2015 Discover Cup Model Contest

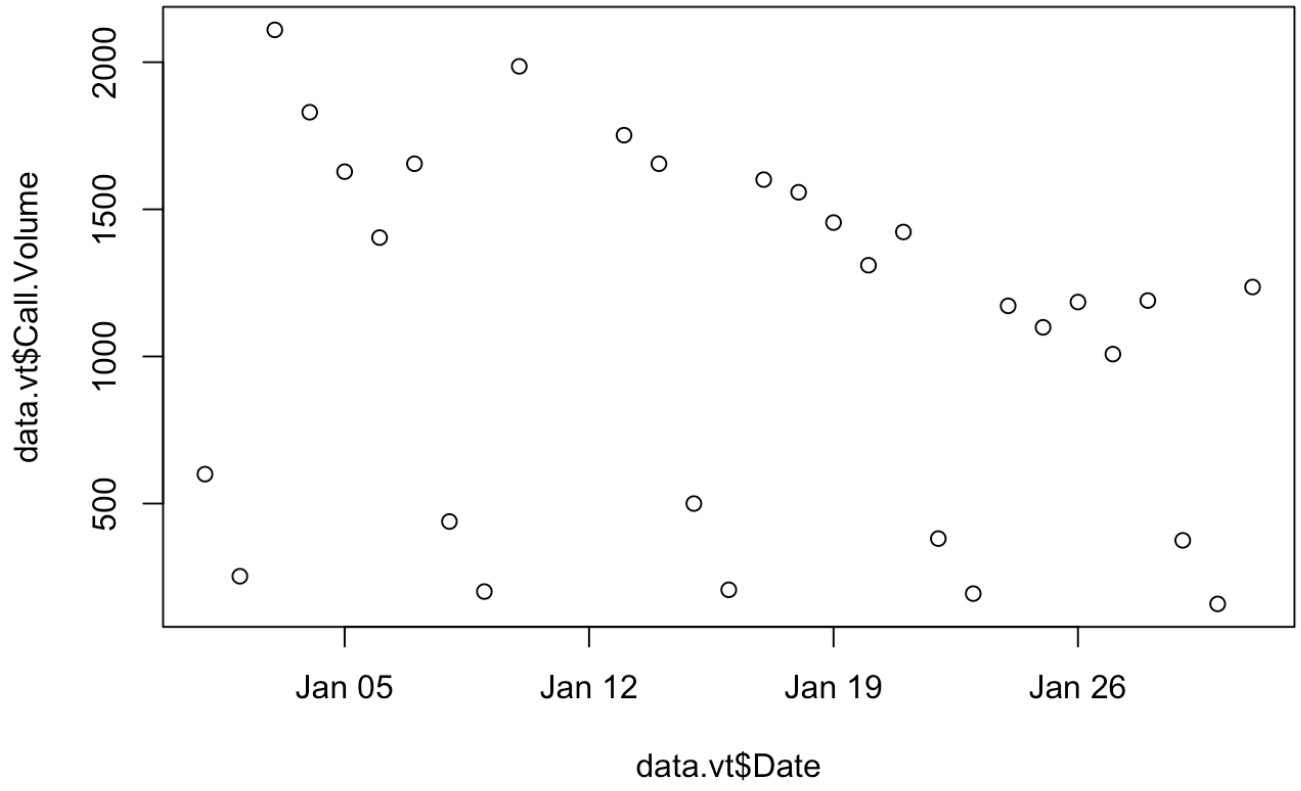
Hao YU

September 30, 2015

```
#***Data Processing***#

data<-read.table("~/Desktop/data.txt",sep="|",header=T)
data.od <- read.csv("~/Desktop/data.od.csv")
data.vt <- read.csv("~/Desktop/data.vt.csv")

data.vt$Date<-as.Date(data.vt$Date,"%m/%d/%Y")
data$Call_date<-as.Date(data$Call_date,"%Y-%m-%d")

summary(lm(data.vt$Call.Volume~data.vt$Handling_time))
```

```
##
## Call:
## lm(formula = data.vt$Call.Volume ~ data.vt$Handling_time)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -196.423  -62.280    2.916   35.575  215.062
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)           -33.566931  35.880133  -0.936    0.358
## data.vt$Handling_time   0.006559   0.000185  35.456   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 91.06 on 27 degrees of freedom
## Multiple R-squared:  0.979,  Adjusted R-squared:  0.9782
## F-statistic:  1257 on 1 and 27 DF,  p-value: < 2.2e-16
```

```
#linear relationship

plot(data.vt$Date,data.vt$Call.Volume)
```




```
#periodical time series
```

