arima_code.R

kriti

Tue Aug 21 21:00:47 2018

```
###A simple arima model##


##What is Arima model###
# ARIMA is the abbreviation for AutoRegressive Integrated Moving Average. Auto Regressive (AR) t
erms
# refer to the lags of the differenced series,
# Moving Average (MA) terms refer to the lags of errors and I is the number of difference used t
o make the time series stationary.
#


# Assumptions of ARIMA model
# 1. Data should be stationary - by stationary it means that the properties of the series does
n't depend
# on the time when it is captured. A white noise series and series with cyclic behavior can also
 be considered as stationary series.
# 2. Data should be univariate - ARIMA works on a single variable. Auto-regression is all about
 regression with the past values.


library('ggplot2')
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
library('forecast')
```

```
## Warning: package 'forecast' was built under R version 3.4.4
```

```
library('tseries')
```

```
## Warning: package 'tseries' was built under R version 3.4.4
```

```
library('fpp')
```

```
## Warning: package 'fpp' was built under R version 3.4.4
```

```
## Loading required package: fma
```

```
## Warning: package 'fma' was built under R version 3.4.4
```

```
## Loading required package: expsmooth
```

```
## Warning: package 'expsmooth' was built under R version 3.4.4
```

```
## Loading required package: lmtest
```

```
## Loading required package: zoo
```

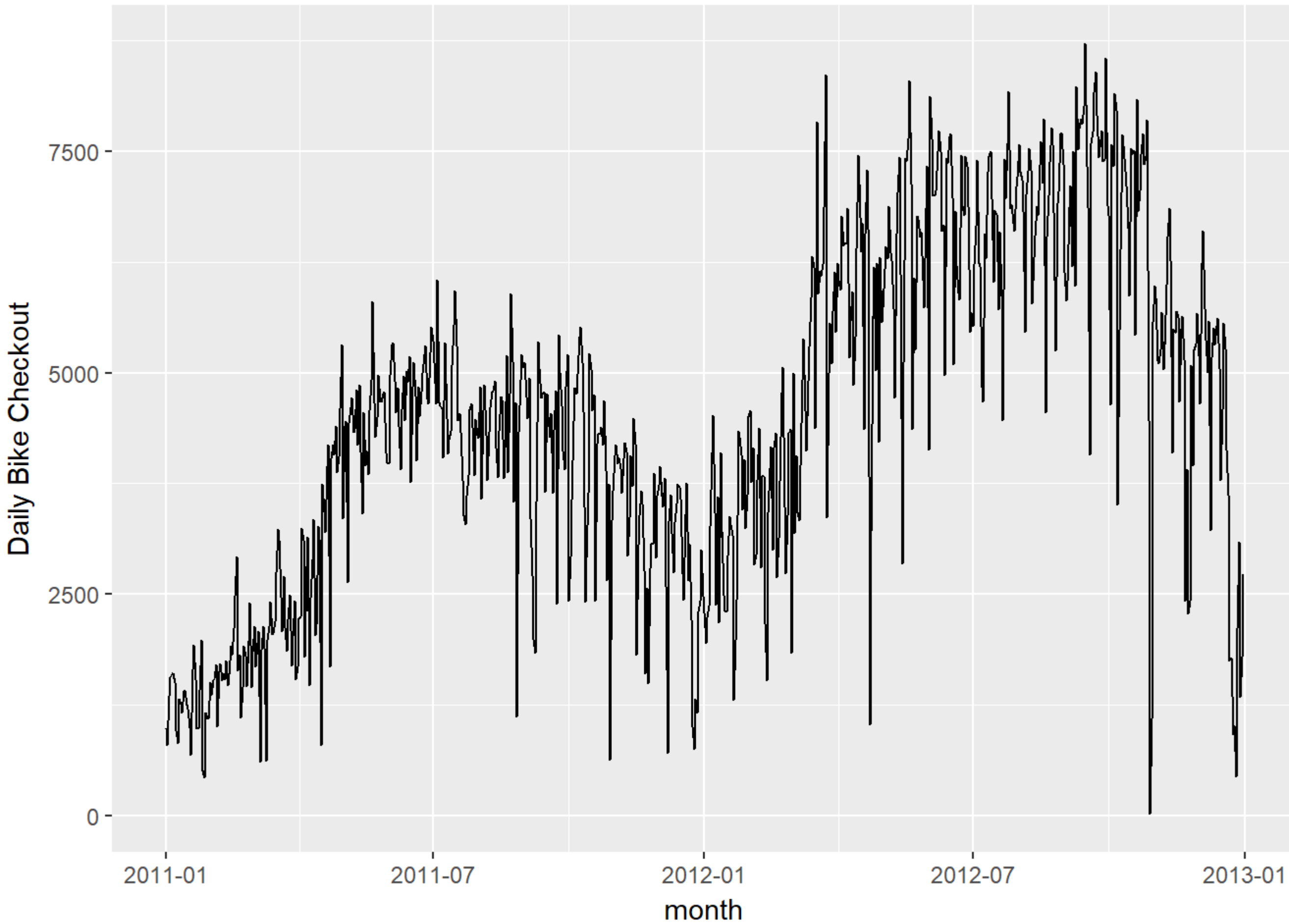
```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
getwd()
```

```
## [1] "C:/Users/kriti/Documents"
```

```
setwd('C:/arima model')
daily_data=read.csv('day.csv',header=TRUE,stringsAsFactors = FALSE)

daily_data$date=as.Date(daily_data$dteday)

ggplot(daily_data,aes(date,cnt))+geom_line()+scale_x_date('month')+ylab("Daily Bike Checkout")+x
lab("")
```




```
# 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.
# 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 decomp
osition.


##creating a time series object

count_ts=ts(daily_data[,c('cnt')])
count_ts
```

```
## Time Series:
## Start = 1
## End = 731
## Frequency = 1
##   [1]  985  801 1349 1562 1600 1606 1510  959  822 1321 1263 1162 1406 1421
##  [15] 1248 1204 1000  683 1650 1927 1543  981  986 1416 1985  506  431 1167
##  [29] 1098 1096 1501 1360 1526 1550 1708 1005 1623 1712 1530 1605 1538 1746
##  [43] 1472 1589 1913 1815 2115 2475 2927 1635 1812 1107 1450 1917 1807 1461
##  [57] 1969 2402 1446 1851 2134 1685 1944 2077  605 1872 2133 1891  623 1977
##  [71] 2132 2417 2046 2056 2192 2744 3239 3117 2471 2077 2703 2121 1865 2210
##  [85] 2496 1693 2028 2425 1536 1685 2227 2252 3249 3115 1795 2808 3141 1471
##  [99] 2455 2895 3348 2034 2162 3267 3126  795 3744 3429 3204 3944 4189 1683
## [113] 4036 4191 4073 4400 3872 4058 4595 5312 3351 4401 4451 2633 4433 4608
## [127] 4714 4333 4362 4803 4182 4864 4105 3409 4553 3958 4123 3855 4575 4917
## [141] 5805 4660 4274 4492 4978 4677 4679 4758 4788 4098 3982 3974 4968 5312
## [155] 5342 4906 4548 4833 4401 3915 4586 4966 4460 5020 4891 5180 3767 4844
## [169] 5119 4744 4010 4835 4507 4790 4991 5202 5305 4708 4648 5225 5515 5362
## [183] 5119 4649 6043 4665 4629 4592 4040 5336 4881 4086 4258 4342 5084 5538
## [197] 5923 5302 4458 4541 4332 3784 3387 3285 3606 3840 4590 4656 4390 3846
## [211] 4475 4302 4266 4845 3574 4576 4866 4294 3785 4326 4602 4780 4792 4905
## [225] 4150 3820 4338 4725 4694 3805 4153 5191 3873 4758 5895 5130 3542 4661
## [239] 1115 4334 4634 5204 5058 5115 4727 4484 4940 3351 2710 1996 1842 3544
## [253] 5345 5046 4713 4763 4785 3659 4760 4511 4274 4539 3641 4352 4795 2395
## [267] 5423 5010 4630 4120 3907 4839 5202 2429 2918 3570 4456 4826 4765 4985
## [281] 5409 5511 5117 4563 2416 2913 3644 5217 5041 4570 4748 2424 4195 4304
## [295] 4308 4381 4187 4687 3894 2659 3747  627 3331 3669 4068 4186 3974 4046
## [309] 3926 3649 4035 4205 4109 2933 3368 4067 3717 4486 4195 1817 3053 3392
## [323] 3663 3520 2765 1607 2566 1495 2792 3068 3071 3867 2914 3613 3727 3940
## [337] 3614 3485 3811 2594  705 3322 3620 3190 2743 3310 3523 3740 3709 3577
## [351] 2739 2431 3403 3750 2660 3068 2209 1011  754 1317 1162 2302 2423 2999
## [365] 2485 2294 1951 2236 2368 3272 4098 4521 3425 2376 3598 2177 4097 3214
## [379] 2493 2311 2298 2935 3376 3292 3163 1301 1977 2432 4339 4270 4075 3456
## [393] 4023 3243 3624 4509 4579 3761 4151 2832 2947 3784 4375 2802 3830 3831
## [407] 2169 1529 3422 3922 4169 3005 4154 4318 2689 3129 3777 4773 5062 3487
## [421] 2732 3389 4322 4363 1834 4990 3194 4066 3423 3333 3956 4916 5382 4569
## [435] 4118 4911 5298 5847 6312 6192 4378 7836 5892 6153 6093 6230 6871 8362
## [449] 3372 4996 5558 5102 5698 6133 5459 6235 6041 5936 6772 6436 6457 6460
## [463] 6857 5169 5585 5918 4862 5409 6398 7460 7132 6370 6691 4367 6565 7290
## [477] 6624 1027 3214 5633 6196 5026 6233 4220 6304 5572 5740 6169 6421 6296
## [491] 6883 6359 6273 5728 4717 6572 7030 7429 6118 2843 5115 7424 7384 7639
## [505] 8294 7129 4359 6073 5260 6770 6734 6536 6591 6043 5743 6855 7338 4127
## [519] 8120 7641 6998 7001 7055 7494 7736 7498 6598 6664 4972 7421 7363 7665
## [533] 7702 6978 5099 6825 6211 5905 5823 7458 6891 6779 7442 7335 6879 5463
## [547] 5687 5531 6227 6660 7403 6241 6207 4840 4672 6569 6290 7264 7446 7499
## [561] 6969 6031 6830 6786 5713 6591 5870 4459 7410 6966 7592 8173 6861 6904
## [575] 6685 6597 7105 7216 7580 7261 7175 6824 5464 7013 7273 7534 7286 5786
## [589] 6299 6544 6883 6784 7347 7605 7148 7865 4549 6530 7006 7375 7765 7582
## [603] 6053 5255 6917 7040 7697 7713 7350 6140 5810 6034 6864 7112 6203 7504
## [617] 5976 8227 7525 7767 7870 7804 8009 8714 7333 6869 4073 7591 7720 8167
## [631] 8395 7907 7436 7538 7733 7393 7415 8555 6889 6778 4639 7572 7328 8156
## [645] 7965 3510 5478 6392 7691 7570 7282 7109 6639 5875 7534 7461 7509 5424
## [659] 8090 6824 7058 7466 7693 7359 7444 7852 4459   22 1096 5566 5986 5847
## [673] 5138 5107 5259 5686 5035 5315 5992 6536 6852 6269 4094 5495 5445 5698
```

