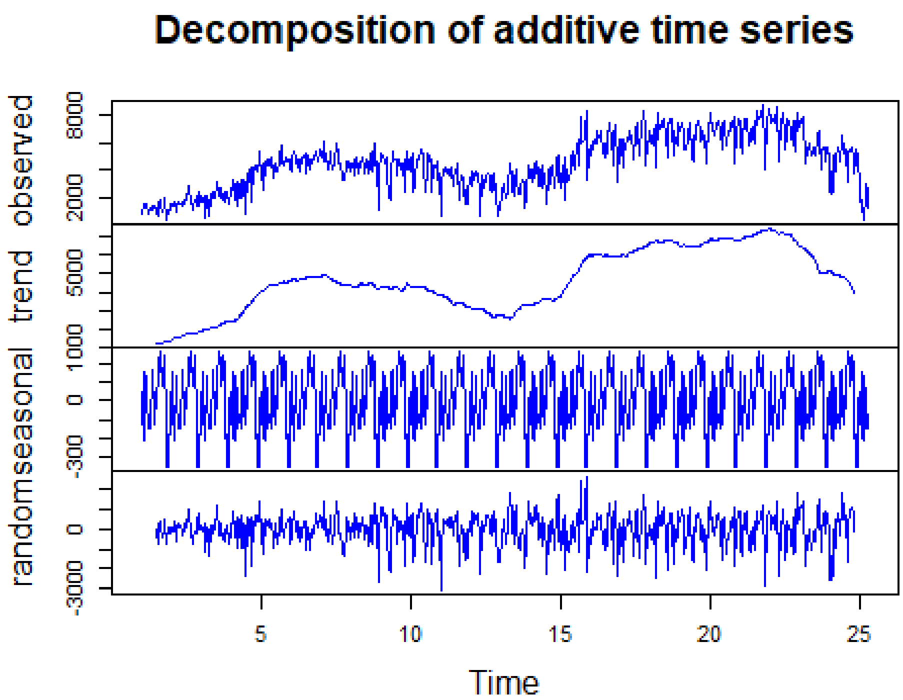


Q2 - Does the series count_ma appear to have trends or seasonality?

Yes, the count_ma time series shows a general tendency of rental increasing in the first two seasons then decreases in the last two seasons. The time series also shows a repeating short-term cycle through the two years.

Q3 - Use decompose() or stl() to examine and possibly remove components of the series

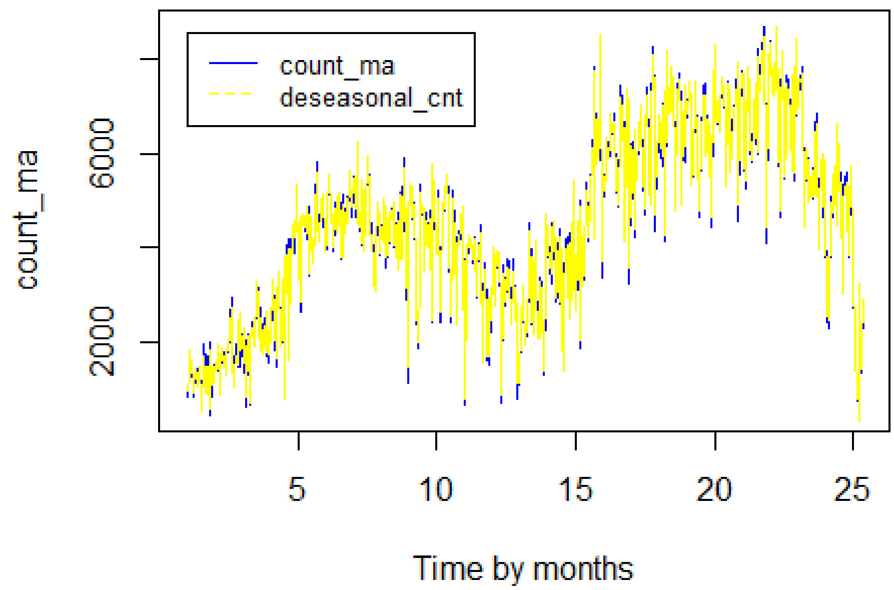
```
count_ma.decomposed <- decompose(count_ma)
plot(count_ma.decomposed, col="blue")
```



The seasonal component confirms our hypothesis, and shows up even a more interesting seasonal monthly cycle.

