Title: GYM CROWDEDNESS

Group Number: 66

First Name	Last Name	Online Students? (Y or N)	Monday or Tuesday	Shared with ITMD 525? (Y or N)
Sindhu	Reddy	N	Tuesday	N
Prarthana	Shah	N	Tuesday	N

Table of Contents

1. Introduction and Motivations

In today's sedentary lifestyle, it is very important for people to work-out for overall fitness and health. Nowadays people are very busy and they don't have enough time to spare for the gym. In this scenario, it would be very helpful if one can know the best timing to visit the gym.
Hence, we will build the best model to predict how crowded the gym is, and important factors which influence the crowdedness.

2. Data Description

The dataset consists of 26,000 people counts taken at about every 10 minutes over the last 1.5 year (from August 2015 to March 2017). It includes below mentioned attributes:

- Number of people
- date (string; datetime of data)
- timestamp (int; number of seconds since beginning of day)
- day_of_week (int; 0 [monday] - 6 [sunday])
- is_weekend (int; 0 or 1) [boolean, if 1, it's either saturday or sunday, otherwise 0]
- is_holiday (int; 0 or 1) [boolean, if 1 it's a federal holiday, 0 otherwise]
- temperature (float; degrees fahrenheit)
- is_start_of_semester (int; 0 or 1) [boolean, if 1 it's the beginning of a school semester, 0 otherwise]
- month (int; 1 [jan] - 12 [dec])
- hour (int; 0 - 23)

Source : https://www.kaggle.com/nsrose7224/crowdedness-at-the-campus-gym
Domain : Fitness
Dataset size : 62185 rows and 10 columns(variables)

3. Research Problems and Solutions

Problems: What is the best time to go to the gym? Which period of the year attracts more students to the gym? Is the gym crowded on weekend or weekday?

Solutions: We have achieved the solutions of above problems by performing Multinomial Ordinal Regression analysis, time series analysis and plotting box plots.

4. Model Learning

4.1. Data Processing

Original Data:

	A	B	C	D	E	F	G	H	I	J	K
1	date	number_people	timestamp	day_of_week	is_weekend	is_holiday	temperature	is_start_of_semester	is_during_semester	month	hour
2	8/14/2015	37	61211	4	0	0	71.76	0	0	8	17
3	8/14/2015	45	62414	4	0	0	71.76	0	0	8	17
4	8/14/2015	40	63015	4	0	0	71.76	0	0	8	17
5	8/14/2015	44	63616	4	0	0	71.76	0	0	8	17

For Multinominal Ordinal Regression Analysis we converted the below columns:

1) Month: We converted ordinal data into 4 categories: Summer , Fall , Winter and Spring . We created three dummy variables for the month data. December to February is considered as Winter, March to May is considered as Spring, April to June is considered as Summer and the rest as Fall. Refer the below screenshot for your reference.

2)Hour: We even converted hour ordinal data into 4 categories: Morning, Afternoon, Evening and Night. We created three dummy variables for the hour data. 4:00 am to 12:00 pm is considered as Morning, 12:00 pm to 4:00 pm is considered as Afternoon, 4:00 pm to 8:00 pm is considered as Evening and the rest as Night. Refer the below screenshot for your reference.

3)Temperature: We converted numerical data of temperature into 3 categories: low_temp, medium_temp, high_temp. We created two dummy variables for the temperature data. Low temperature being 38F to 54F, Medium temperature being 54.1F to 70F and the highest temperature is 70.1F to 88F.

4)Number_people: We converted the numerical data of People into 3 categories : Category 1 being less number of people (0-48),Category 2 being medium number of people(48 – 96) and category 3 being more number of people(97 – 145).

In all the above cases we always choose n-1 variables as our dummy variables to avoid perfect multicollinearity problem.

Dependant variable(Y) = number_people and all other variables are considered as independent variables

number_people	is_weekend	morning	afternoon	evening	low_temp	medium_temp	summer	fall	spring
1	0	0	0	1	0	0	1	0	0
1	0	0	0	1	0	0	1	0	0
1	0	0	0	1	0	0	1	0	0
1	0	0	0	1	0	0	1	0	0
1	0	0	0	1	0	0	1	0	0
1	0	0	0	1	0	0	1	0	0

For time series analysis, we modified our data into 2 columns: date and average number of people. For this we calculate the average number of people for each date as we have multiple observation for each day (observations were taken every 10 minutes for a single day). Also, we modified our date column to make it unique.

A	B
date1	Average_people
8/14/2015	36.82857143
8/15/2015	21.73
8/16/2015	21.06
8/17/2015	35.51
8/18/2015	34.37623762
8/19/2015	35.57
8/20/2015	32.62
8/21/2015	34.07142857
8/22/2015	30.67

4.2. Data Analytics Tasks and Processes

As mentioned earlier we will be using Multinomial Ordinal Regression Analysis. Multinomial Ordinal Regression Analysis is used when we have multiple response categories. It can be thought of as an extension of the logistic regression model which can be applied to when there are more than two response variables.

#Install the library MASS

```
library(MASS)
```

#Read and Load the data into R

```
mydata=read.csv("ordinal.csv",header=TRUE, sep=',')
```

#Split the data into train and test data

```
mydata=mydata[sample(nrow(mydata)),]
select.data= sample (1:nrow(mydata), 0.8*nrow(mydata))
train.data= mydata[select.data,]
test.data= mydata[-select.data,]
```

```
~/Documents/DataAnalytics/Project
> library(MASS)
> mydata=read.csv("ordinal.csv",header=TRUE, sep=',')
> mydata=mydata[sample(nrow(mydata)),]
> select.data= sample (1:nrow(mydata), 0.8*nrow(mydata))
> train.data= mydata[select.data,]
> test.data= mydata[-select.data,]
```