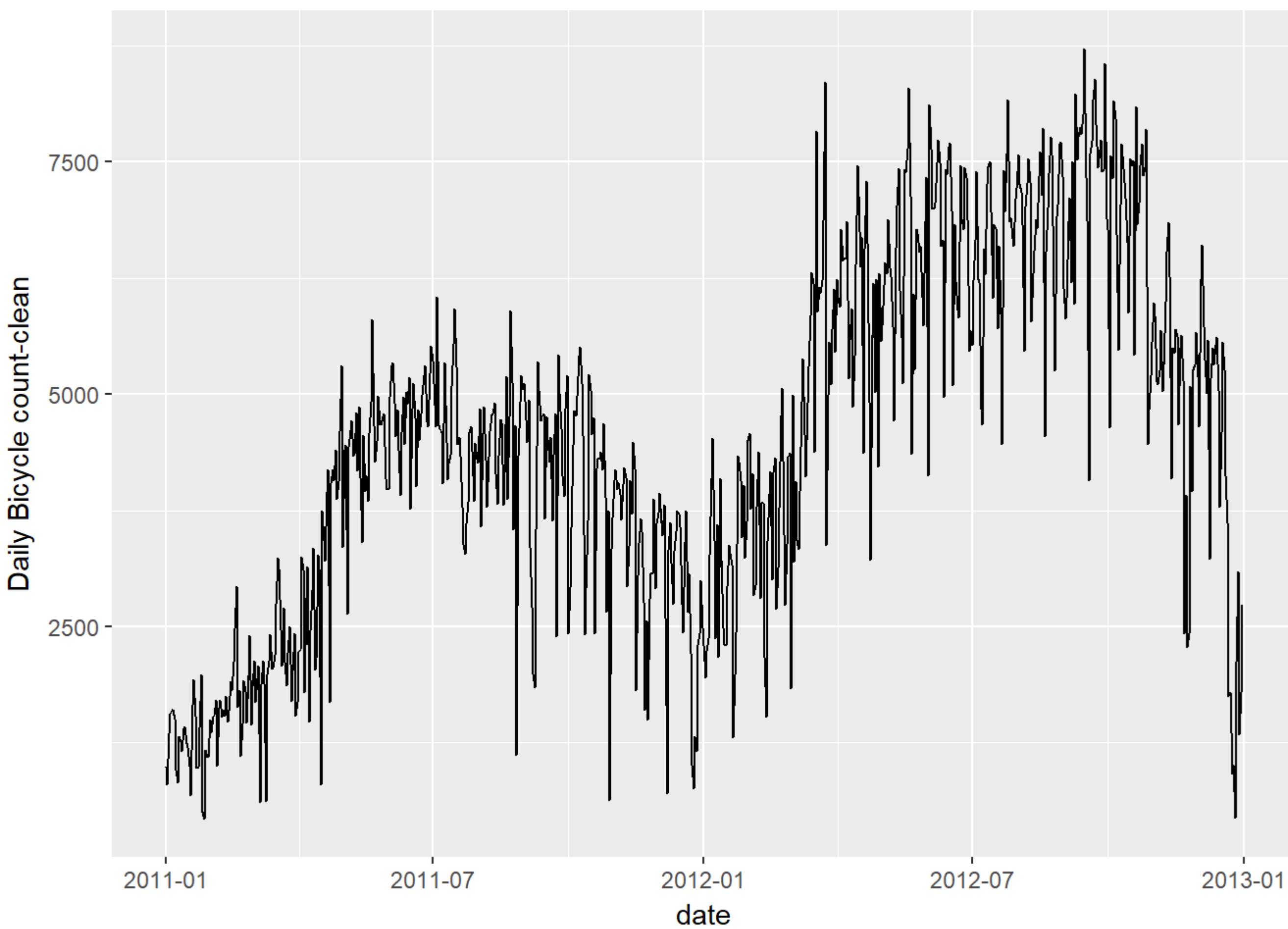
```
## [687] 5629 4669 5499 5634 5146 2425 3910 2277 2424 5087 3959 5260 5323 5668
## [701] 5191 4649 6234 6606 5729 5375 5008 5582 3228 5170 5501 5319 5532 5611
## [715] 5047 3786 4585 5557 5267 4128 3623 1749 1787  920 1013  441 2114 3095
## [729] 1341 1796 2729
```

```
str(count_ts)
```

```
##  Time-Series [1:731] from 1 to 731: 985 801 1349 1562 1600 1606 1510 959 822 1321 ...
```

```
daily_data$clean_cnt = tsclean(count_ts)

ggplot(data=daily_data,aes(x=date,y=clean_cnt))+geom_line()+
  ylab('Daily Bicycle count-clean')
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```

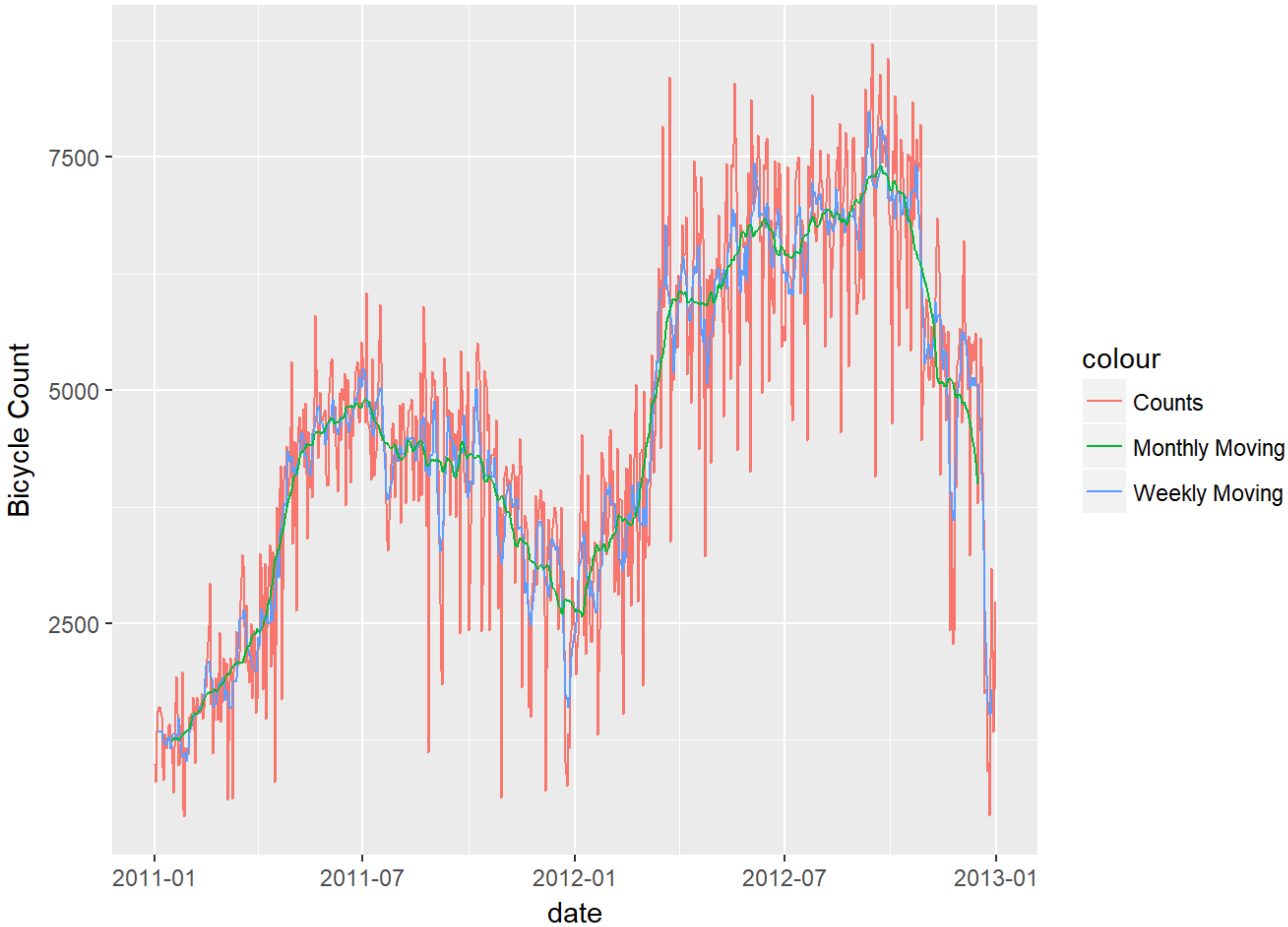