Q4 - Create a time series deseasonal_cnt by removing the seasonal component

```
deseasonal_cnt <- count_ma - count_ma.decomposed$seasonal
plot(count_ma, col = "blue", xlab="Time by months")
legend(1, 8600, legend=c("count_ma", "deseasonal_cnt"), col=c("blue",
"yellow"), lty=1:2, cex=0.8)
lines(deseasonal_cnt, col = 'yellow')
```



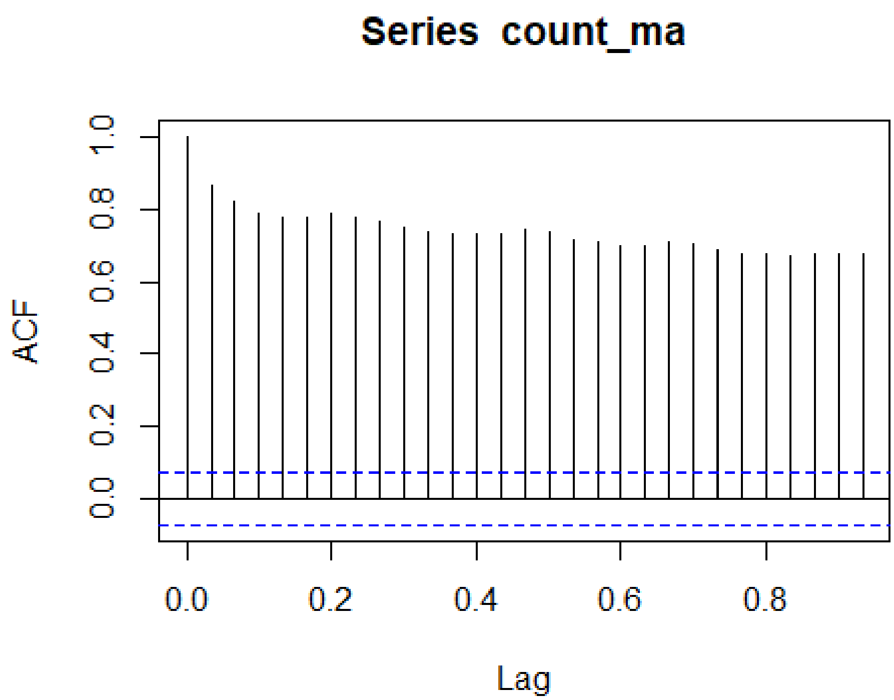
3 - Stationarity

** Q - Is the serie count_ma stationary? If not, how to make stationary**

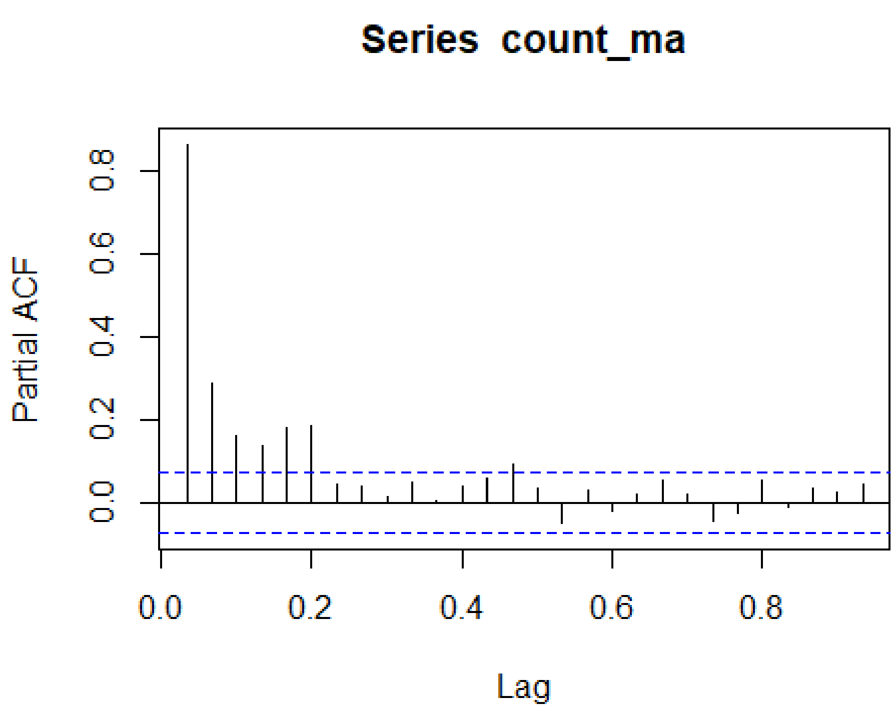
```
adf.test(count_ma, alternative = "stationary")

##
##  Augmented Dickey-Fuller Test
##
## data:  count_ma
## Dickey-Fuller = -1.3084, Lag order = 9, p-value = 0.871
## alternative hypothesis: stationary
```

The p-value is greater that 0.05 , therefore count_ma is not stationary ACF describes how well the present value of the series is related with its past values. Here, it also confirms that the series is not stationary since the autocorrelation is falling gradually.

```
acf(count_ma)
```




```
pacf(count_ma)
```