#Correlation value

```
> cor(cbind(number_people,timestamp,day_of_week,is_weekend,is_holiday,temperature,is_start_of_semester,is_during_semester,month,hour))
                  number_people     timestamp  day_of_week    is_weekend   is_holiday temperature is_start_of_semester is_during_semester        month
number_people        1.00000000  0.5502183683 -0.162061859 -0.1739578118 -0.048249348  0.37332730          0.182682899        0.335350362 -0.097853547
timestamp            0.55021837  1.0000000000 -0.001793191 -0.0005088075  0.002850738  0.18484949          0.009550905        0.044675899 -0.023221076
day_of_week         -0.16206186 -0.0017931908  1.000000000  0.7913381969 -0.075862038  0.01116873         -0.011782025       -0.004823629  0.015558686
is_weekend          -0.17395781 -0.0005088075  0.791338197  1.0000000000 -0.031898834  0.02067334         -0.016645775       -0.036127191  0.008462346
is_holiday          -0.04824935  0.0028507378 -0.075862038 -0.0318988342  1.000000000 -0.08852659         -0.014857908       -0.070798436 -0.094942289
temperature          0.37332730  0.1848494903  0.011168731  0.0206733408 -0.088526592  1.00000000          0.093241863        0.152475895  0.063124581
is_start_of_semester 0.18268290  0.0095509053 -0.011782025 -0.0166457755 -0.014857908  0.09324186          1.000000000        0.209862098 -0.137159611
is_during_semester   0.33535036  0.0446758988 -0.004823629 -0.0361271915 -0.070798436  0.15247590          0.209862098        1.000000000  0.096555677
month               -0.09785355 -0.0232210763  0.015558686  0.0084623464 -0.094942289  0.06312458         -0.137159611        0.096555677  1.000000000
hour                 0.55204944  0.9990774390 -0.001914274 -0.0005172888  0.002843165  0.18512073          0.010090723        0.045580857 -0.023623502
                            hour
number_people         0.5520494424
timestamp             0.9990774390
day_of_week          -0.0019142743
is_weekend           -0.0005172888
is_holiday            0.0028431649
temperature           0.1851207320
is_start_of_semester  0.0100907232
is_during_semester    0.0455808573
month                -0.0236235024
hour                  1.0000000000
```

#Identify Multicollinearity problem

```
> m1=lm(number_people~timestamp+day_of_week+is_weekend+is_holiday+temperature+is_start_of_semester+is_during_semester+month+hour)
> vif(m1)
     timestamp    day_of_week     is_weekend     is_holiday    temperature is_start_of_semester   is_during_semester
    542.541207       2.698149       2.690411       1.026901       1.074686       1.080424       1.088171
         month           hour
     1.049708     542.630330
```

From above vif values we can clearly observe that timestamp and hour has multicollinearity problem. Hence, we have removed timestamp and kept only hour variable.

Furthermore, for performing multinomial ordinal regression analysis, we have processed and converted our data into categorical data and then into numerical format and removed redundant columns like ‘is_holiday’, ‘is_start_of_semester’ and ‘is_during_semester’as explained in our report.

#Build the model – polr function is used for Multinomial Ordinal Regression analysis.

m1=polr(as.factor(number_people)~is_weekend+morning+afternoon+evening+low_temp+medium_temp+summer+fall+spring,data=train.data,Hess=TRUE)

summary(m1)

```
> m1=polr(as.factor(number_people)~is_weekend+morning+afternoon+evening+low_temp+medium_temp+summer+fall+spring,data=train.data,Hess=TRUE)
> summary(m1)
Call:
polr(formula = as.factor(number_people) ~ is_weekend + morning +
    afternoon + evening + low_temp + medium_temp + summer + fall +
    spring, data = train.data, Hess = TRUE)

Coefficients:
             Value Std. Error t value
is_weekend  -0.8008    0.03151 -25.417
morning     -1.0630    0.03778 -28.135
afternoon   -0.2155    0.03670  -5.873
evening      0.9761    0.03225  30.270
low_temp    -2.1904    0.07326 -29.899
medium_temp -0.6012    0.05847 -10.283
summer      -1.9589    0.04713 -41.566
fall        -0.4566    0.03570 -12.789
spring      -1.0402    0.04269 -24.366

Intercepts:
    Value    Std. Error t value
1|2  -0.2707   0.0706    -3.8354
2|3   4.0371   0.0996    40.5202

Residual Deviance: 41469.63
AIC: 41491.63
```

#Build model m2 – Backward Elimination with step

step(m1,direction="backward",trace=F)

```
> step(m1,direction="backward",trace=F)
Call:
polr(formula = as.factor(number_people) ~ is_weekend + morning +
    afternoon + evening + low_temp + medium_temp + summer + fall +
    spring, data = train.data, Hess = TRUE)

Coefficients:
 is_weekend     morning   afternoon     evening    low_temp medium_temp      summer        fall      spring
 -0.7835507  -1.0734644  -0.2629158   0.9679866  -2.2363298  -0.6627262  -1.9435923  -0.4679561  -1.0559645

Intercepts:
       1|2        2|3
-0.3452467  4.0040304

Residual Deviance: 41371.18
AIC: 41393.18
```

#Build model m3 – Step function using Both

step(m1,direction="both",trace=F)

```
> step(m1,direction="both",trace=F)
Call:
polr(formula = as.factor(number_people) ~ is_weekend + morning +
    afternoon + evening + low_temp + medium_temp + summer + fall +
    spring, data = train.data, Hess = TRUE)

Coefficients:
 is_weekend     morning   afternoon     evening    low_temp medium_temp      summer        fall      spring
 -0.7835507  -1.0734644  -0.2629158   0.9679866  -2.2363298  -0.6627262  -1.9435923  -0.4679561  -1.0559645

Intercepts:
       1|2        2|3
-0.3452467  4.0040304

Residual Deviance: 41371.18
AIC: 41393.18
```