```
##the spikes have been removed

##tsclean is a package/function the at removes the outliers

##computing the moving average to get a smoother trend
##order=7 for weekly trend
daily_data$cnt_ma = ma(daily_data$clean_cnt,order=7)

##order=30 for a monthly trend
daily_data$cnt_ma30= ma(daily_data$clean_cnt,order=30)

###plotting the new data variables

ggplot()+
  geom_line(data=daily_data, aes(x=date,y=clean_cnt,colour="Counts"))+
  geom_line(data=daily_data, aes(x=date,y=cnt_ma,colour="Weekly Moving"))+
  geom_line(data=daily_data, aes(x=date,y=cnt_ma30,colour="Monthly Moving"))+
  ylab('Bicycle Count')
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```

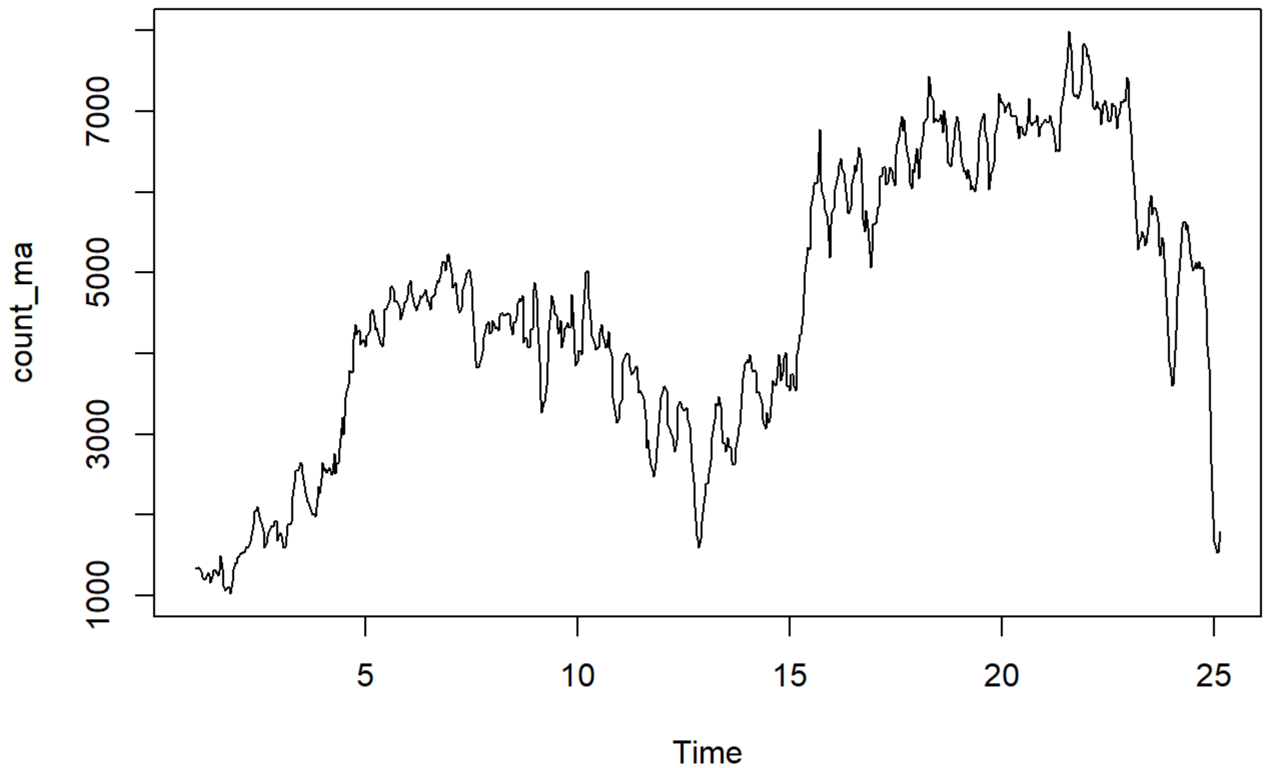
```
## Warning: Removed 6 rows containing missing values (geom_path).
```

```
## Warning: Removed 30 rows containing missing values (geom_path).
```

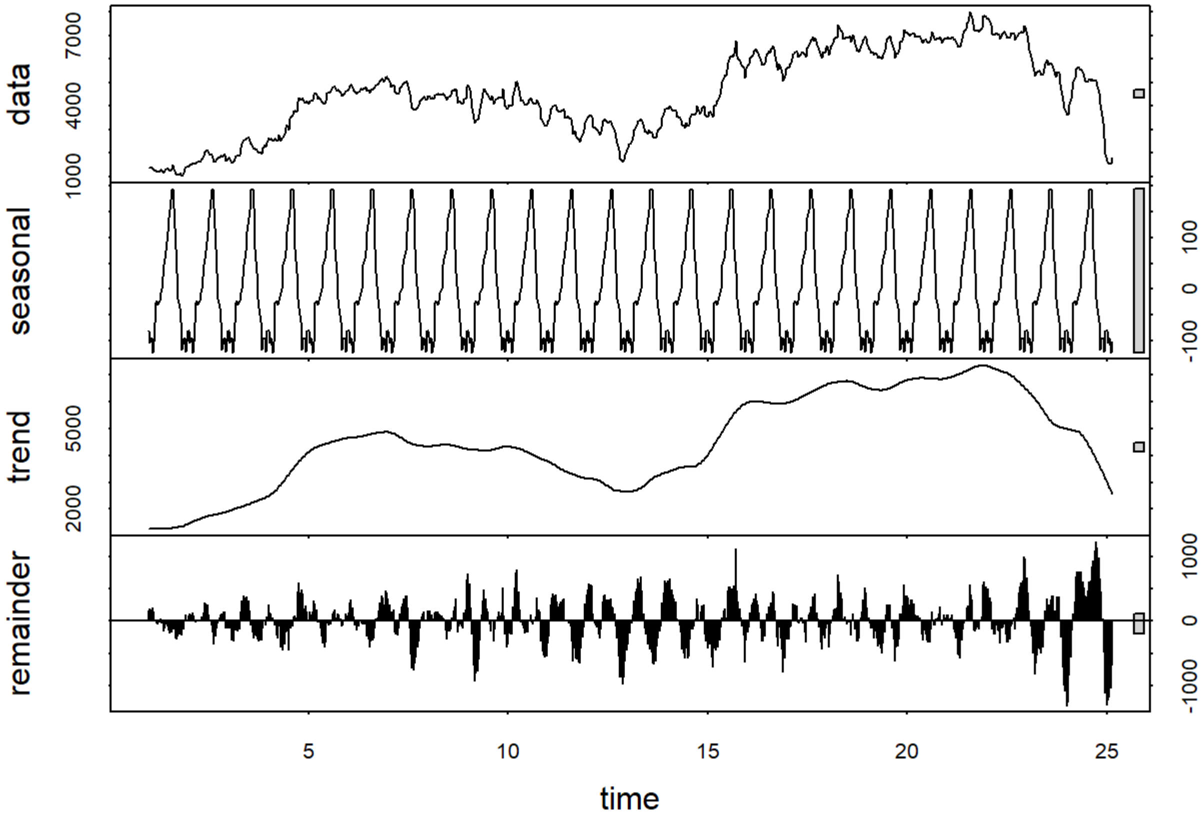


7500
5000
2500
Bicycle Count
2011-01
2011-07
2012-01
2012-07
2013-01
date
colour
Counts
Monthly Moving
Weekly Moving