To make the time series stationary, we'll be using Differencing. Differencing is a process of subtracting each data point in the series from its successor. First, we need to know how many differencing is needed.

```
count_ma.diff1 <- diff(count_ma,differences = 1)
adf.test(count_ma.diff1, alternative = "stationary")
```

```
## Warning in adf.test(count_ma.diff1, alternative = "stationary"): p-value
smaller
## than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  count_ma.diff1
## Dickey-Fuller = -13.499, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

The p-value is smaller than 0.05 . count_ma.diff1 is stationary. We conclude that we only need **one** differencing to make count_ma stationary.

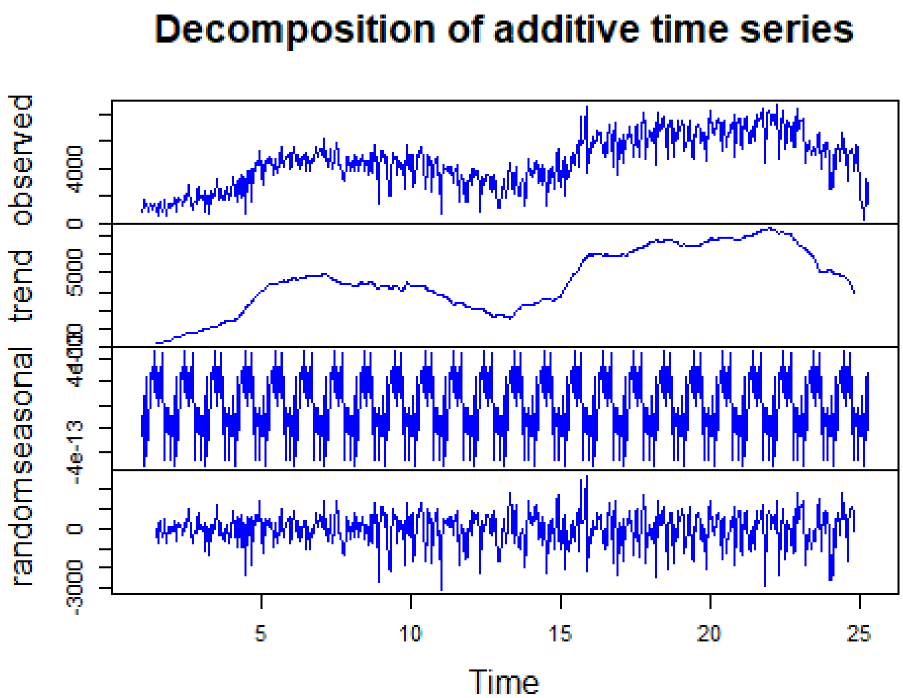
4 - Forecasting with ARIMA Models

I - Fitting ARIMA model

Q1 - Fit an ARIMA model to deseasonal_cnt

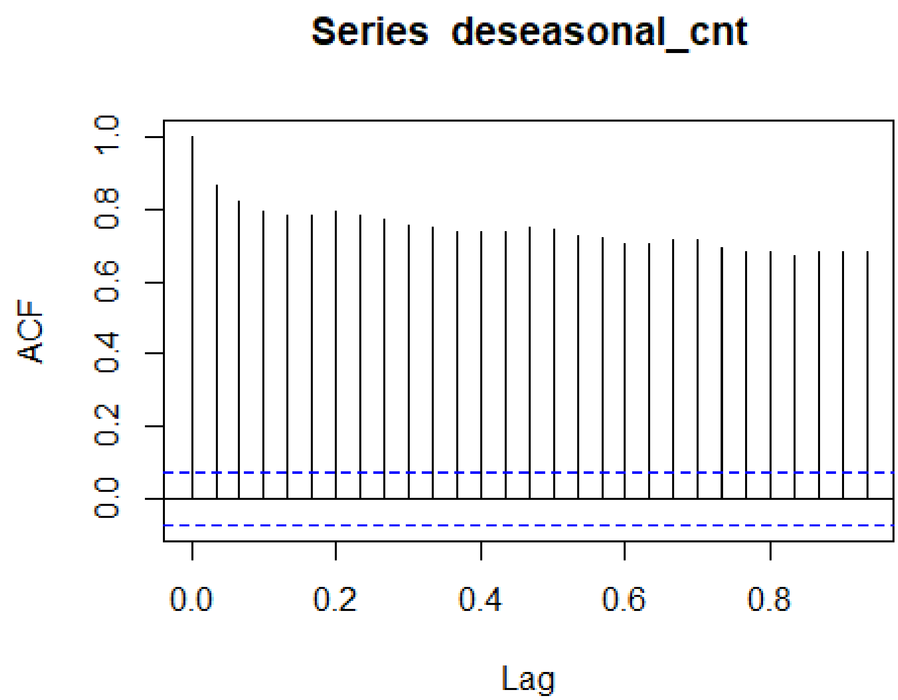
First, we need to check if the time series has a tendency.