#Build model m4 – Best Subset using AdjR2

library(leaps)

leaps(x=train.data[,2:10],y=train.data[,1],names=names(train.data)[2:10],method="adjr2")

```
> leaps(x=train.data[,2:10],y=train.data[,1],names=names(train.data)[2:10],method="adjr2")
$which
  is_weekend morning afternoon evening low_temp medium_temp summer  fall spring
1      FALSE   FALSE     FALSE    TRUE    FALSE       FALSE  FALSE FALSE  FALSE
1      FALSE    TRUE     FALSE   FALSE    FALSE       FALSE  FALSE FALSE  FALSE
1      FALSE   FALSE     FALSE   FALSE    FALSE       FALSE   TRUE FALSE  FALSE
1      FALSE   FALSE     FALSE   FALSE     TRUE       FALSE  FALSE FALSE  FALSE
1      FALSE   FALSE     FALSE   FALSE    FALSE       FALSE  FALSE  TRUE  FALSE
1       TRUE   FALSE     FALSE   FALSE    FALSE       FALSE  FALSE FALSE  FALSE
1      FALSE   FALSE     FALSE   FALSE    FALSE        TRUE  FALSE FALSE  FALSE
1      FALSE   FALSE     FALSE   FALSE    FALSE       FALSE  FALSE FALSE   TRUE
1      FALSE   FALSE      TRUE   FALSE    FALSE       FALSE  FALSE FALSE  FALSE
2      FALSE   FALSE     FALSE    TRUE    FALSE       FALSE   TRUE FALSE  FALSE
2      FALSE    TRUE     FALSE    TRUE    FALSE       FALSE  FALSE FALSE  FALSE
2      FALSE   FALSE     FALSE    TRUE    FALSE       FALSE  FALSE  TRUE  FALSE
2      FALSE   FALSE     FALSE    TRUE     TRUE       FALSE  FALSE FALSE  FALSE
2       TRUE   FALSE     FALSE    TRUE    FALSE       FALSE  FALSE FALSE  FALSE
2      FALSE   FALSE     FALSE    TRUE    FALSE        TRUE  FALSE FALSE  FALSE
2      FALSE   FALSE      TRUE    TRUE    FALSE       FALSE  FALSE FALSE  FALSE
2      FALSE   FALSE     FALSE    TRUE    FALSE       FALSE  FALSE FALSE   TRUE
2      FALSE    TRUE     FALSE   FALSE    FALSE       FALSE   TRUE FALSE  FALSE
2      FALSE    TRUE     FALSE   FALSE    FALSE       FALSE  FALSE  TRUE  FALSE
3      FALSE   FALSE     FALSE    TRUE     TRUE       FALSE   TRUE FALSE  FALSE
3      FALSE    TRUE     FALSE    TRUE    FALSE       FALSE   TRUE FALSE  FALSE
3      FALSE   FALSE     FALSE    TRUE    FALSE        TRUE   TRUE FALSE  FALSE
3       TRUE   FALSE     FALSE    TRUE    FALSE       FALSE   TRUE FALSE  FALSE
3      FALSE    TRUE     FALSE    TRUE    FALSE       FALSE  FALSE  TRUE  FALSE
3       TRUE    TRUE     FALSE    TRUE    FALSE       FALSE  FALSE FALSE  FALSE
3       TRUE   FALSE     FALSE    TRUE    FALSE       FALSE  FALSE  TRUE  FALSE
3      FALSE    TRUE     FALSE    TRUE     TRUE       FALSE  FALSE FALSE  FALSE
3      FALSE   FALSE     FALSE    TRUE    FALSE       FALSE   TRUE FALSE   TRUE
3      FALSE   FALSE     FALSE    TRUE    FALSE       FALSE   TRUE  TRUE  FALSE
4      FALSE    TRUE     FALSE    TRUE     TRUE       FALSE   TRUE FALSE  FALSE
4       TRUE   FALSE     FALSE    TRUE     TRUE       FALSE   TRUE FALSE  FALSE
4      FALSE   FALSE     FALSE    TRUE     TRUE       FALSE   TRUE FALSE   TRUE
4      FALSE    TRUE     FALSE    TRUE    FALSE        TRUE   TRUE FALSE  FALSE
4       TRUE    TRUE     FALSE    TRUE    FALSE       FALSE   TRUE FALSE  FALSE
4      FALSE   FALSE     FALSE    TRUE     TRUE        TRUE   TRUE FALSE  FALSE
4      FALSE   FALSE      TRUE    TRUE     TRUE       FALSE   TRUE FALSE  FALSE
4       TRUE   FALSE     FALSE    TRUE    FALSE        TRUE   TRUE FALSE  FALSE
4      FALSE   FALSE     FALSE    TRUE     TRUE       FALSE   TRUE  TRUE  FALSE
4      FALSE   FALSE     FALSE    TRUE    FALSE        TRUE   TRUE FALSE   TRUE
5       TRUE    TRUE     FALSE    TRUE     TRUE       FALSE   TRUE FALSE  FALSE
5      FALSE    TRUE     FALSE    TRUE     TRUE       FALSE   TRUE FALSE   TRUE
5       TRUE   FALSE     FALSE    TRUE     TRUE       FALSE   TRUE FALSE   TRUE
5      FALSE    TRUE     FALSE    TRUE     TRUE        TRUE   TRUE FALSE  FALSE
5       TRUE    TRUE     FALSE    TRUE    FALSE        TRUE   TRUE FALSE  FALSE
```

```
5      FALSE    TRUE     FALSE    TRUE     TRUE       FALSE   TRUE  TRUE  FALSE
5      FALSE    TRUE      TRUE    TRUE     TRUE       FALSE   TRUE FALSE  FALSE