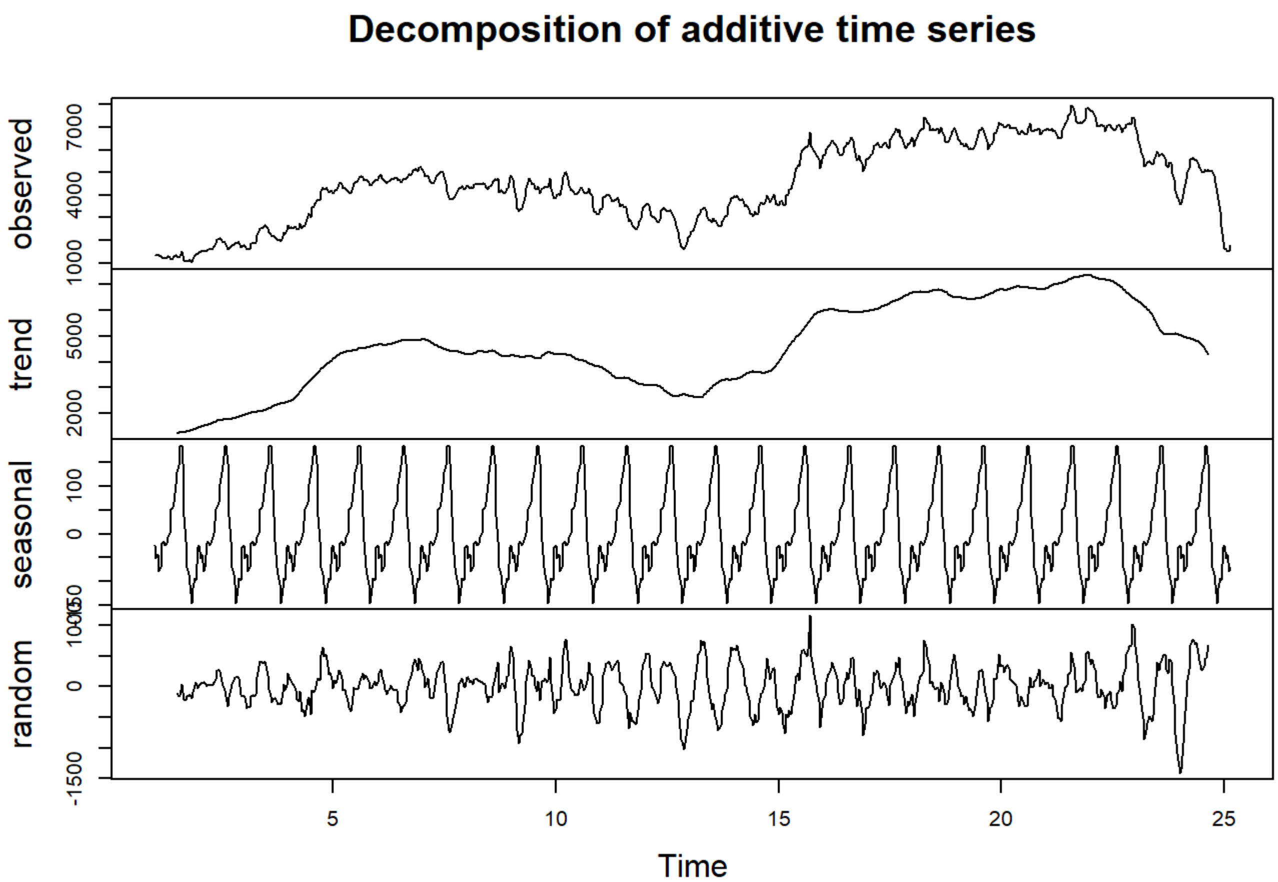
```
##decompose our the datat

##trying to understand seasonal component,trend component,cycle component
# Seasonal component refers to fluctuations in the data related to calendar cycles.
# For example, more people might be riding bikes in the summer and during warm weather, and less
 during colder months.
# Usually, seasonality is fixed at some number; for instance, quarter or month of the year.
#
# Trend component is the overall pattern of the series: Is the number of bikes rented increasing
 or decreasing over time?
#
# 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 i
s referred to as residual or error.

#The process of extracting these components is referred to as decomposition.


# First,, we calculate seasonal component of the data using stl(). STL is a flexible function fo
r decomposing and forecasting the series.
# It calculates the seasonal component of the series using smoothing,
# and adjusts the original series by subtracting seasonality in two simple lines:
#
#

count_ma = ts(na.omit(daily_data$cnt_ma),frequency = 30)
plot(count_ma)
```



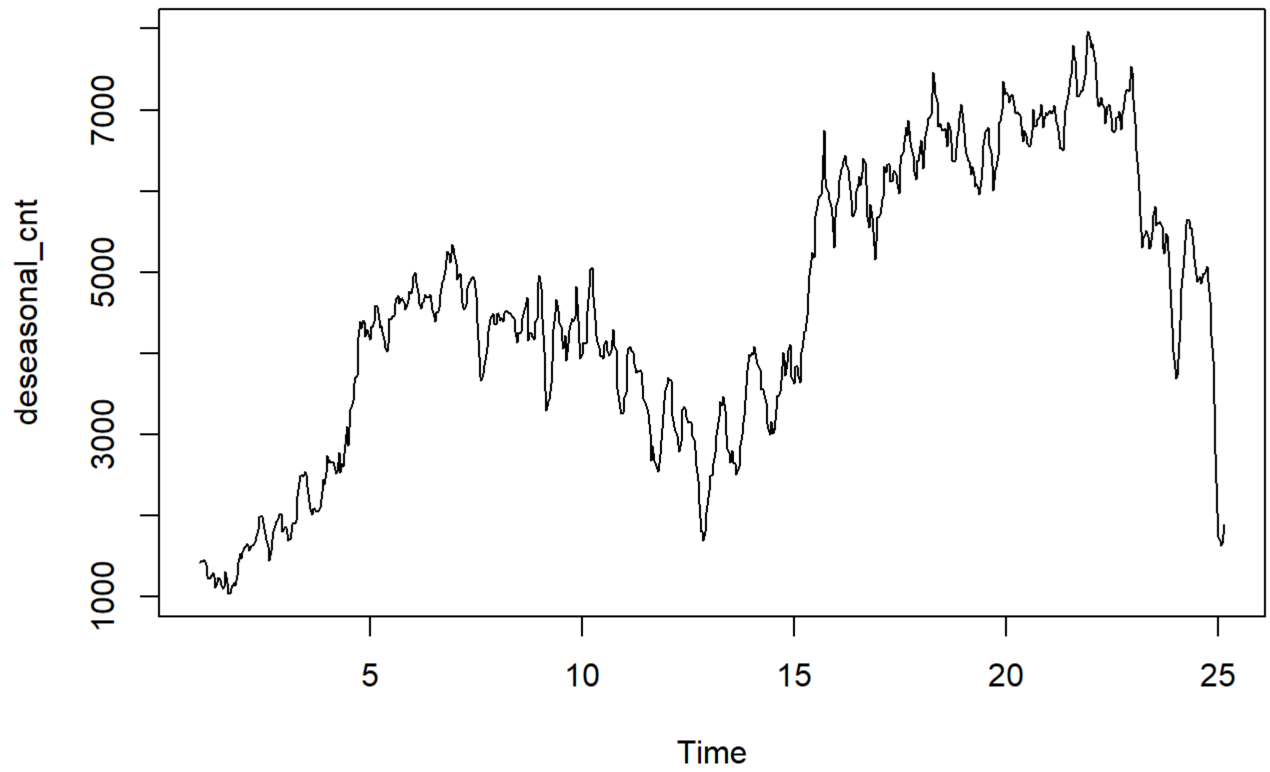

```
decomp = stl(count_ma,s.window="periodic")
plot(decomp)
```



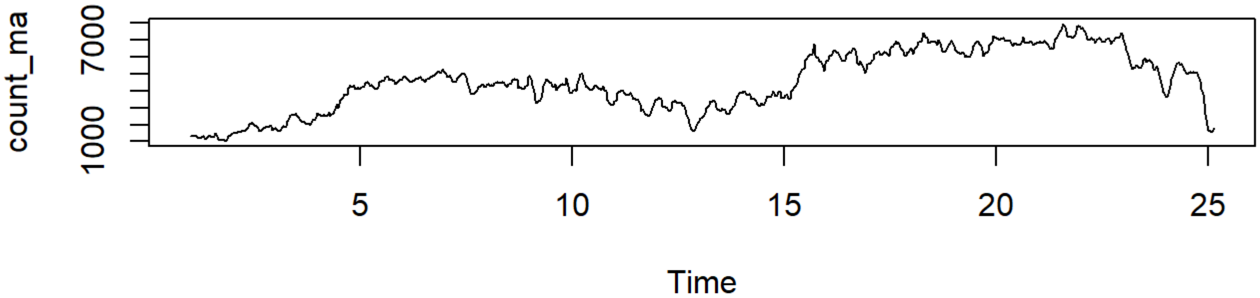

```
##alternative decompose function
decomp1=decompose(count_ma)
plot(decomp1)
```