```
deseasonal_cnt.decomposed <- decompose(deseasonal_cnt)
plot(deseasonal_cnt.decomposed, col="blue")
```

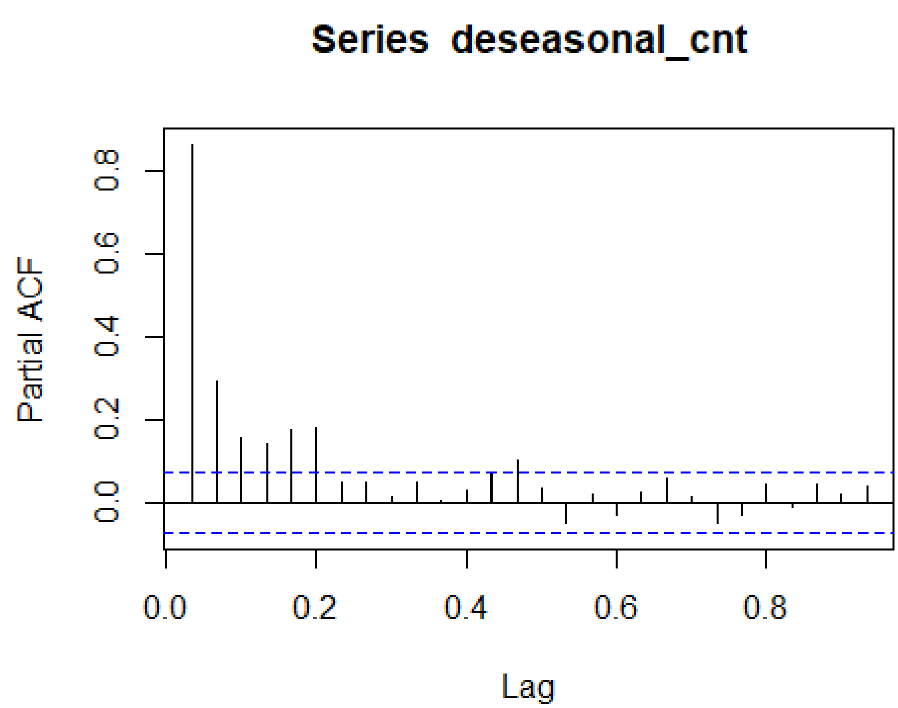


deseasonal_cnt doesn't have a tendency but have a seasonality. Now we check if the time series is stationary.

```
adf.test(deseasonal_cnt, alternative = "stationary")

##
##  Augmented Dickey-Fuller Test
##
## data:  deseasonal_cnt
## Dickey-Fuller = -1.2425, Lag order = 9, p-value = 0.899
## alternative hypothesis: stationary

acf(deseasonal_cnt)
```




```r
pacf(deseasonal_cnt)
```



deseasonal_cnt is not stationnary, we need to do a differencing.

```r
deseasonal_cnt.diff1 <- diff(deseasonal_cnt,differences = 1)
adf.test(deseasonal_cnt.diff1, alternative = "stationary")
```

```
## Warning in adf.test(deseasonal_cnt.diff1, alternative = "stationary"): p-
value
## smaller than printed p-value

## 
##  Augmented Dickey-Fuller Test
## 
## data:  deseasonal_cnt.diff1
## Dickey-Fuller = -13.594, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

deseasonal_cnt.diff1 is stationary, since its p-value is less that 0.05 .adf.test returns also the lag order q = 8, and we have d = 1. From PACF, it's clearly that within 6 lags the AR is significant. which means, we can use p = 6 .

```r
deseasonal_cnt.arima <- arima(deseasonal_cnt.diff1, order = c(6,0,8))
```

```
## Warning in arima(deseasonal_cnt.diff1, order = c(6, 0, 8)): possible
convergence
## problem: optim gave code = 1
```

```r
deseasonal_cnt.arima
```

```
## 
## Call:
## arima(x = deseasonal_cnt.diff1, order = c(6, 0, 8))
## 
## Coefficients:
##           ar1     ar2      ar3      ar4      ar5      ar6      ma1
ma2
##       -0.2466  0.0672  -1.0525  -0.4391  -0.0742  -0.5872  -0.3239  -
0.3464
## s.e.   0.2390  0.2043   0.3156   0.2940   0.1501   0.3374   0.2425
0.2659
##          ma3      ma4      ma5     ma6      ma7      ma8  intercept
##       0.9827  -0.2393  -0.3142  0.5325  -0.4767  -0.1443     1.4301
## s.e.  0.2724   0.4544   0.1088  0.3837   0.1632   0.0520     6.1858
## 
## sigma^2 estimated as 678024:  log likelihood = -5938.24,  aic = 11908.49
```

Maybe the process that we've followed is not the best way to select p d and q. We need to evaluate this model and iterate.