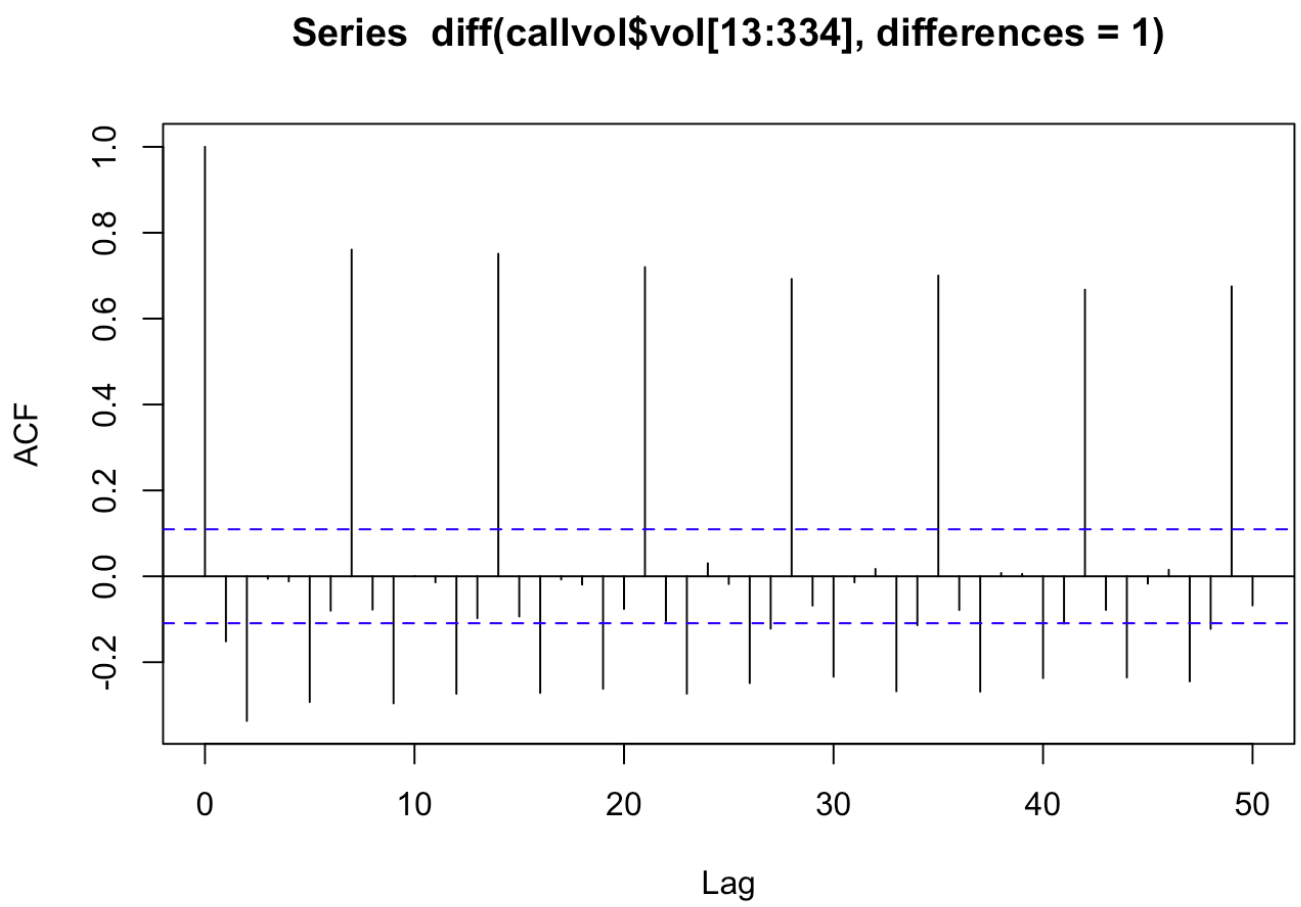
```
#***Question 1***#

Vol=c()
dat= as.Date(10592:10925,origin="1970-01-01")
dat<-as.data.frame.Date(dat)

for (i in 10592:10925){
  Vol[i-10591]=nrow(subset(data,data$Call_date==i))}

callvol<-cbind(dat,Vol)
#the daily call volume data

callvol[,3]<-weekdays(callvol$dat)
#add week days

callvol[1:31,4]<-c("Jan")
callvol[32:59,4]<-c("Feb")
callvol[60:90,4]<-c("Mar")
callvol[91:120,4]<-c("Apr")
callvol[121:151,4]<-c("May")
callvol[152:181,4]<-c("Jun")
callvol[182:212,4]<-c("Jul")
callvol[213:243,4]<-c("Aug")
callvol[244:273,4]<-c("Sep")
callvol[274:304,4]<-c("Oct")
callvol[305:334,4]<-c("Nov")
#add month

names(callvol)<-c("date","vol","day","month")

summary(aov(callvol$vol~callvol$day))
```

```
##              Df   Sum Sq  Mean Sq F value Pr(>F)
## callvol$day   6 96862026 16143671   126.8 <2e-16 ***
## Residuals   327 41620971   127281
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

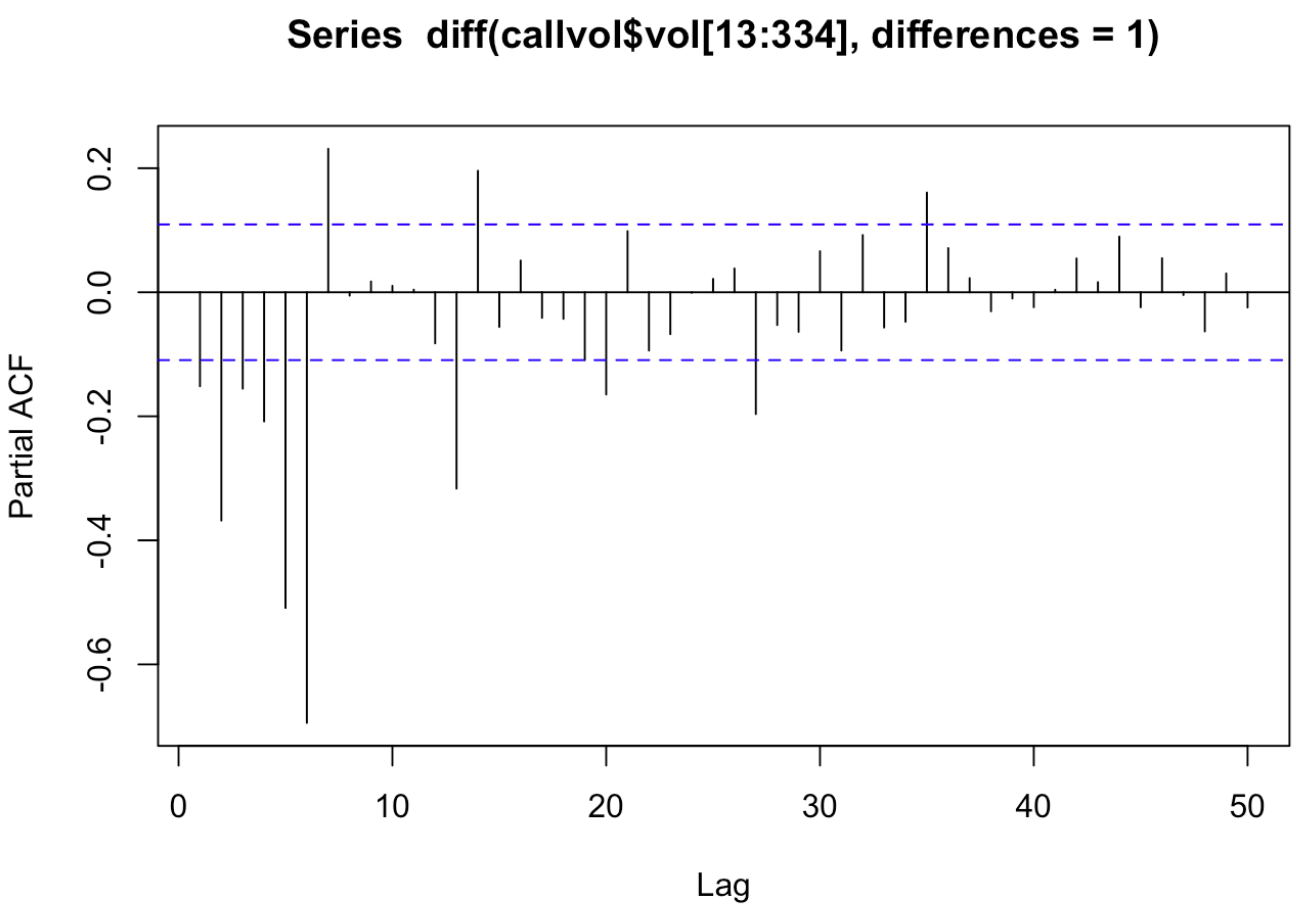
```
#ANOVA for week days

library(astsa)
library(forecast)
```

```
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Loading required package: timeDate
## This is forecast 6.1
##
##
## Attaching package: 'forecast'
##
## The following object is masked from 'package:astsa':
##
##     gas
```