5      FALSE   FALSE     FALSE    TRUE     TRUE        TRUE   TRUE FALSE   TRUE
6       TRUE    TRUE     FALSE    TRUE     TRUE       FALSE   TRUE FALSE   TRUE
6       TRUE    TRUE     FALSE    TRUE     TRUE        TRUE   TRUE FALSE  FALSE
6       TRUE   FALSE     FALSE    TRUE     TRUE       FALSE   TRUE  TRUE   TRUE
6       TRUE   FALSE     FALSE    TRUE     TRUE        TRUE   TRUE FALSE   TRUE
6      FALSE    TRUE     FALSE    TRUE     TRUE       FALSE   TRUE  TRUE   TRUE
6       TRUE    TRUE     FALSE    TRUE     TRUE       FALSE   TRUE  TRUE  FALSE
6       TRUE    TRUE      TRUE    TRUE     TRUE       FALSE   TRUE FALSE  FALSE
6      FALSE    TRUE     FALSE    TRUE     TRUE        TRUE   TRUE FALSE   TRUE
6       TRUE    TRUE     FALSE    TRUE    FALSE        TRUE   TRUE FALSE   TRUE
6       TRUE   FALSE      TRUE    TRUE     TRUE       FALSE   TRUE FALSE   TRUE
7       TRUE    TRUE     FALSE    TRUE     TRUE       FALSE   TRUE  TRUE   TRUE
7       TRUE    TRUE     FALSE    TRUE     TRUE        TRUE   TRUE FALSE   TRUE
7       TRUE    TRUE      TRUE    TRUE     TRUE       FALSE   TRUE FALSE   TRUE
7       TRUE   FALSE     FALSE    TRUE     TRUE        TRUE   TRUE  TRUE   TRUE
7       TRUE    TRUE      TRUE    TRUE     TRUE        TRUE   TRUE FALSE  FALSE
7      FALSE    TRUE     FALSE    TRUE     TRUE        TRUE   TRUE  TRUE   TRUE
7       TRUE    TRUE     FALSE    TRUE     TRUE        TRUE   TRUE  TRUE  FALSE
7       TRUE   FALSE      TRUE    TRUE     TRUE       FALSE   TRUE  TRUE   TRUE
7       TRUE   FALSE      TRUE    TRUE     TRUE        TRUE   TRUE FALSE   TRUE
7      FALSE    TRUE      TRUE    TRUE     TRUE       FALSE   TRUE  TRUE   TRUE
8       TRUE    TRUE     FALSE    TRUE     TRUE        TRUE   TRUE  TRUE   TRUE
8       TRUE    TRUE      TRUE    TRUE     TRUE       FALSE   TRUE  TRUE   TRUE
8       TRUE    TRUE      TRUE    TRUE     TRUE        TRUE   TRUE FALSE   TRUE
8       TRUE   FALSE      TRUE    TRUE     TRUE        TRUE   TRUE  TRUE   TRUE
8      FALSE    TRUE      TRUE    TRUE     TRUE        TRUE   TRUE  TRUE   TRUE
8       TRUE    TRUE      TRUE    TRUE     TRUE        TRUE   TRUE  TRUE  FALSE
8       TRUE    TRUE      TRUE    TRUE    FALSE        TRUE   TRUE  TRUE   TRUE
8       TRUE    TRUE      TRUE   FALSE     TRUE        TRUE   TRUE  TRUE   TRUE
8       TRUE    TRUE      TRUE    TRUE     TRUE        TRUE  FALSE  TRUE   TRUE
9       TRUE    TRUE      TRUE    TRUE     TRUE        TRUE   TRUE  TRUE   TRUE

$label
 [1] "(Intercept)" "is_weekend"  "morning"     "afternoon"   "evening"     "low_temp"    "medium_temp" "summer"      "fall"        "spring"

$size
 [1]  2  2  2  2  2  2  2  2  2  3  3  3  3  3  3  3  3  3  4  4  4  4  4  4  4  4  4  5  5  5  5  5  5  5  5  5  6  6  6  6  6  6  6  6
[48]  6  6  7  7  7  7  7  7  7  7  7  7  8  8  8  8  8  8  8  8  8  8  9  9  9  9  9  9  9  9 10

$adjr2
 [1]  5.966713e-02  3.893835e-02  2.023296e-02  1.484779e-02  1.412457e-02  1.129674e-02  6.079691e-03  8.629600e-04 -1.995425e-05  7.945805e-02
[11]  7.478441e-02  7.391537e-02  7.149019e-02  7.085561e-02  6.485007e-02  6.273952e-02  6.060977e-02  5.956933e-02  5.277003e-02  1.024639e-01
[21]  9.491553e-02  9.215175e-02  9.048134e-02  8.881652e-02  8.615276e-02  8.536542e-02  8.444325e-02  8.436862e-02  8.420456e-02  1.148540e-01
[31]  1.135968e-01  1.124638e-01  1.061206e-01  1.061179e-01  1.056327e-01  1.035186e-01  1.030249e-01  1.027855e-01  1.008597e-01  1.261414e-01
[41]  1.243693e-01  1.240380e-01  1.173660e-01  1.171721e-01  1.170356e-01  1.154088e-01  1.152590e-01  1.151273e-01  1.150513e-01  1.360872e-01
[51]  1.288924e-01  1.269734e-01  1.268624e-01  1.268465e-01  1.266070e-01  1.264318e-01  1.263877e-01  1.258879e-01  1.249583e-01  1.385528e-01
[61]  1.383137e-01  1.364176e-01  1.302435e-01  1.295005e-01  1.292247e-01  1.291731e-01  1.276302e-01  1.273757e-01  1.273152e-01  1.411560e-01
[71]  1.390447e-01  1.389403e-01  1.305265e-01  1.300698e-01  1.297267e-01  1.261529e-01  1.180224e-01  1.090457e-01  1.420522e-01
```