II - Fit an ARIMA with Auto-ARIMA

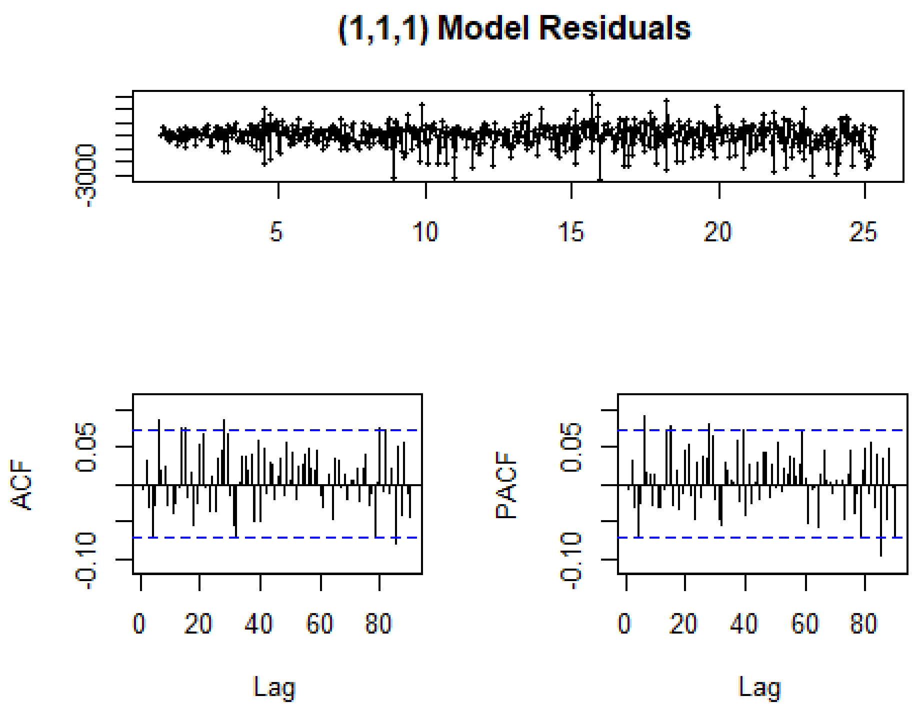
Q1 - Use auto.arima() function to fit an ARIMA model of deseasonal_cnt

```r
deseasonal_cnt.autoarima <- auto.arima(deseasonal_cnt, seasonal = FALSE)
deseasonal_cnt.autoarima
```

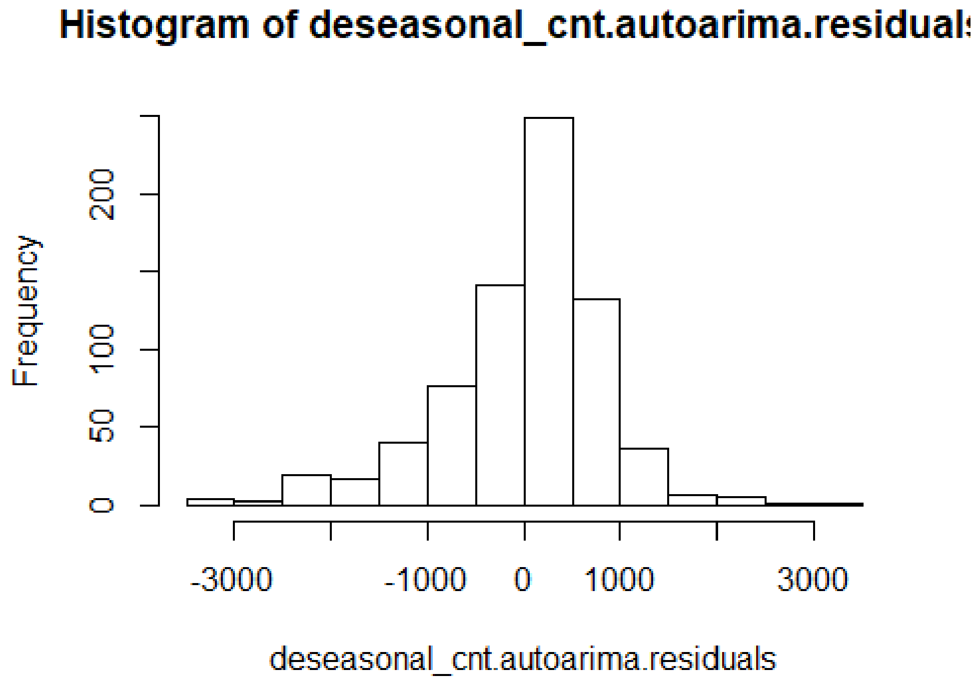
```
## Series: deseasonal_cnt
## ARIMA(1,1,1)
##
## Coefficients:
##          ar1      ma1
##       0.3037  -0.8659
## s.e.  0.0446   0.0222
##
## sigma^2 estimated as 712654:  log likelihood=-5954.27
## AIC=11914.55   AICc=11914.58   BIC=11928.33
```

Q2 - Check residuals, which should have no patterns and be normally distributed

```
deseasonal_cnt.autoarima.residuals <- deseasonal_cnt.autoarima$residuals
tsdisplay(deseasonal_cnt.autoarima.residuals, main='(1,1,1) Model Residuals')
```




```
hist(deseasonal_cnt.autoarima.residuals)
```




```
shapiro.test(deseasonal_cnt.autoarima.residuals)

##
##  Shapiro-Wilk normality test
##
## data:  deseasonal_cnt.autoarima.residuals
## W = 0.95311, p-value = 1.706e-14
```

The residuals are not normally distributed. The auto.arima() function didn't give us a good model. We should iterate!