```
#ARIMA model
acf(diff(callvol$vol[13:334],differences = 1),lag.max = 50)
```



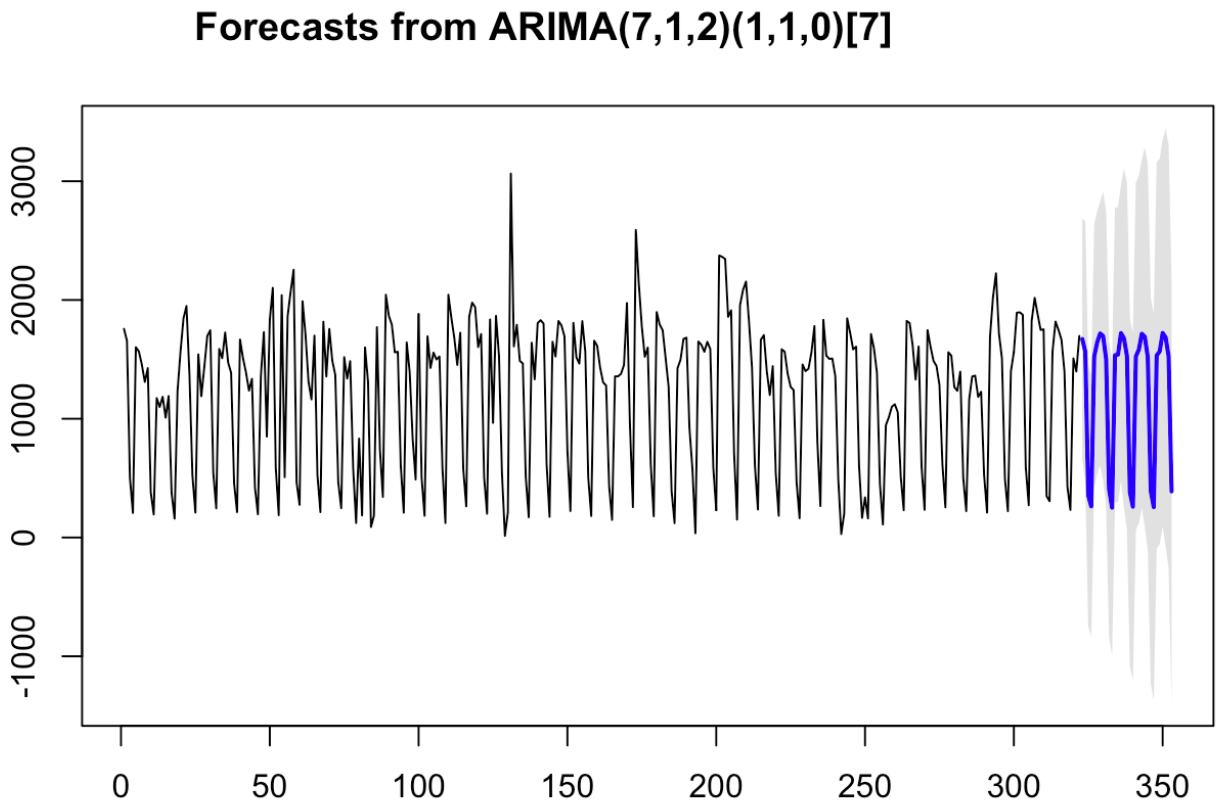

```
pacf(diff(callvol$vol[13:334],differences = 1),lag.max = 50)
```



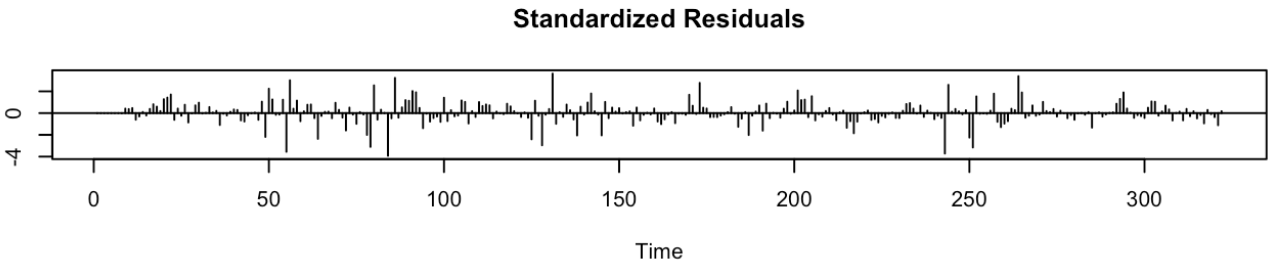

```
#acf and pacf show that p=7, d=1, q=2
vol.fit<-arima(callvol$vol[13:334],order=c(7,1,2),seasonal=list(order=c(1,1,0),
period=7))
AIC(vol.fit)
```

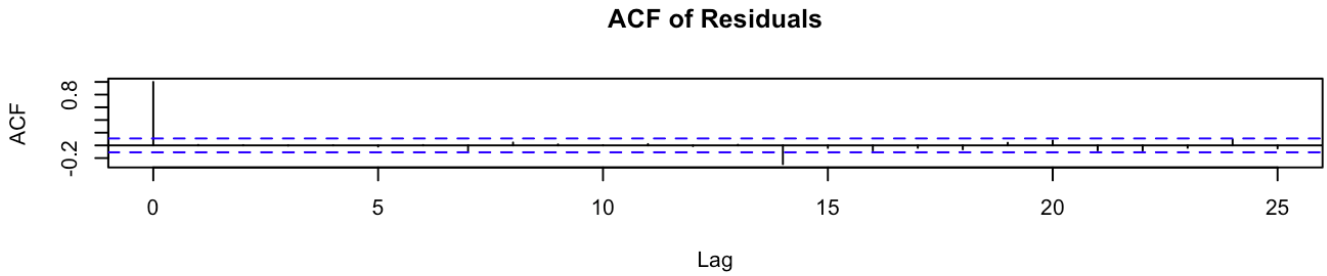
```
## [1] 4619.323
```

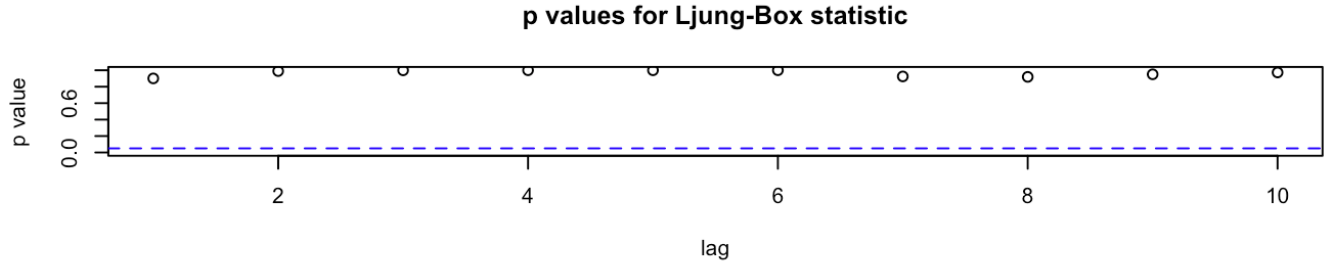
```
#AIC values as lowest
pre.vol<-forecast.Arima(vol.fit,h=31,level=c(99.5))
plot.forecast(pre.vol)
```