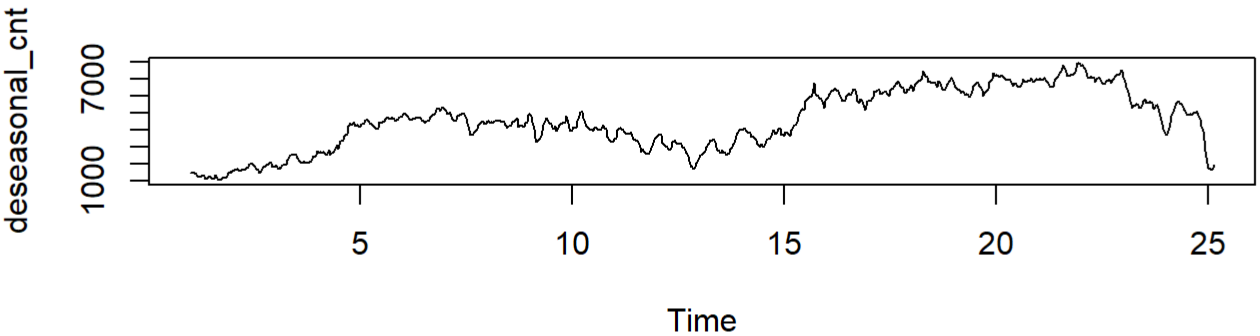



```
deseasonal_cnt<-seasadj(decomp)
plot(deseasonal_cnt)
```




```
##comparing the time series models
par(mfrow=c(2,1))
plot(count_ma)
plot(deseasonal_cnt)
```


count_ma
1000
7000
5
10
15
20
25
Time

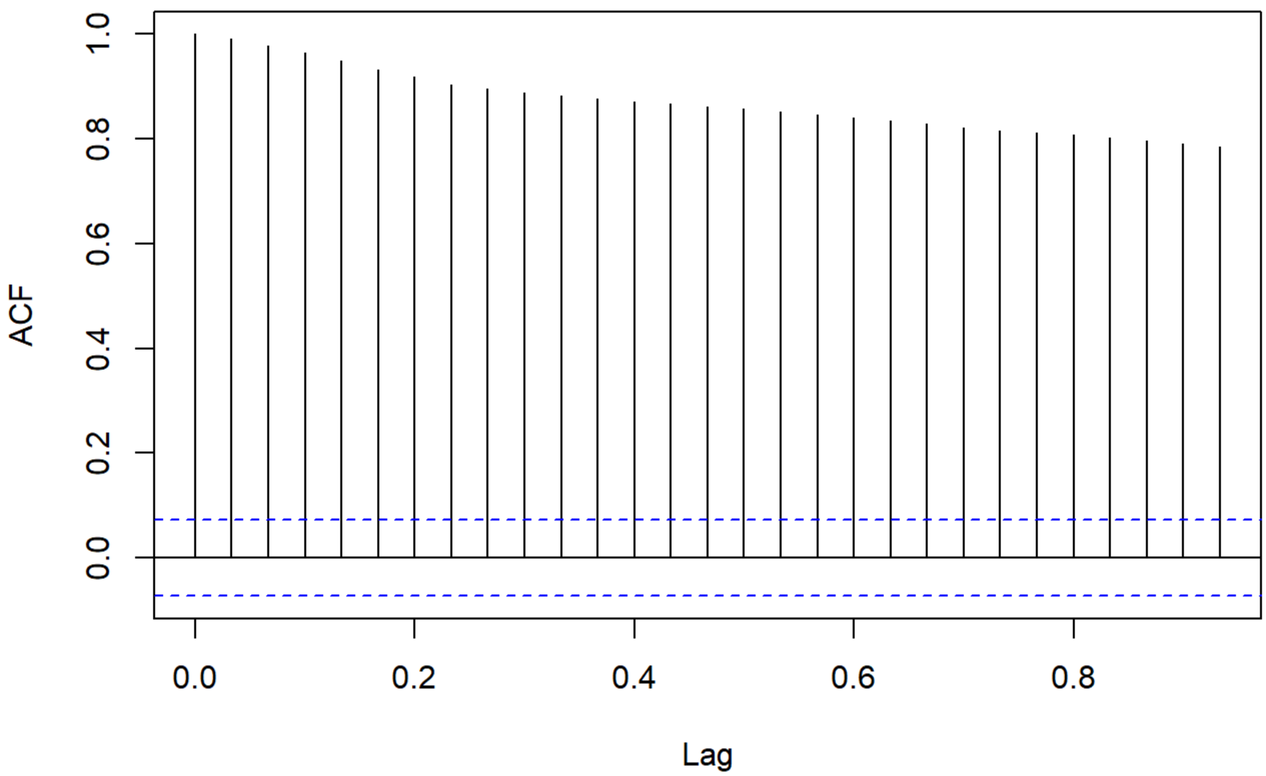

deseasonal_cnt
1000
7000
5
10
15
20
25
Time

```
##we dont observe any seaonality or trend in the data


# Note that stl() by default assumes additive model structure.
# Use allow.multiplicative.trend=TRUE to incorporate the multiplicative model.


##creating the a stationary time series


##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 invaria
nt. This assumption makes intuitive sense:
# Since ARIMA uses previous lags of series to model its behavior, modeling stable series with co
nsistent properties involves less uncertainty.
# An example of a stationary series, where data values oscillate with a steady variance around t
he mean of 1.
# For a non-stationary series;mean of this series will differ across different time windows.
#


##doing the augmented Dickey-Fuller(ADF) test

##we see that the time series is stationary

##the null hypothesis-Series is non stationary
##the alternate hypothesis-Series is stationary


# ADF procedure tests whether the change in Y can be explained by lagged value and a linear tren
d.
# If contribution of the lagged value to the change in Y is non-significant and there is a prese
nce
# of a trend component, the series is non-stationary and null hypothesis will not be rejected.

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

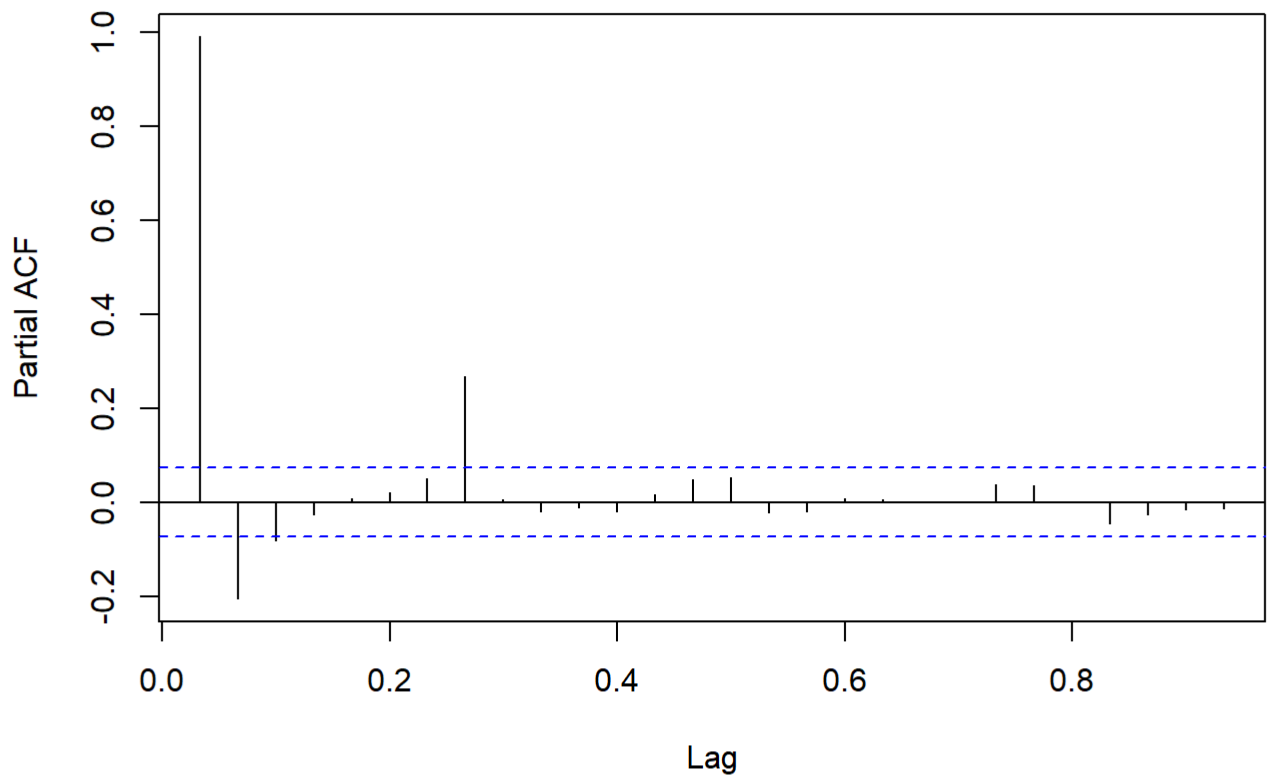
```
## Warning in adf.test(count_ma, alternative = "stationary"): p-value greater
## than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  count_ma
## Dickey-Fuller = -0.2557, Lag order = 8, p-value = 0.99
## alternative hypothesis: stationary
```

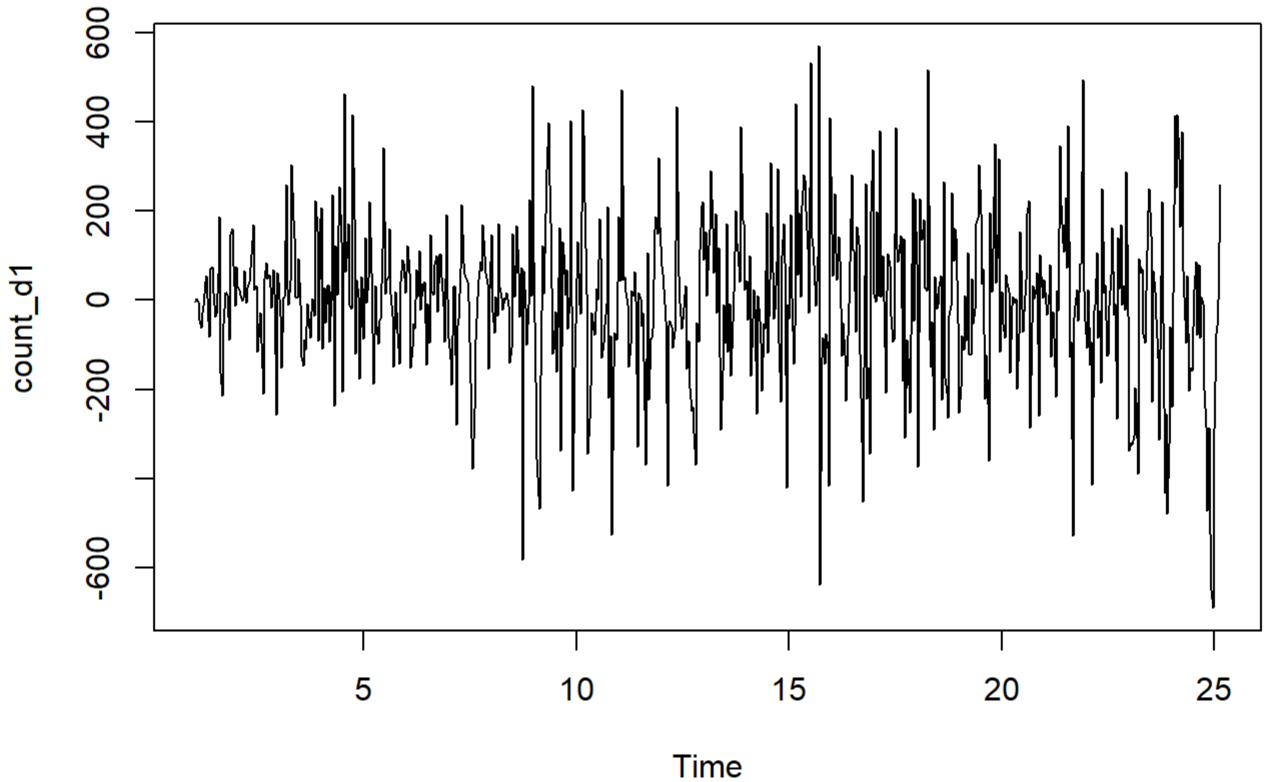
```
##the test shows that series is non stationary