#Build model m5 – Best Subset using Cp

leaps(x=train.data[,2:10],y=train.data[,1],names=names(train.data)[2:10],method="Cp")

```
> leaps(x=train.data[,2:10],y=train.data[,1],names=names(train.data)[2:10],method="Cp")
$which
  is_weekend morning afternoon evening low_temp medium_temp summer  fall spring
1      FALSE   FALSE     FALSE    TRUE    FALSE       FALSE  FALSE FALSE  FALSE
1      FALSE    TRUE     FALSE   FALSE    FALSE       FALSE  FALSE FALSE  FALSE
1      FALSE   FALSE     FALSE   FALSE    FALSE       FALSE   TRUE FALSE  FALSE
1      FALSE   FALSE     FALSE   FALSE     TRUE       FALSE  FALSE FALSE  FALSE
1      FALSE   FALSE     FALSE   FALSE    FALSE       FALSE  FALSE  TRUE  FALSE
1       TRUE   FALSE     FALSE   FALSE    FALSE       FALSE  FALSE FALSE  FALSE
1      FALSE   FALSE     FALSE   FALSE    FALSE        TRUE  FALSE FALSE  FALSE
1      FALSE   FALSE     FALSE   FALSE    FALSE       FALSE  FALSE FALSE   TRUE
1      FALSE   FALSE      TRUE   FALSE    FALSE       FALSE  FALSE FALSE  FALSE
2      FALSE   FALSE     FALSE    TRUE    FALSE       FALSE   TRUE FALSE  FALSE
2      FALSE    TRUE     FALSE    TRUE    FALSE       FALSE  FALSE FALSE  FALSE
2      FALSE   FALSE     FALSE    TRUE    FALSE       FALSE  FALSE  TRUE  FALSE
2      FALSE   FALSE     FALSE    TRUE     TRUE       FALSE  FALSE FALSE  FALSE
2       TRUE   FALSE     FALSE    TRUE    FALSE       FALSE  FALSE FALSE  FALSE
2      FALSE   FALSE     FALSE    TRUE    FALSE        TRUE  FALSE FALSE  FALSE
2      FALSE   FALSE      TRUE    TRUE    FALSE       FALSE  FALSE FALSE  FALSE
2      FALSE   FALSE     FALSE    TRUE    FALSE       FALSE  FALSE FALSE   TRUE
2      FALSE    TRUE     FALSE   FALSE    FALSE       FALSE   TRUE FALSE  FALSE
2      FALSE    TRUE     FALSE   FALSE    FALSE       FALSE  FALSE  TRUE  FALSE
3      FALSE   FALSE     FALSE    TRUE     TRUE       FALSE   TRUE FALSE  FALSE
3      FALSE    TRUE     FALSE    TRUE    FALSE       FALSE   TRUE FALSE  FALSE
3      FALSE   FALSE     FALSE    TRUE    FALSE        TRUE   TRUE FALSE  FALSE
3       TRUE   FALSE     FALSE    TRUE    FALSE       FALSE   TRUE FALSE  FALSE
3      FALSE    TRUE     FALSE    TRUE    FALSE       FALSE  FALSE  TRUE  FALSE
3       TRUE    TRUE     FALSE    TRUE    FALSE       FALSE  FALSE FALSE  FALSE
3       TRUE   FALSE     FALSE    TRUE    FALSE       FALSE  FALSE  TRUE  FALSE
3      FALSE    TRUE     FALSE    TRUE     TRUE       FALSE  FALSE FALSE  FALSE
3      FALSE   FALSE     FALSE    TRUE    FALSE       FALSE   TRUE FALSE   TRUE
3      FALSE   FALSE     FALSE    TRUE    FALSE       FALSE   TRUE  TRUE  FALSE
4      FALSE    TRUE     FALSE    TRUE     TRUE       FALSE   TRUE FALSE  FALSE
4       TRUE   FALSE     FALSE    TRUE     TRUE       FALSE   TRUE FALSE  FALSE
4      FALSE   FALSE     FALSE    TRUE     TRUE       FALSE   TRUE FALSE   TRUE
4      FALSE    TRUE     FALSE    TRUE    FALSE        TRUE   TRUE FALSE  FALSE
4       TRUE    TRUE     FALSE    TRUE    FALSE       FALSE   TRUE FALSE  FALSE
4      FALSE   FALSE     FALSE    TRUE     TRUE        TRUE   TRUE FALSE  FALSE
4      FALSE   FALSE      TRUE    TRUE     TRUE       FALSE   TRUE FALSE  FALSE
4       TRUE   FALSE     FALSE    TRUE    FALSE        TRUE   TRUE FALSE  FALSE
4      FALSE   FALSE     FALSE    TRUE     TRUE       FALSE   TRUE  TRUE  FALSE
4      FALSE   FALSE     FALSE    TRUE    FALSE        TRUE   TRUE FALSE   TRUE
5       TRUE    TRUE     FALSE    TRUE     TRUE       FALSE   TRUE FALSE  FALSE
5      FALSE    TRUE     FALSE    TRUE     TRUE       FALSE   TRUE FALSE   TRUE
5       TRUE   FALSE     FALSE    TRUE     TRUE       FALSE   TRUE FALSE   TRUE
5      FALSE    TRUE     FALSE    TRUE     TRUE        TRUE   TRUE FALSE  FALSE
5       TRUE    TRUE     FALSE    TRUE    FALSE        TRUE   TRUE FALSE  FALSE
5       TRUE   FALSE     FALSE    TRUE     TRUE        TRUE   TRUE FALSE  FALSE
5      FALSE   FALSE     FALSE    TRUE     TRUE       FALSE   TRUE  TRUE   TRUE
5      FALSE    TRUE     FALSE    TRUE     TRUE       FALSE   TRUE  TRUE  FALSE
```

```
5      FALSE    TRUE      TRUE    TRUE     TRUE       FALSE   TRUE FALSE  FALSE
5      FALSE   FALSE     FALSE    TRUE     TRUE        TRUE   TRUE FALSE   TRUE
6       TRUE    TRUE     FALSE    TRUE     TRUE       FALSE   TRUE FALSE   TRUE
6       TRUE    TRUE     FALSE    TRUE     TRUE        TRUE   TRUE FALSE  FALSE
6       TRUE   FALSE     FALSE    TRUE     TRUE       FALSE   TRUE  TRUE   TRUE
6       TRUE   FALSE     FALSE    TRUE     TRUE        TRUE   TRUE FALSE   TRUE
6      FALSE    TRUE     FALSE    TRUE     TRUE       FALSE   TRUE  TRUE   TRUE
6       TRUE    TRUE     FALSE    TRUE     TRUE       FALSE   TRUE  TRUE  FALSE
6       TRUE    TRUE      TRUE    TRUE     TRUE       FALSE   TRUE FALSE  FALSE
6      FALSE    TRUE     FALSE    TRUE     TRUE        TRUE   TRUE FALSE   TRUE
6       TRUE    TRUE     FALSE    TRUE    FALSE        TRUE   TRUE FALSE   TRUE
6       TRUE   FALSE      TRUE    TRUE     TRUE       FALSE   TRUE FALSE   TRUE
7       TRUE    TRUE     FALSE    TRUE     TRUE       FALSE   TRUE  TRUE   TRUE
7       TRUE    TRUE     FALSE    TRUE     TRUE        TRUE   TRUE FALSE   TRUE
7       TRUE    TRUE      TRUE    TRUE     TRUE       FALSE   TRUE FALSE   TRUE
7       TRUE   FALSE     FALSE    TRUE     TRUE        TRUE   TRUE  TRUE   TRUE
7       TRUE    TRUE      TRUE    TRUE     TRUE        TRUE   TRUE FALSE  FALSE
7      FALSE    TRUE     FALSE    TRUE     TRUE        TRUE   TRUE  TRUE   TRUE
7       TRUE    TRUE     FALSE    TRUE     TRUE        TRUE   TRUE  TRUE  FALSE
7       TRUE   FALSE      TRUE    TRUE     TRUE       FALSE   TRUE  TRUE   TRUE
7       TRUE   FALSE      TRUE    TRUE     TRUE        TRUE   TRUE FALSE   TRUE
7      FALSE    TRUE      TRUE    TRUE     TRUE       FALSE   TRUE  TRUE   TRUE
8       TRUE    TRUE     FALSE    TRUE     TRUE        TRUE   TRUE  TRUE   TRUE
8       TRUE    TRUE      TRUE    TRUE     TRUE       FALSE   TRUE  TRUE   TRUE
8       TRUE    TRUE      TRUE    TRUE     TRUE        TRUE   TRUE FALSE   TRUE
8       TRUE   FALSE      TRUE    TRUE     TRUE        TRUE   TRUE  TRUE   TRUE
8      FALSE    TRUE      TRUE    TRUE     TRUE        TRUE   TRUE  TRUE   TRUE
8       TRUE    TRUE      TRUE    TRUE     TRUE        TRUE   TRUE  TRUE  FALSE
8       TRUE    TRUE      TRUE    TRUE    FALSE        TRUE   TRUE  TRUE   TRUE
8       TRUE    TRUE      TRUE   FALSE     TRUE        TRUE   TRUE  TRUE   TRUE
8       TRUE    TRUE      TRUE    TRUE     TRUE        TRUE  FALSE  TRUE   TRUE
9       TRUE    TRUE      TRUE    TRUE     TRUE        TRUE   TRUE  TRUE   TRUE

$label
 [1] "(Intercept)" "is_weekend"  "morning"     "afternoon"   "evening"     "low_temp"    "medium_temp" "summer"      "fall"        "spring"

$size
 [1]  2  2  2  2  2  2  2  2  2  3  3  3  3  3  3  3  3  3  4  4  4  4  4  4  4  4  4  5  5  5  5  5  5  5  5  5  6  6  6  6  6  6  6  6
[48]  6  6  7  7  7  7  7  7  7  7  7  7  8  8  8  8  8  8  8  8  8  8  9  9  9  9  9  9  9  9  9 10

$Cp
 [1] 4778.7998 5980.6833 7065.2476 7377.4868 7419.4201 7583.3819 7885.8735 8188.3468 8239.5394 3632.2221 3903.2004 3953.5880 4094.2001 4130.9933
[15] 4479.1960 4601.5658 4725.0493 4785.3743 5179.5989 2299.2891 2736.9391 2897.1800 2994.0293 3090.5536 3244.9959 3290.6455 3344.1120 3348.4385
[29] 3357.9510 1581.8918 1654.7857 1720.4745 2088.2361 2088.3930 2116.5247 2239.0964 2267.7225 2281.5970 2393.2553  928.4545 1031.1959 1050.4060
[43] 1437.2267 1448.4664 1456.3790 1550.6955 1559.3837 1567.0184 1571.4249  352.8218  769.9458  881.2035  887.6387  888.5583  902.4447  912.6002
[57]  915.1589  944.1356  998.0253  210.8742  224.7354  334.6616  692.5989  735.6769  751.6689  754.6585  844.1065  858.8582  862.3661   60.9572
[71]  183.3517  189.4088  677.1793  703.6552  723.5461  930.7339 1402.0860 1922.4909   10.0000
```