```
tsdiag(vol.fit)
```







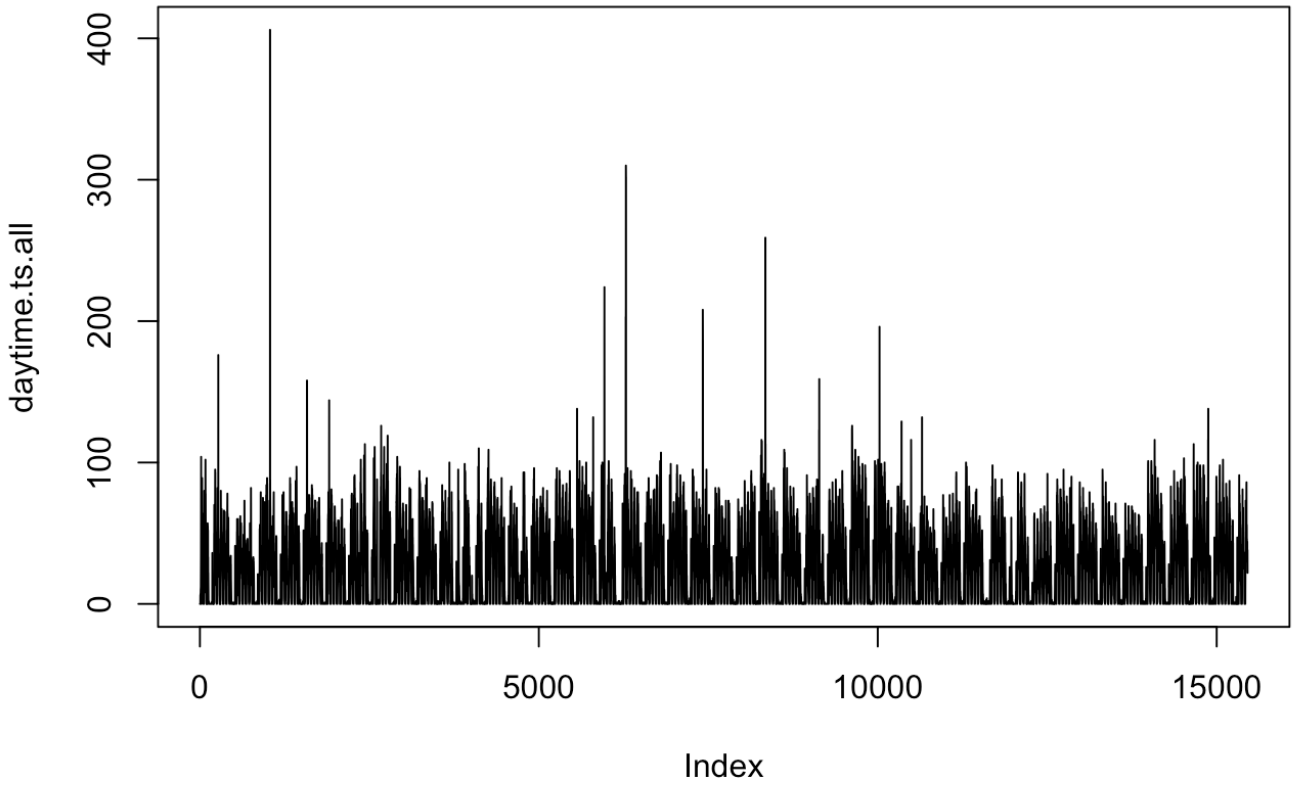
```
#p values show that all white noise

pre.vol
```

```
##     Point Forecast     Lo 99.5  Hi 99.5
## 323      1671.4812   656.54780 2686.415
## 324      1565.5289   471.21905 2659.839
## 325       354.0341  -743.46899 1451.537
## 326       262.5564  -838.77431 1363.887
## 327      1524.0218   419.62195 2628.422
## 328      1643.0427   537.19369 2748.892
## 329      1717.9512   608.33087 2827.572
## 330      1695.6343   480.77450 2910.494
## 331      1502.0976   262.54200 2741.653
## 332       404.0928  -835.87290 1644.059
## 333       251.2089  -989.52067 1491.939
## 334      1534.8112   293.74319 2775.879
## 335      1540.5394   299.19474 2781.884
## 336      1721.1586   479.59995 2962.717
## 337      1677.6365   249.45156 3105.822
## 338      1529.5671    72.07054 2987.064
## 339       379.2249 -1079.61361 1838.063
## 340       259.5627 -1200.66702 1719.792
## 341      1524.7232    63.26036 2986.186
## 342      1581.8893   119.84999 3043.929
## 343      1715.7564   252.25453 3179.258
## 344      1689.9180    95.20815 3284.628
## 345      1518.8177  -102.20159 3139.837
## 346       392.7621 -1229.09859 2014.623
## 347       255.9548 -1367.16553 1879.075
## 348      1533.0565   -90.84670 3156.960
## 349      1567.5044   -56.92716 3191.936
## 350      1721.8454    96.64818 3347.043
## 351      1685.1480   -70.60978 3440.906
## 352      1525.5246  -254.13455 3305.184
## 353       388.0946 -1392.62252 2168.812
```

```
#***Question 2***#

library(chron)

#for (a in 10592:10925){
#  data.sub<-subset(data,data$Call_date==a)
#  one.day<-chron(times=data.sub$IVR_entry)
#  for (b in 1:length(one.day))
#    for (c in 1:48) {
#      if (daytime[c,1] <= one.day[b] & daytime[c+1,1] > one.day[b] & is.na(day
time[c,1] <= one.day[b] & daytime[c+1,1] > one.day[b]) == FALSE)
#        daytime[c,a-10590] <- daytime[c,a-10590] + 1
#    }
#  daytime[48,a-10590]<-length(one.day)-sum(daytime[,a-10590])
#}
#consume too much time so directly load from local file
#divide volumes into each time period on each day

daytime <- read.csv("~/Desktop/daytime.csv")

names(daytime)[2:335]<-as.character(chron(as.numeric(names(daytime)[2:335])))
```

```
## Warning in inherits(dates., "dates"): NAs introduced by coercion
```

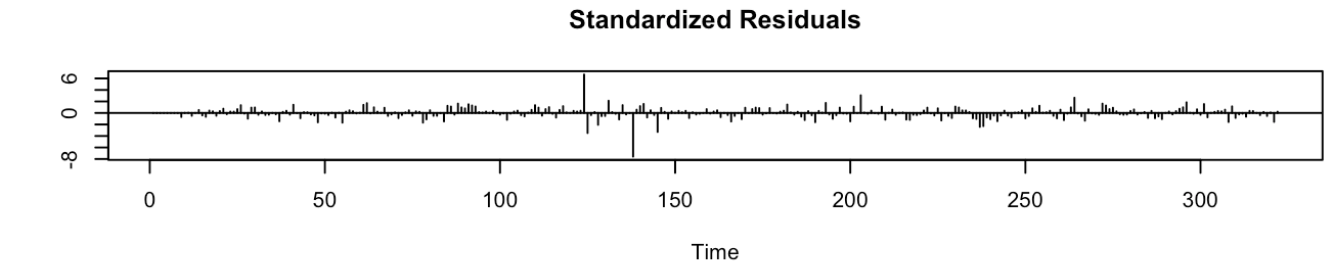
```
daytime.weekday<-names(daytime)[2:335]

daytime.ts.all<-c()

for (i in 14:335){
  daytime.ts.all<-c(daytime.ts.all,daytime[,i])
}
#all interval time series