##inorder to treat this series we make a diff factor(d order) for the model

par(mfrow=c(1,1))

acf(count_ma,main='')
```




```
pacf(count_ma,main='')
```




```
##THE pacf plot show p component for AR model
##The ACF shows the Q component for MA model

##trying with diff degree of 1

count_d1=diff(count_ma,difference=1)

plot(count_d1)
```

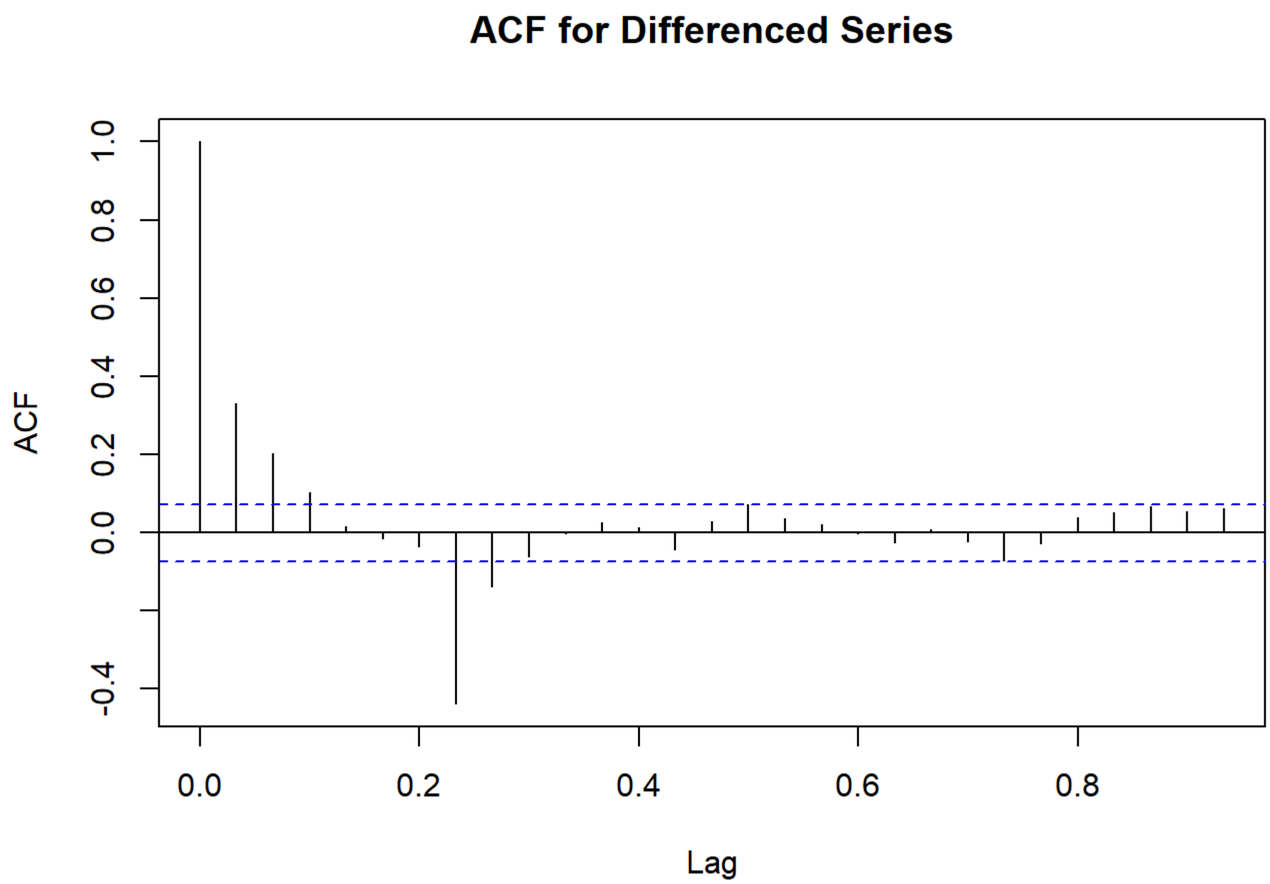



```
adf.test(count_d1,alternative = "stationary")
```

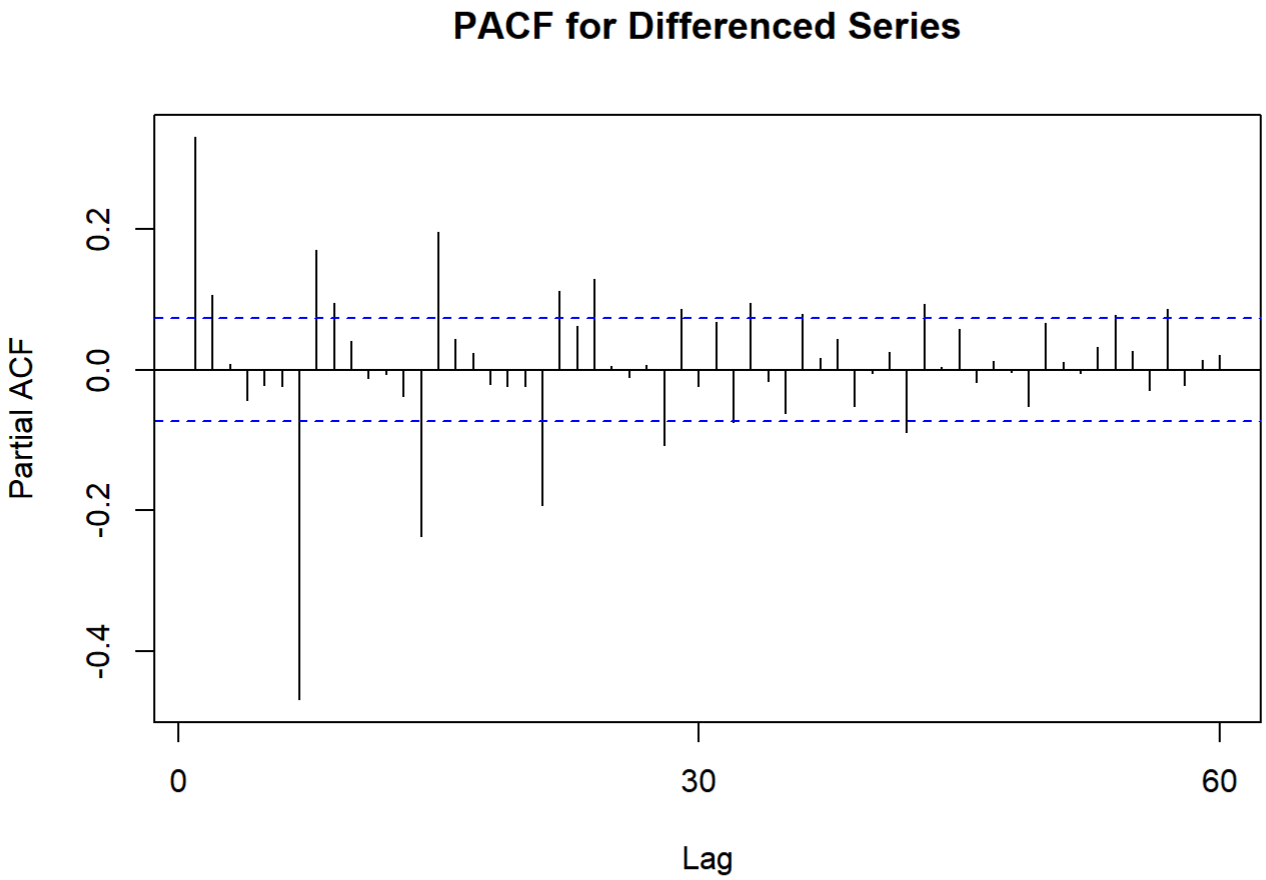
```
## Warning in adf.test(count_d1, alternative = "stationary"): p-value smaller
## than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  count_d1
## Dickey-Fuller = -9.9055, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```
##we have taken care of the non stationary element