All the methods for model selection yields the same model which has all X-variables and hence AIC value is also same for the models generated using backward selection, stepwise (direction= backward and both) and best subset selection (using adjr2 and Cp method).

So, we selected m1 as our final model and below are the next steps of ordinal logistic regression

#Finding p-values of all the variables

```
~/Documents/DataAnalytics/Project
> (ctable=coef(summary(m1)))
                Value Std. Error    t value
is_weekend  -0.8008268 0.03150801 -25.416611
morning     -1.0630013 0.03778183 -28.135251
afternoon   -0.2155136 0.03669772  -5.872670
evening      0.9761086 0.03224620  30.270498
low_temp    -2.1903916 0.07325892 -29.899317
medium_temp -0.6012326 0.05846986 -10.282779
summer      -1.9588995 0.04712698 -41.566411
fall        -0.4565822 0.03570252 -12.788515
spring      -1.0401966 0.04269096 -24.365735
1|2         -0.2706746 0.07057311  -3.835379
2|3          4.0371309 0.09963251  40.520218
>
> p=pnorm(abs(ctable[,"t value"]),lower.tail=FALSE)*2
> (ctable=cbind(ctable,"p value"=p))
                Value Std. Error    t value       p value
is_weekend  -0.8008268 0.03150801 -25.416611 1.652503e-142
morning     -1.0630013 0.03778183 -28.135251 3.631183e-174
afternoon   -0.2155136 0.03669772  -5.872670  4.288309e-09
evening      0.9761086 0.03224620  30.270498 2.803963e-201
low_temp    -2.1903916 0.07325892 -29.899317 2.008422e-196
medium_temp -0.6012326 0.05846986 -10.282779  8.426251e-25
summer      -1.9588995 0.04712698 -41.566411  0.000000e+00
fall        -0.4565822 0.03570252 -12.788515  1.900704e-37
spring      -1.0401966 0.04269096 -24.365735 3.949038e-131
1|2         -0.2706746 0.07057311  -3.835379  1.253709e-04
2|3          4.0371309 0.09963251  40.520218  0.000000e+00
>
```

Since we are using all the x-variables, we find p-value of each independent variable to check if they are significant or not. From the above graph it is clear that all the variables have p-value less than 0.05 at 95% confidence level.