plot(daytime.ts.all,type="l")
```

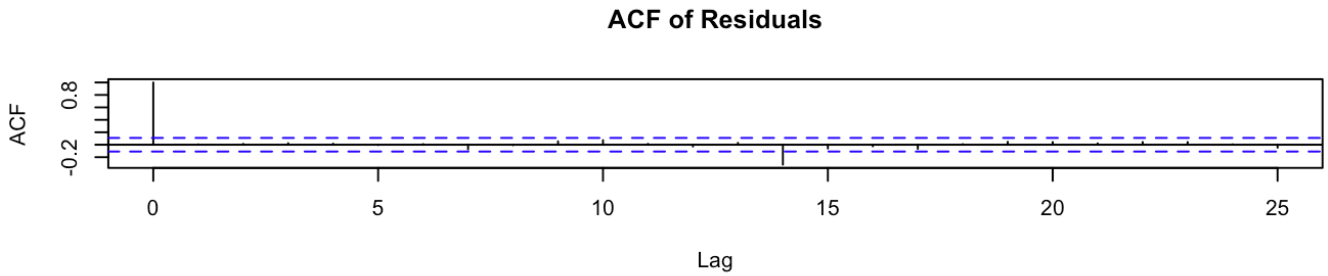


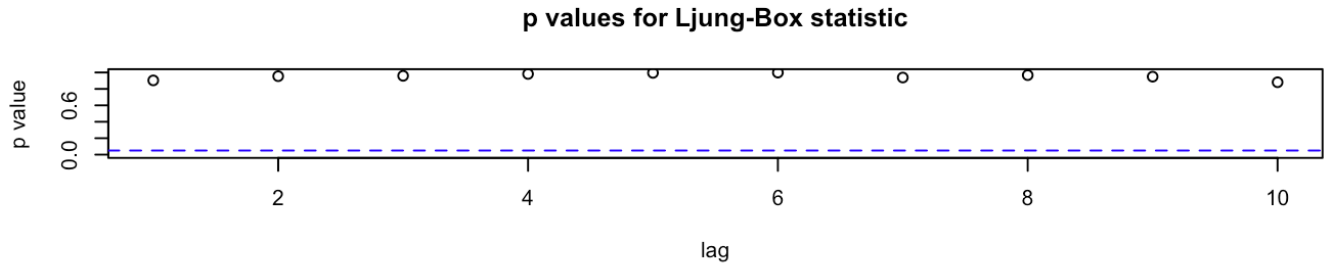

```
daytime.wd<-daytime
daytime.wd[49,2:335]<-weekdays(10592:10925)

#ARIMA model loop
forecast.time<-matrix(nrow = 48,ncol = 31)
forecast.time[1:48,1:31]<-0
for (i in 1:48){
  time.fit<-arima(as.numeric(daytime.wd[i,14:335]),order=c(7,1,1),seasonal=lis
t(order=c(1,1,0), period=7))
  forecast.time.fit<-(forecast.Arima(time.fit,h=31))
  forecast.time[i,1:31]<-forecast.time.fit$mean[1:31]}

AIC(time.fit)
```

```
## [1] 2317.608
```

```
tsdiag(time.fit)
```








```
forecast.time.round<-round(forecast.time)

for (i in 1:48){
  for (j in 1:31){
    if (forecast.time.round[i,j]<0){
      forecast.time.round[i,j]=0}}}

#point 0 to all negative values
forecast.time.round
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
##  [1,]    2    3    2    0    1    5    3    1    3     1     0     0     4
##  [2,]    3    1    0    1    0    1    2    2    1     0     1     0     1
##  [3,]    0    0    0    0    1    0    1    0    0     0     0     0     0
##  [4,]    2    0    1    0    2    0    2    2    0     1     0     1     0
##  [5,]    0    1    1    0    1    1    0    0    1     1     0     0     1
##  [6,]    0    0    0    1    0    1    0    0    0     0     1     0     1
##  [7,]    1    1    0    0    0    0    0    1    0     0     0     0     0
##  [8,]    1    0    1    0    0    0    0    0    0     0     0     0     0
##  [9,]    1    0    0    0    0    1    0    1    0     0     0     0     0
## [10,]    0    1    1    0    0    0    0    0    0     0     0     0     0
## [11,]    0    0    0    0    0    0    0    0    0     0     0     0     0
## [12,]    2    1    0    0    1    1    0    2    1     0     0     1     1
## [13,]    3    0    0    0    0    1    1    2    0     0     0     0     0
## [14,]    3    3    1    0    3    3    3    4    4     1     0     3     4
## [15,]   18    9   14    0   18   13   11   18   10    14     0    19    13
## [16,]   17   15   16    0   20   24   18   17   15    17     0    19    26
## [17,]   32   29   21    1   31   31   34   27   25    18     0    30    35
## [18,]   46   34   24    0   47   50   34   42   31    20     0    44    42
## [19,]   57   57   32    0   60   56   69   50   53    25     0    58    48
## [20,]   87   62   28    0   87   74   80   85   66    34     0    90    67
## [21,]   72   78   39    1   83   77   87   73   84    44     2    81    73
## [22,]   81   61   41    1   77   91   77   78   61    50     2    75    87
## [23,]   73   54   34    4   73   70   65   74   61    42     5    69    64
## [24,]   69   63   35    0   65   66   74   67   64    38     0    65    58
## [25,]   68   53   30    2   59   55   58   68   57    36     1    67    52
## [26,]   66   45   22    0   49   51   54   71   44    25     0    53    47
## [27,]   61   58   12    1   59   58   59   59   56    10     1    59    56
## [28,]   62   55    1    0   59   60   62   67   53     1     0    59    55
## [29,]   60   57    0    0   60   57   67   61   58     0     0    60    52
## [30,]   69   58    1    2   64   69   74   71   59     1     1    67    69
## [31,]   77   60    1    3   60   58   79   72   60     1     2    58    54
## [32,]   73   64    5    2   68   68   73   72   62     3     1    67    60
## [33,]   47   50    1    0   55   40   48   48   48     0     0    50    36
## [34,]   51   50    1    0   55   45   54   58   43     1     0    55    46
## [35,]   46   32    1    0   36   37   45   52   27     0     0    30    31
## [36,]   38   40    0    0   40   44   52   35   41     0     0    42    37
## [37,]   41   35    0    7   32   37   52   42   35     0     6    30    34
## [38,]   38   26    0   20   26   38   40   40   25     0    15    26    37
## [39,]   34   29    0   43   30   39   39   31   27     0    42    27    35
## [40,]   37   27    1   31   27   46   40   36   27     1    30    28    42
## [41,]   22   21    0   12   22   34   31   19   16     0    12    23    29
## [42,]   28   71    2   24   22   32   37   28   46     1    22    21    33
## [43,]   40   52    2   15   29   39   36   35   37     1    13    28    38
## [44,]   31   86    1   26   27   33   40   34   64     1    27    25    29
## [45,]   30   46    1   19   28   32   34   30   46     1    18    28    31
## [46,]   32   29    1   22   20   37   39   33   27     0    21    20    37
## [47,]   32   30    1   16   17   35   33   29   29     0    14    16    34
## [48,]   20   16    1    9   12   19   22   21   15     0     9    12    17
##       [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
##  [1,]     3     2     3     1     0     1     4     3     1     3     1
##  [2,]     2     2     1     0     1     0     1     2     2     1     0
##  [3,]     0     0     0     0     0     0     0     0     0     0     0
```