##plot acf and pacf graphs again
par(mfrow=c(1,1))
acf(count_d1,main="ACF for Differenced Series")
```



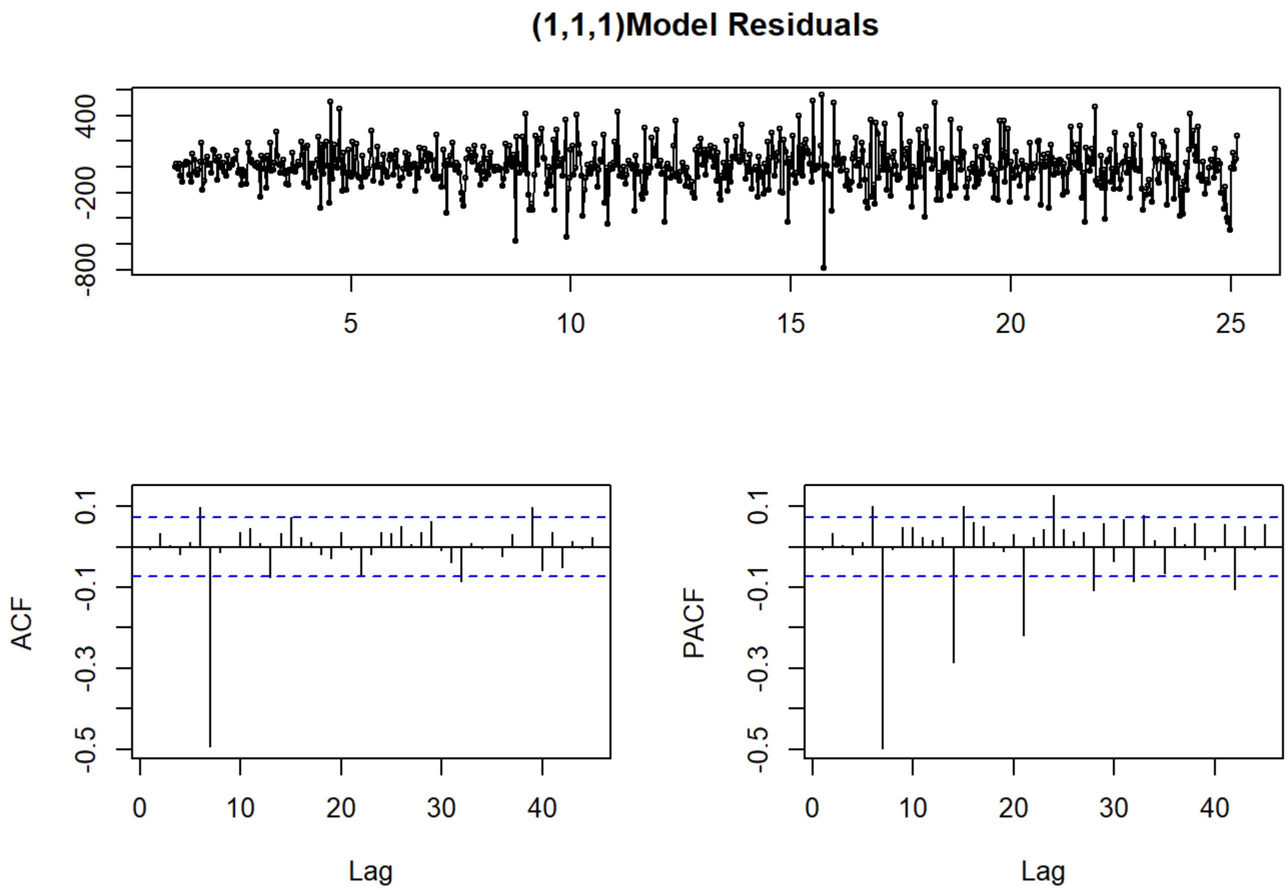

```
Pacf(count_d1,main="PACF for Differenced Series")
```



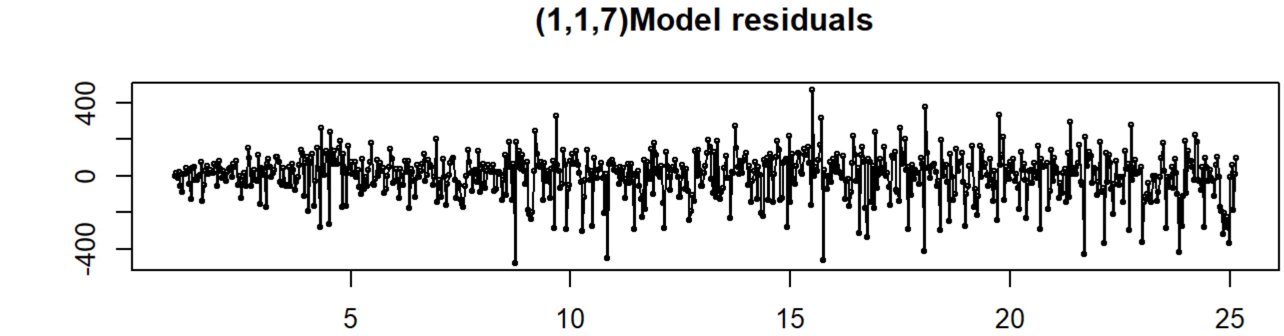

```
##from the plots we see that the value of p,q range from 1-7

##trying ARIMA(1,1,1)

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

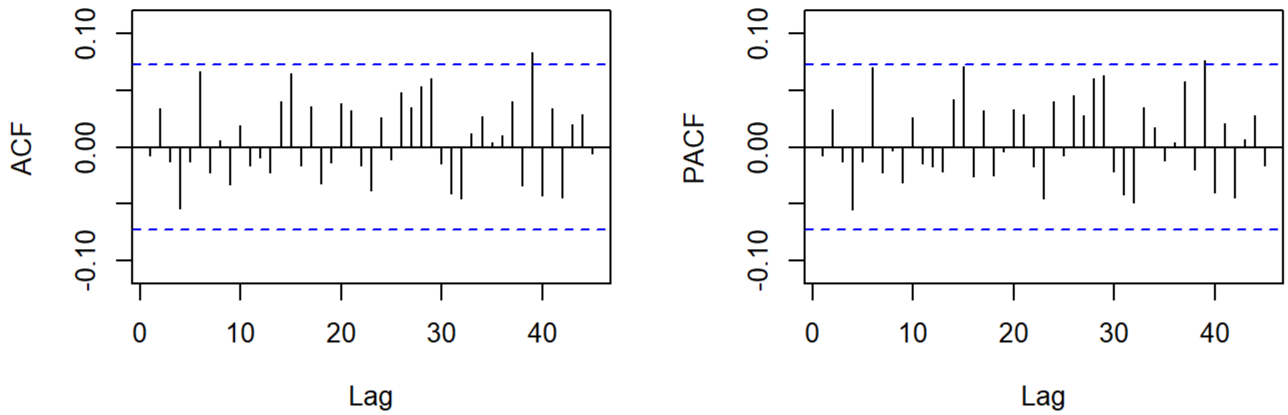
```
## Series: deseasonal_cnt
## ARIMA(1,1,1)
##
## Coefficients:
##          ar1      ma1
##       0.5510  -0.2496
## s.e.  0.0751   0.0849
##
## sigma^2 estimated as 26180:  log likelihood=-4708.91
## AIC=9423.82   AICc=9423.85   BIC=9437.57
```

```
tsdisplay(residuals(fit),lag.max=45,main='(1,1,1)Model Residuals')
```




```
###from the plots we still see that we have significant spikes in the data and repeating at 7


##trying ARIMA(1,1,7)

fit2<-arima(deseasonal_cnt,order=c(1,1,7))
tsdisplay(residuals(fit2),lag.max=45,main='(1,1,7)Model residuals')
```