 # Interpretation using OR

```
> ci=confint(m1)
Waiting for profiling to be done...
> exp(cbind(OR=coef(m1),ci))
                  OR      2.5 %    97.5 %
is_weekend  0.4489576 0.42197044 0.4774324
morning     0.3454175 0.32067862 0.3718730
afternoon   0.8061273 0.75007726 0.8661305
evening     2.6541078 2.49168236 2.8274259
low_temp    0.1118729 0.09690878 0.1291487
medium_temp 0.5481356 0.48896023 0.6149312
summer      0.1410135 0.12852884 0.1546090
fall        0.6334449 0.59064102 0.6793686
spring      0.3533852 0.32489025 0.3841669
>
```

From the above table, we can understand that when the value of is_weekend changes from 0 to 1, the odds ration of the base case(very low) improves from very low to low, high and very high by 0.4489576. And its clear from the table odds ratio improves a lot almost 2.65% when it is evening, and is least during summer since not many people visit gym because of holidays and since its summer people would like to run outside rather than gym.

Model selection and evaluation is done in the next phase.

We would also like to forecast the outcomes, so that in future we know what is the trend and go the gym, when the number of people are very low.For this purpose we are using Time series Analysis. Time series analysis mostly has methods which helps us analyze time series data to find meaningful statistics and characteristics of the data. Using time series, we are forecasting the future values based on our previous observation.

Below are the steps we followed to forecast the range of people.

#Install the libraries

```
library(tseries)
library(fBasics)
library(forecast)
```

#Import Data

```
mydata=read.table('datewise.csv', header=TRUE, sep=',')
```

#Create variable

```
avg_people=mydata$Average_people
```

#Create Time Series Object Using Zoo package

```
avg_people_ts = zoo(avg_people, as.Date(as.character(mydata$date1),format = "%m/%d/%Y"))
```

~/Documents/DataAnalytics/Project

```
> library(tseries)
> library(fBasics)
> mydata=read.table('datewise.csv', header=TRUE, sep=',')
> avg_people=mydata$Average_people
> avg_people_ts = zoo(avg_people, as.Date(as.character(mydata$date1),format = "%m/%d/%Y"))
>
```

 # train and test data

```
dt_train_data=window(avg_people_ts,start=as.Date("2015-8-15"),end=("2016-12-31"))
dt_test_data=window(avg_people_ts,start=as.Date("2017-1-1"),end=("2017-3-18"))
```

~/Documents/DataAnalytics/Project

```
> dt_train_data=window(avg_people_ts,start=as.Date("2015-8-15"),end=("2016-12-31"))
> dt_test_data=window(avg_people_ts,start=as.Date("2017-1-1"),end=("2017-3-18"))
>
```

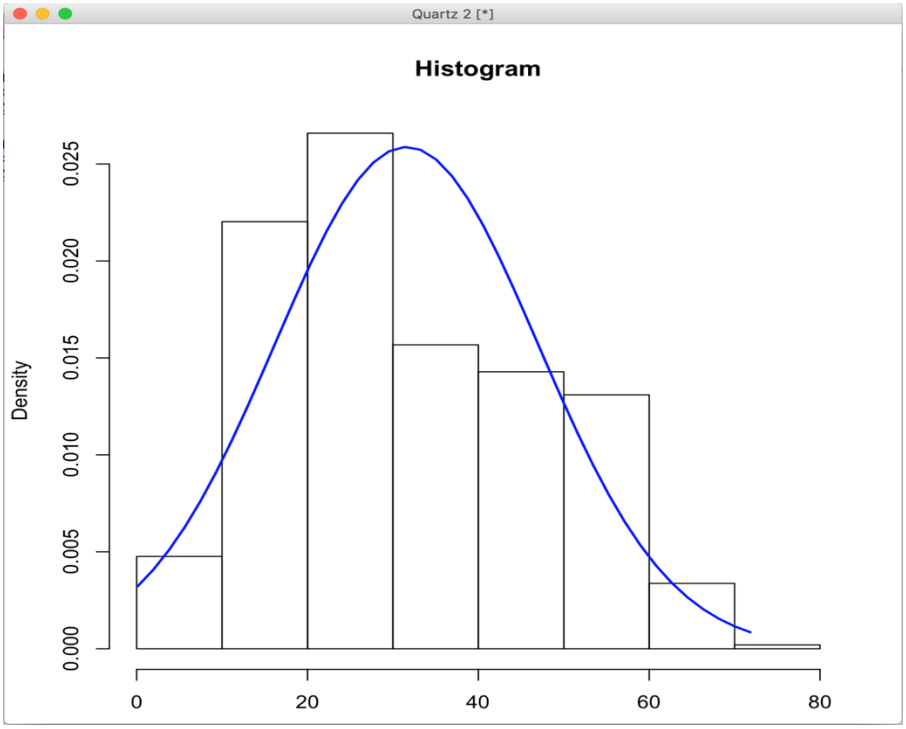
#Compute summary statistics

```
~/Documents/DataAnalytics/Project
> basicStats(dt_train_data)
                        x
nobs          504.000000
NAs             0.000000
Minimum         0.120000
Maximum        71.843750
1. Quartile    19.374692
3. Quartile    44.202767
Mean           31.582239
Median         27.179862
Sum         15917.448301
SE Mean         0.686638
LCL Mean       30.233206
UCL Mean       32.931271
Variance      237.621992
Stdev          15.414992
Skewness        0.372723
Kurtosis       -0.828474
>
```