```
##  [4,]     1     2     0     1     0     1     0     1     2     0     1
##  [5,]     1     0     1     1     0     0     1     1     0     1     1
##  [6,]     0     0     0     0     1     0     1     0     0     0     0
##  [7,]     1     1     1     0     0     0     0     1     1     0     0
##  [8,]     0     0     0     0     0     0     0     0     0     0     0
##  [9,]     1     1     0     0     0     0     0     1     1     0     0
## [10,]     0     0     0     0     0     0     0     0     0     0     0
## [11,]     0     0     0     0     0     0     0     0     0     0     0
## [12,]     0     2     1     0     0     1     1     0     2     1     0
## [13,]     1     3     0     0     0     0     0     1     2     0     0
## [14,]     2     3     3     1     0     3     4     2     4     4     1
## [15,]    11    18    10    14     0    19    13    11    18    10    14
## [16,]    18    17    15    16     0    19    25    18    17    15    16
## [17,]    35    29    27    19     0    30    34    34    28    26    18
## [18,]    32    39    27    17     0    41    41    28    36    24    14
## [19,]    58    47    48    21     0    52    44    55    41    43    15
## [20,]    78    87    65    32     0    89    69    79    86    65    33
## [21,]    80    72    81    43     1    82    75    83    73    83    44
## [22,]    73    79    61    47     2    76    89    75    80    62    49
## [23,]    66    74    58    39     5    71    66    66    75    60    41
## [24,]    72    68    64    37     0    65    61    73    68    64    38
## [25,]    56    68    56    34     2    64    53    57    68    56    35
## [26,]    52    69    45    24     0    51    49    53    70    44    24
## [27,]    56    60    57    11     1    59    57    57    60    56    11
## [28,]    60    65    54     1     0    58    57    61    66    54     1
## [29,]    66    61    58     0     0    60    53    66    61    58     0
## [30,]    74    71    59     1     1    66    69    74    71    59     1
## [31,]    81    73    60     1     2    59    55    80    72    60     1
## [32,]    78    73    62     4     2    67    64    76    73    62     4
## [33,]    47    47    47     0     0    51    37    47    47    47     0
## [34,]    51    54    45     1     0    55    45    52    56    44     0
## [35,]    46    49    29     0     0    32    33    45    50    28     0
## [36,]    53    36    40     0     0    41    39    53    36    40     0
## [37,]    52    41    35     0     6    30    35    52    41    35     0
## [38,]    39    39    25     0    17    26    37    40    39    25     0
## [39,]    39    29    25     0    39    25    34    36    27    23     0
## [40,]    40    36    27     0    30    28    43    40    36    27     0
## [41,]    30    16    13     0     8    19    28    27    14    11     0
## [42,]    39    28    57     2    23    22    33    38    28    53     1
## [43,]    39    38    43     2    14    29    39    39    37    41     2
## [44,]    36    33    73     1    27    26    31    38    34    70     1
## [45,]    33    30    46     1    19    28    31    34    30    46     1
## [46,]    38    33    28     1    21    20    37    39    33    28     1
## [47,]    35    30    30     1    15    17    34    34    30    30     1
## [48,]    22    20    15     1     9    12    17    22    20    15     1
##       [,25] [,26] [,27] [,28] [,29] [,30] [,31]
##  [1,]     0     1     4     3     2     3     1
##  [2,]     1     0     1     2     2     1     0
##  [3,]     0     0     0     0     0     0     0
##  [4,]     0     1     0     1     2     0     1
##  [5,]     0     0     1     1     0     1     1
##  [6,]     1     0     1     0     0     0     0
##  [7,]     0     0     0     1     1     0     0
##  [8,]     0     0     0     0     0     0     0
```

```
##  [9,]     0     0     0     1     1     0     0
## [10,]     0     0     0     0     0     0     0
## [11,]     0     0     0     0     0     0     0
## [12,]     0     1     1     0     2     1     0
## [13,]     0     0     0     1     2     0     0
## [14,]     0     3     4     2     3     3     1
## [15,]     0    19    13    11    18    10    14
## [16,]     0    19    25    18    17    15    16
## [17,]     0    30    34    34    28    26    18
## [18,]     0    38    37    26    33    21    11
## [19,]     0    48    38    49    36    39    11
## [20,]     0    90    69    79    87    66    33
## [21,]     2    82    75    82    73    83    44
## [22,]     3    76    89    75    80    62    49
## [23,]     5    70    66    66    75    60    40
## [24,]     0    65    60    73    68    64    38
## [25,]     2    65    53    56    68    56    35
## [26,]     0    52    48    53    70    45    24
## [27,]     1    59    57    57    60    57    11
## [28,]     0    59    56    61    66    54     1
## [29,]     0    60    53    66    61    58     0
## [30,]     1    66    69    74    71    59     1
## [31,]     2    58    55    80    72    60     1
## [32,]     1    68    62    77    73    62     4
## [33,]     0    50    36    46    46    47     0
## [34,]     0    55    45    51    55    44     0
## [35,]     0    31    32    45    49    28     0
## [36,]     0    41    38    53    36    40     0
## [37,]     6    30    34    52    41    35     0
## [38,]    16    26    37    40    39    25     0
## [39,]    37    23    31    35    25    21     0
## [40,]    30    28    43    40    36    27     0
## [41,]     6    17    25    24    11     8     0
## [42,]    23    22    33    39    28    54     2
## [43,]    14    29    39    39    38    42     2
## [44,]    27    26    31    37    34    71     2
## [45,]    19    28    31    34    30    46     1
## [46,]    21    20    37    39    33    28     1
## [47,]    15    17    34    34    30    30     1
## [48,]     9    12    17    22    20    15     1
```