III - Evaluate and iterate

Q1 - If there are visible patterns or bias, plot ACF/PACF

Q2 - Refit model if needed. Compare model errors and fit criteria such as AIC or BIC

We will train 10 different Arima models by changing the p-order value.

```
aic.values <- c()
for (p in (0:9)){
  deseasonal_cnt.arima <- arima(deseasonal_cnt, order = c(p,0,8))
  aic.values <- c(aic.values, deseasonal_cnt.arima$aic)
}

## Warning in arima(deseasonal_cnt, order = c(p, 0, 8)): possible convergence
## problem: optim gave code = 1
```

```
## Warning in arima(deseasonal_cnt, order = c(p, 0, 8)): possible convergence
## problem: optim gave code = 1

## Warning in arima(deseasonal_cnt, order = c(p, 0, 8)): possible convergence
## problem: optim gave code = 1
```

```
which.min(aic.values)
```

```
## [1] 8
```

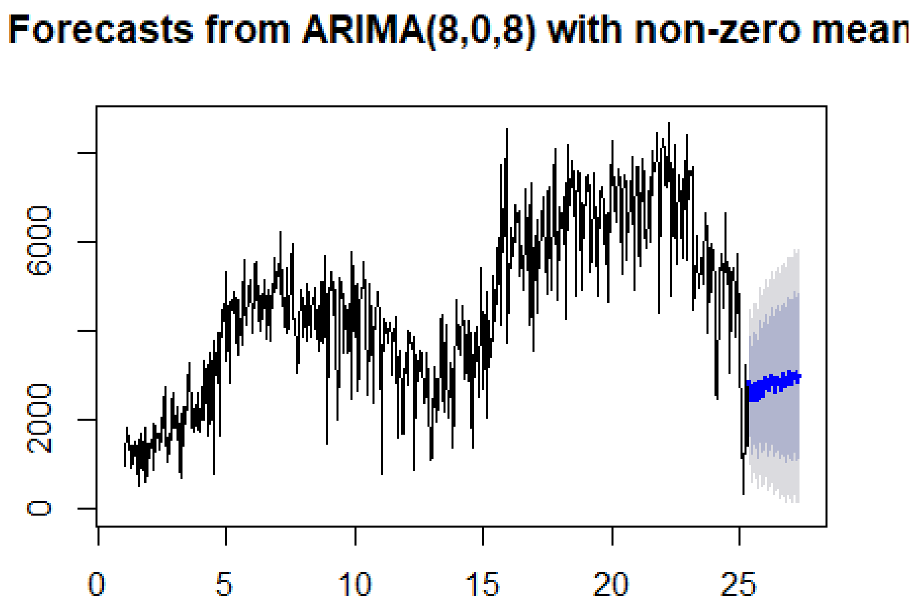
The model order that gave the minimum AIC value is p=8, d=0 and q=8 . Now we train the model to be used for forecasting.

```
deseasonal_cnt.arima <- arima(deseasonal_cnt, order = c(8,0,8))
deseasonal_cnt.arima
```

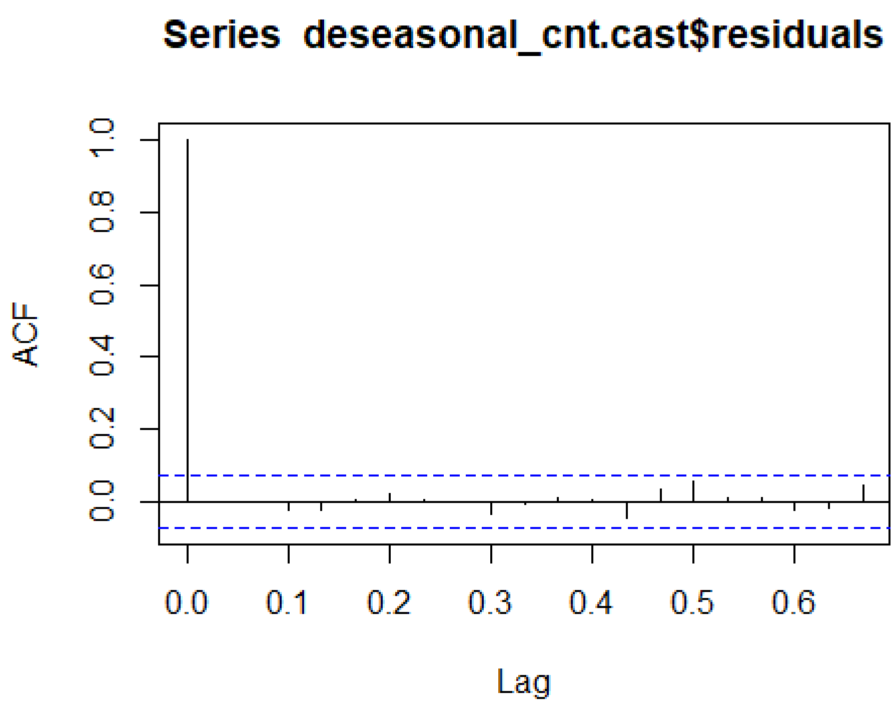
```
##
## Call:
## arima(x = deseasonal_cnt, order = c(8, 0, 8))
##
## Coefficients:
##          ar1     ar2      ar3     ar4      ar5      ar6     ar7      ar8
##       0.2761  0.1003  -0.5356  0.4939  -0.0125  -0.0407  0.8531  -0.1541
## s.e.  0.2054  0.1272   0.0601  0.1851   0.1679   0.0651  0.1127   0.1732
##          ma1     ma2     ma3      ma4     ma5     ma6      ma7      ma8
##       0.1605  0.0770  0.6546  -0.1921  0.0367  0.2042  -0.7626  -0.0691
## s.e.  0.2067  0.0889  0.0526   0.1875  0.1025  0.0601   0.1010   0.1447
##       intercept
##        4502.200
## s.e.   1710.548
##
## sigma^2 estimated as 671448:  log likelihood = -5945.3,  aic = 11926.59
```