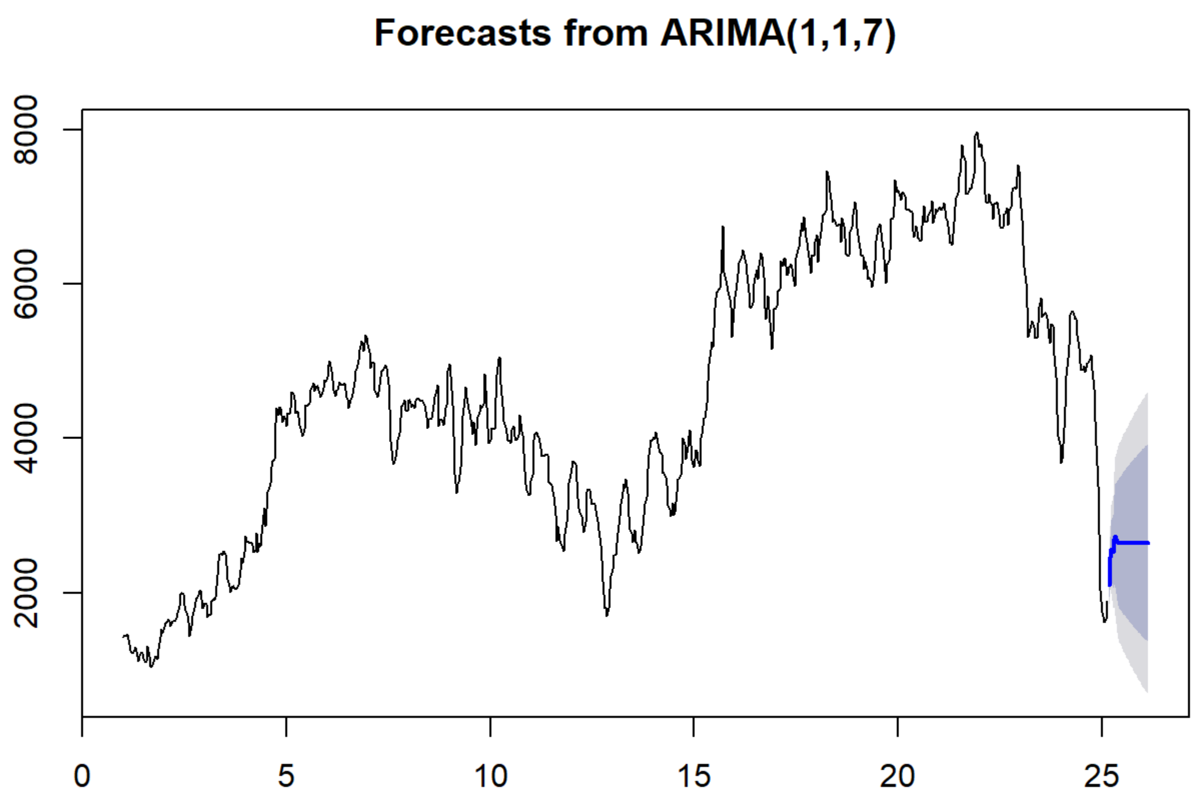




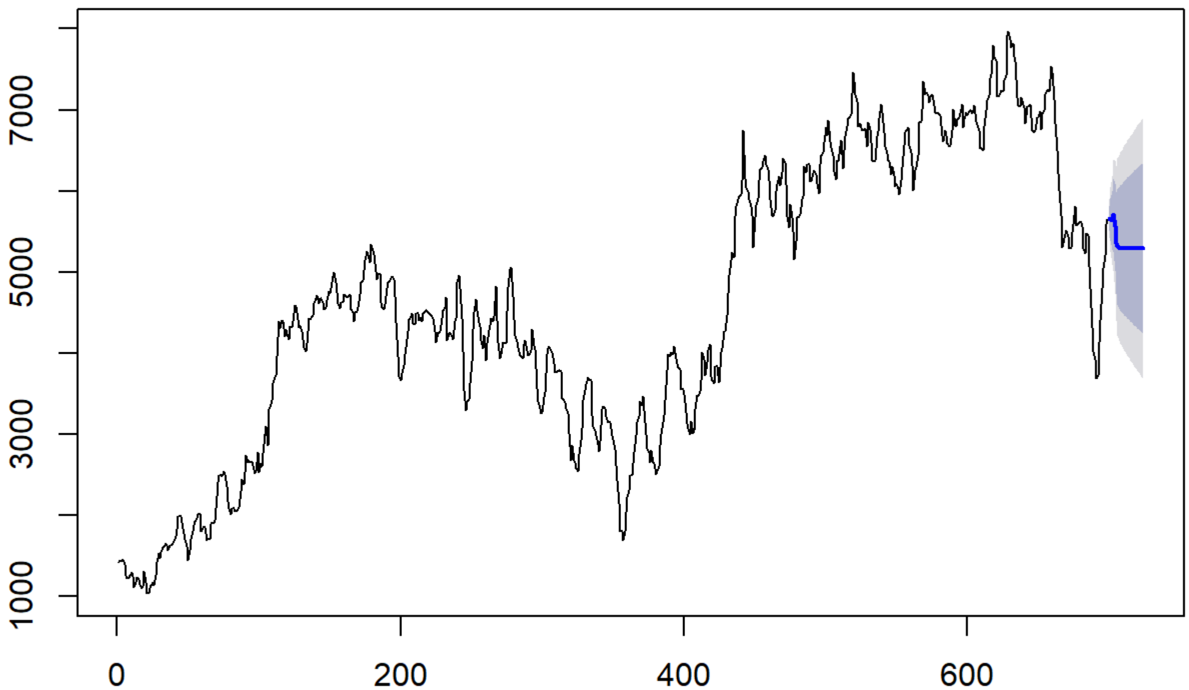

```
fit2
```

```
##
## Call:
## arima(x = deseasonal_cnt, order = c(1, 1, 7))
##
## Coefficients:
##          ar1     ma1     ma2     ma3     ma4     ma5     ma6      ma7
##       0.2803  0.1465  0.1524  0.1263  0.1225  0.1291  0.1471  -0.8353
## s.e.  0.0478  0.0289  0.0266  0.0261  0.0263  0.0257  0.0265   0.0285
##
## sigma^2 estimated as 14392:  log likelihood = -4503.28,  aic = 9024.56
```

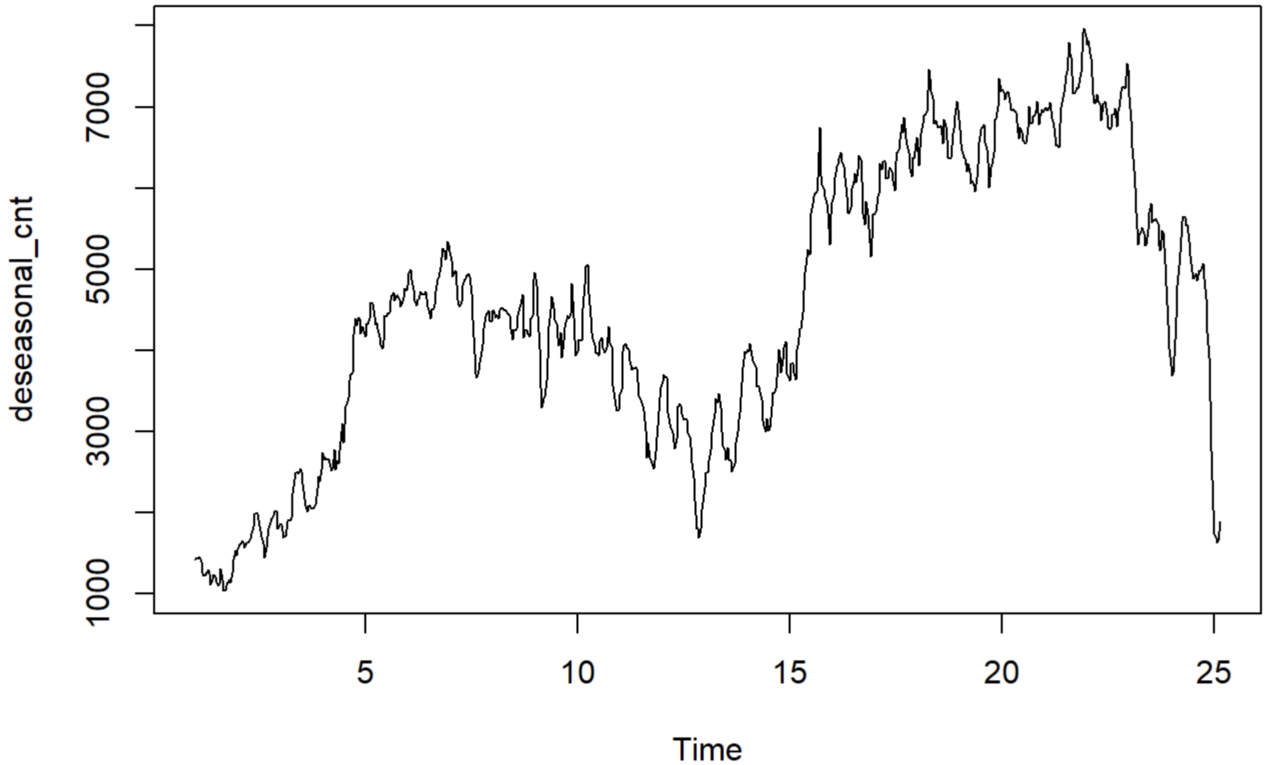
```
##we see that all the spikes in the data have been accounted for by the model


##forecasting the trend using the model

fcast<-forecast(fit2,h=30)
plot(fcast)
```




```
##we see from the forecast that the trend balances
## we can check with a hold out sample

hold<-window(ts(deseasonal_cnt),start=700)
fit_no_holdout = arima(ts(deseasonal_cnt[-c(700:725)]),order=c(1,1,7))

fcast_no_holdout<-forecast(fit_no_holdout,h=25)
plot(fcast_no_holdout,main=" ")
```




```
plot(deseasonal_cnt)
```




```
###we see a diff in the trend of the predicted data
```