```
forecast.time.round[20,1:31]
```

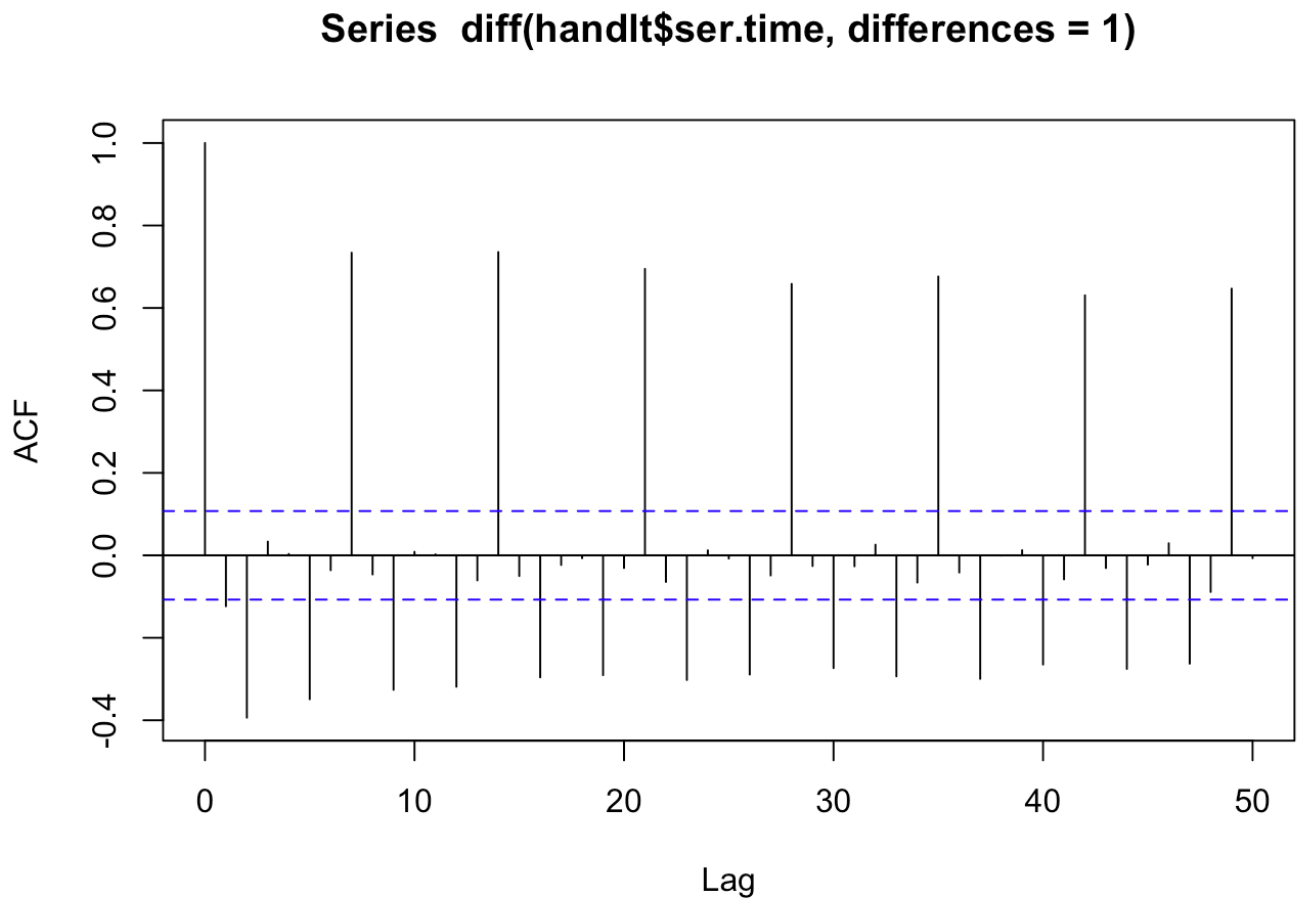
```
##  [1] 87 62 28  0 87 74 80 85 66 34  0 90 67 78 87 65 32  0 89 69 79 86 65
## [24] 33  0 90 69 79 87 66 33
```

```
#call volumes on 09:30:00 from 1999-12-01 to 1999-12-31


forecast.time.round[1:48,15]
```

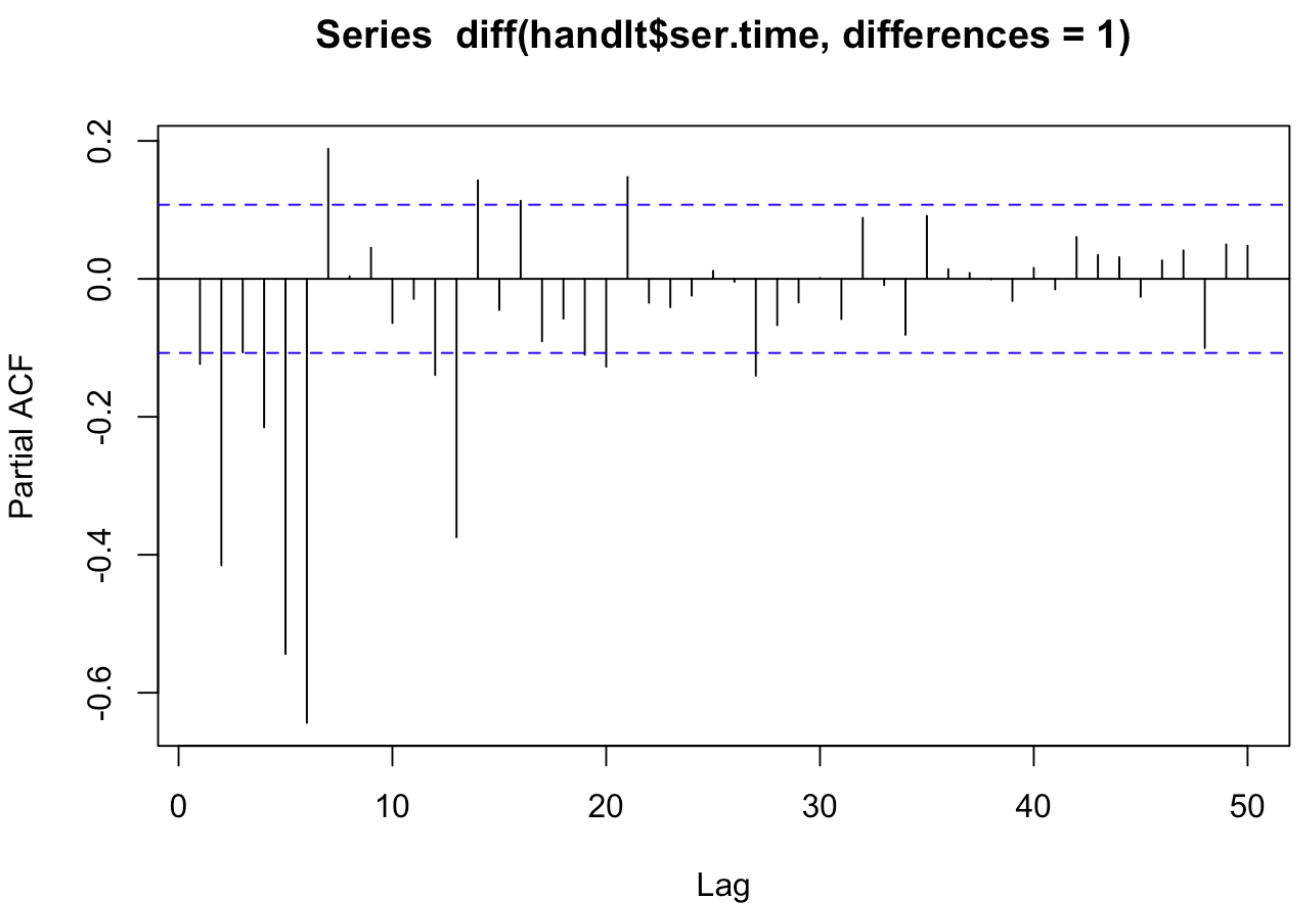
```
##  [1]  2  2  0  2  0  0  1  0  1  0  0  2  3  3 18 17 29 39 47 87 72 79 74
## [24] 68 68 69 60 65 61 71 73 73 47 54 49 36 41 39 29 36 16 28 38 33 30 33
## [47] 30 20
```