Q3 - Calculate forecast using the chosen model

```
deseasonal_cnt.cast <- forecast(deseasonal_cnt.arima)
plot(deseasonal_cnt.cast)
```




```
acf(deseasonal_cnt.cast$residuals, lag.max=20)
```

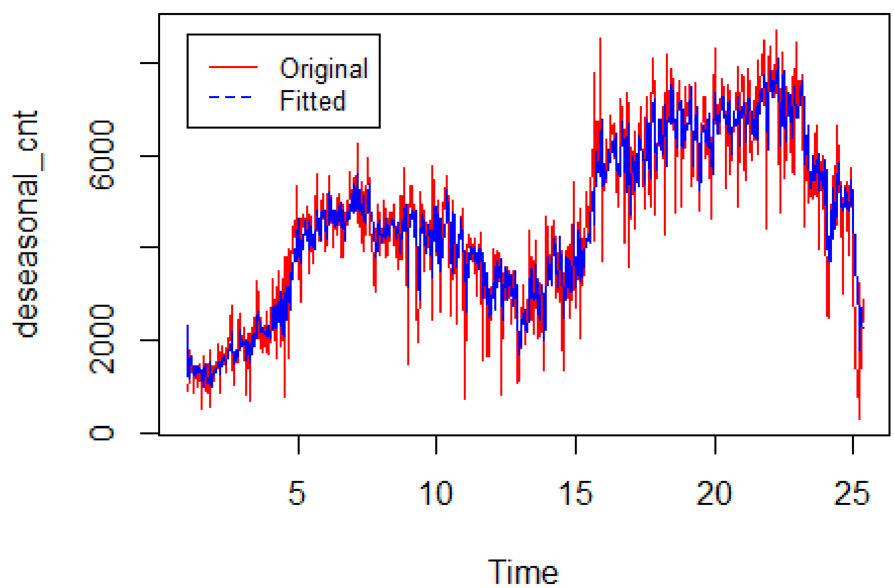



```
Box.test(deseasonal_cnt.cast$residuals, lag=20, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  deseasonal_cnt.cast$residuals
## X-squared = 8.6931, df = 20, p-value = 0.9862
```

Q4 - Plot both the original and the forecasted time series

```
plot(deseasonal_cnt, col="red") # original
legend(1, 8600, legend=c("Original", "Fitted"), col=c("red", "blue"),
lty=1:2, cex=0.8)
lines(fitted(deseasonal_cnt.arima), col="blue") # fitted
```



IV - Forecasting

Q1 - Split the data into training and test times series

```
end.time = time(deseasonal_cnt)[700]
train.set <- window(deseasonal_cnt, end=end.time)
test.set <- window(deseasonal_cnt, start=end.time)
```

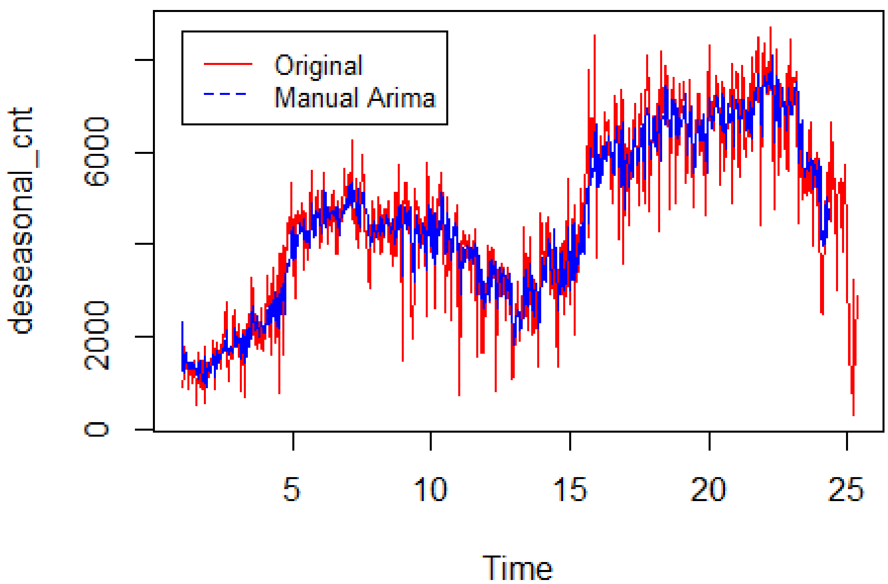
Q2 - fit an Arima model, manually and with Auto-Arima on the training part

```
manual.fit <- Arima(train.set, order=c(8, 0, 8))
manual.fc <- forecast(manual.fit, h=32)
print(paste("Accuracy of the manual Arima model : ", accuracy(manual.fc,
test.set)[2,"RMSE"]))

## [1] "Accuracy of the manual Arima model :  1866.30484049695"
```