```
#call volumes from 00:00:00 to 23:30:00 on 1999-12-15
```

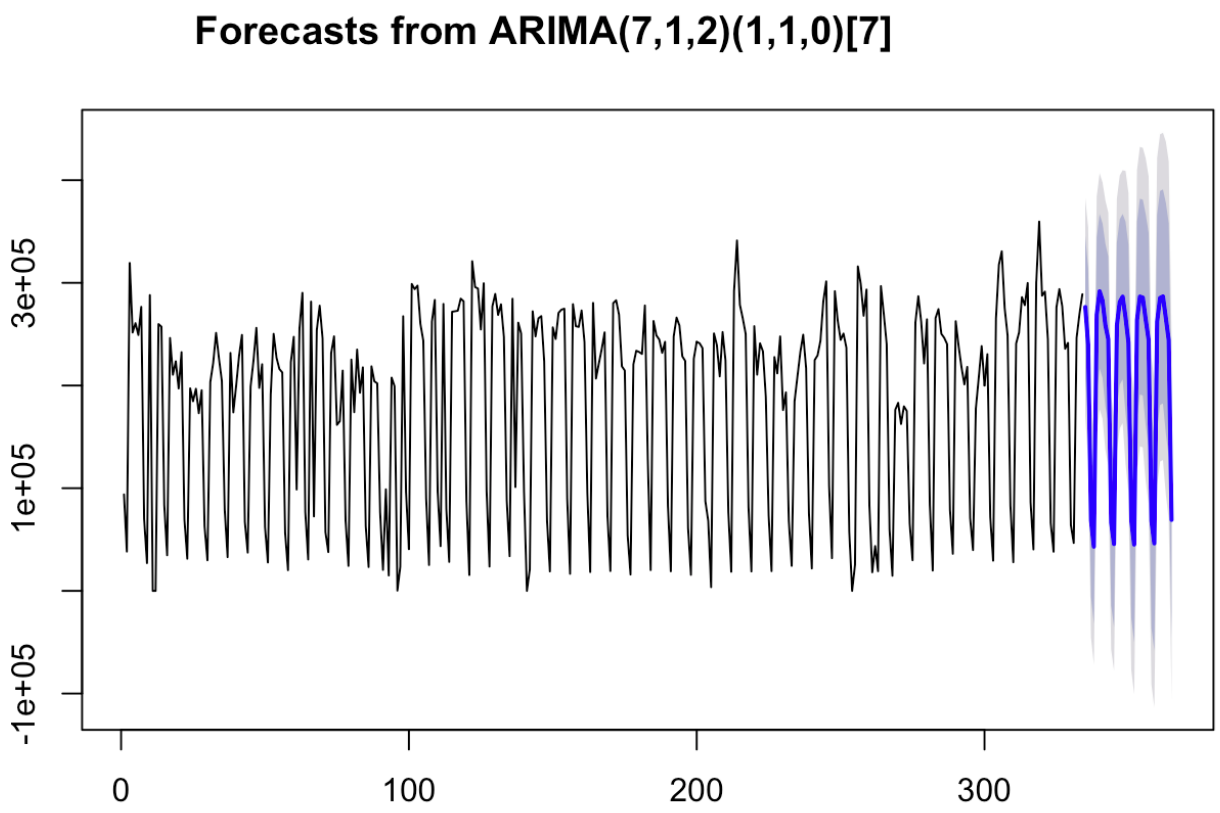
```
#***Question 3***#

data.agent<-subset(data,as.character(data$outcome)=="AGENT  ")
#subtract the data with server.

ser.time=c()
for (i in 10592:10925){
  ser.time[i-10591]=sum(subset(data.agent,data.agent$Call_date==i)[,16])}

handlt<-cbind(dat,ser.time)

#ARIMA model
acf(diff(handlt$ser.time,differences = 1),lag.max = 50)
```




```
pacf(diff(handlt$ser.time,differences = 1),lag.max = 50)
```



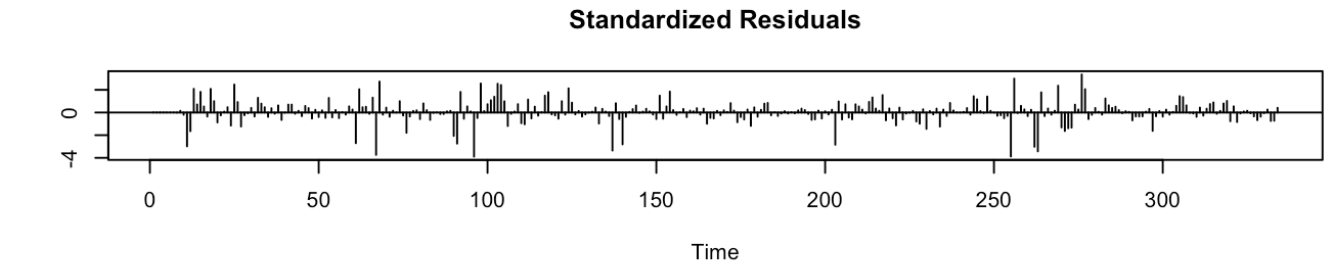

```
ht.fit<-arima(handlt$ser.time,order=c(7,1,2),seasonal=list(order=c(1,1,0), peri
od=7))
pre.ht<-predict(ht.fit,n.ahead=31)
forecast.ht.fit<-(forecast.Arima(ht.fit,h=31))
plot.forecast(forecast.ht.fit)
```

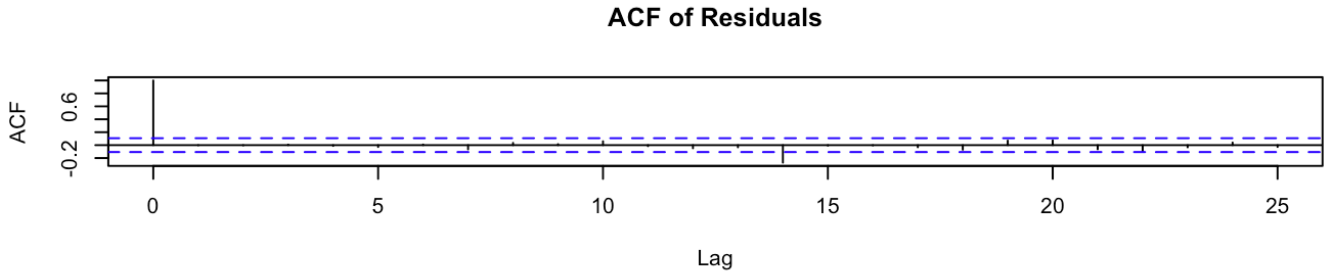


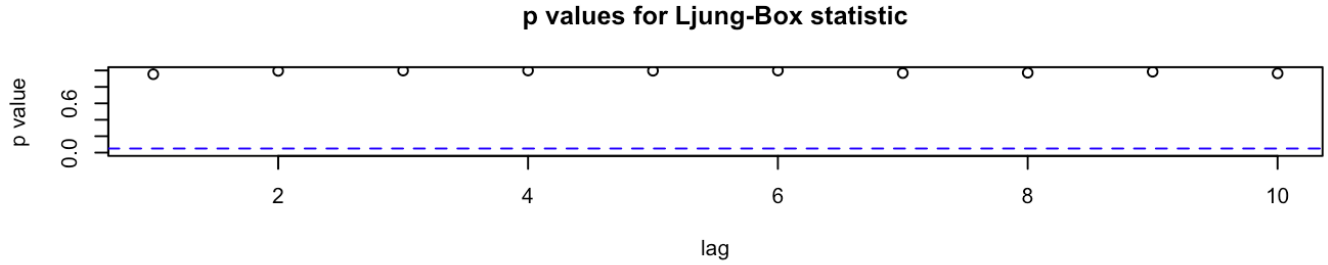

```
AIC(ht.fit)
```

```
## [1] 8059.363
```

```
tsdiag(ht.fit)
```


Standardized Residuals
-4
0
0
50
100
150
200
250
300
Time


ACF of Residuals
ACF
-0.2
0.6
0
5
10
15
20
25
Lag


p values for Ljung-Box statistic
p value
0.0
0.6
2
4
6
8
10
lag