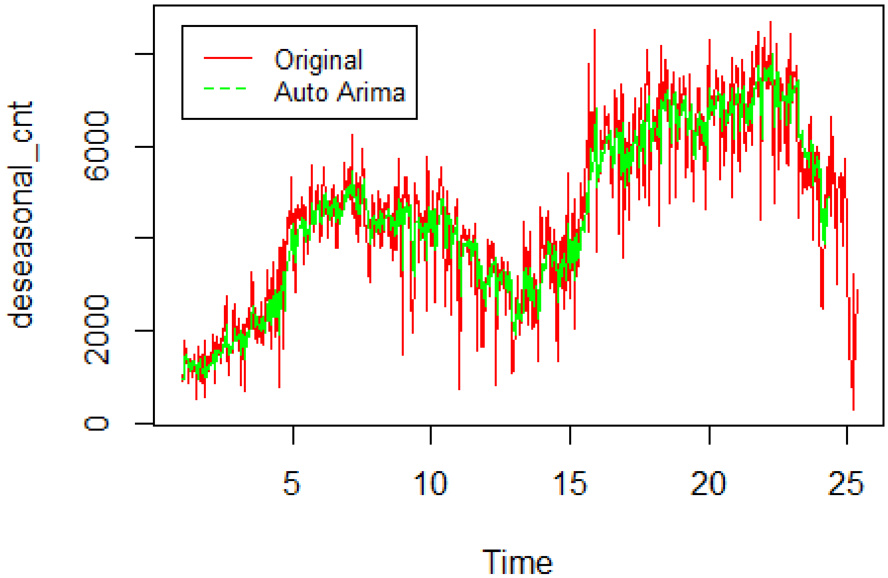
```r
auto.fit <- auto.arima(train.set, seasonal = FALSE)
auto.fc <- forecast(auto.fit, h=32)
print(paste("Accuracy of the auto Arima model : ", accuracy(auto.fc,
test.set)[2,"RMSE"]))
```

```
## [1] "Accuracy of the auto Arima model :  1939.23863437773"
```

```r
plot(deseasonal_cnt, col="red") # original
legend(1, 8600, legend=c("Original", "Manual Arima"), col=c("red", "blue"),
lty=1:2, cex=0.8)
lines(fitted(manual.fc), col="blue") # manuall arima
```

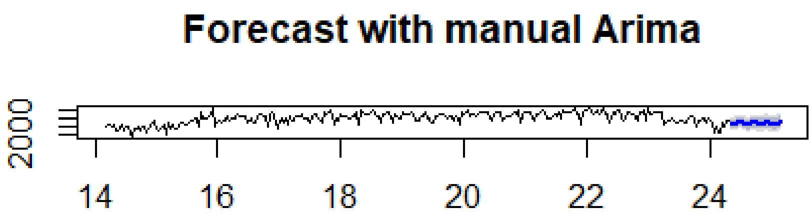


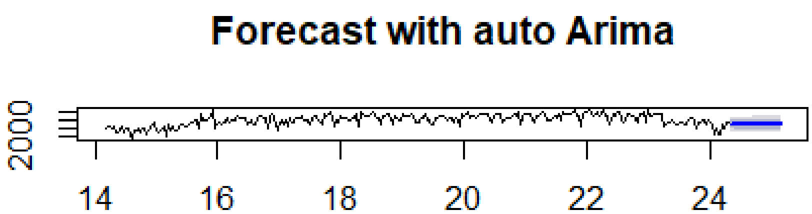

```r
plot(deseasonal_cnt, col="red") # original
legend(1, 8600, legend=c("Original", "Auto Arima"), col=c("red", "green"),
lty=1:2, cex=0.8)
lines(fitted(auto.fit), col="green") # auto arima
```



Q3 - Forecast the next 25 observation and plot the original ts and the forecasted one

```
deseasonal_cnt.forecast.manual <- forecast(manual.fit, h=25)
deseasonal_cnt.forecast.auto <- forecast(auto.fit, h=25)

par(mfrow=c(2,1))
plot(deseasonal_cnt.forecast.manual, main = "Forecast with manual Arima",
include = test.set)
plot(deseasonal_cnt.forecast.auto, main = "Forecast with auto Arima", include
= test.set)
```





The manual Arima model gives a more natural forecast than the auto Arima one.