#Create Histogram

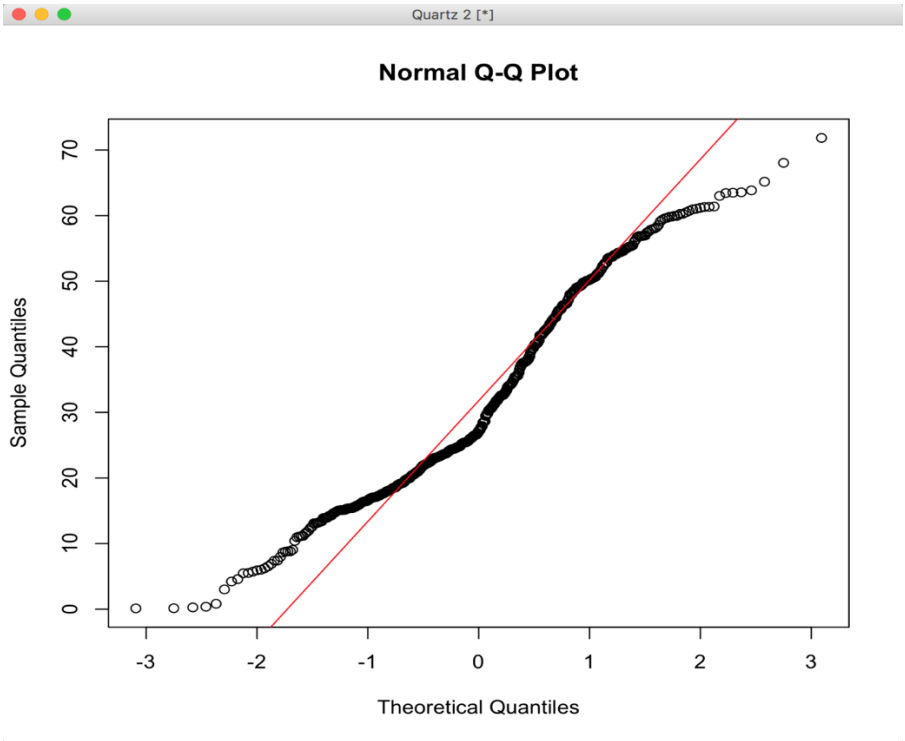
hist(dt_train_data, xlab="Average Number of People", prob=TRUE, main="Histogram")
xfit<-seq(min(dt_train_data),max(dt_train_data),length=40)
yfit<-dnorm(xfit,mean=mean(dt_train_data),sd=sd(dt_train_data))
lines(xfit, yfit, col="blue", lwd=2)

```
~/Documents/DataAnalytics/Project
> hist(dt_train_data, xlab="Average Number of People", prob=TRUE, main="Histogram")
> xfit<-seq(min(dt_train_data),max(dt_train_data),length=40)
> yfit<-dnorm(xfit,mean=mean(dt_train_data),sd=sd(dt_train_data))
> lines(xfit, yfit, col="blue", lwd=2)
```



#Create QQ Plot/ Normal Probablity Plot

```
qqnorm(dt_train_data)
qqline(dt_train_data,col=2)
```

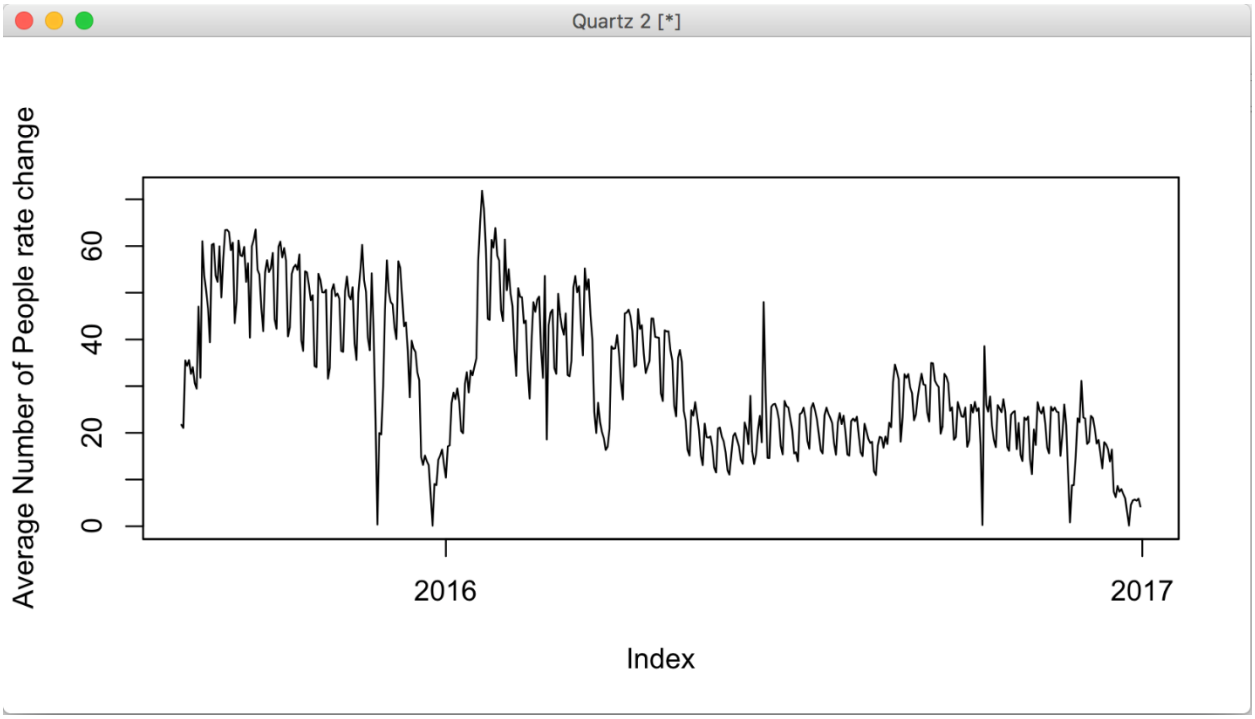


#Normality Test (Jarque-Bera) normality test.

normalTest(dt_train_data,method=c("jb"))

```
~/Documents/DataAnalytics/Project
> normalTest(dt_train_data,method=c("jb"))

Title:
 Jarque - Bera Normalality Test

Test Results:
  STATISTIC:
    X-squared: 25.8538
  P VALUE:
    Asymptotic p Value: 2.432e-06

Description:
 Sat Apr 29 16:53:36 2017 by user:
```

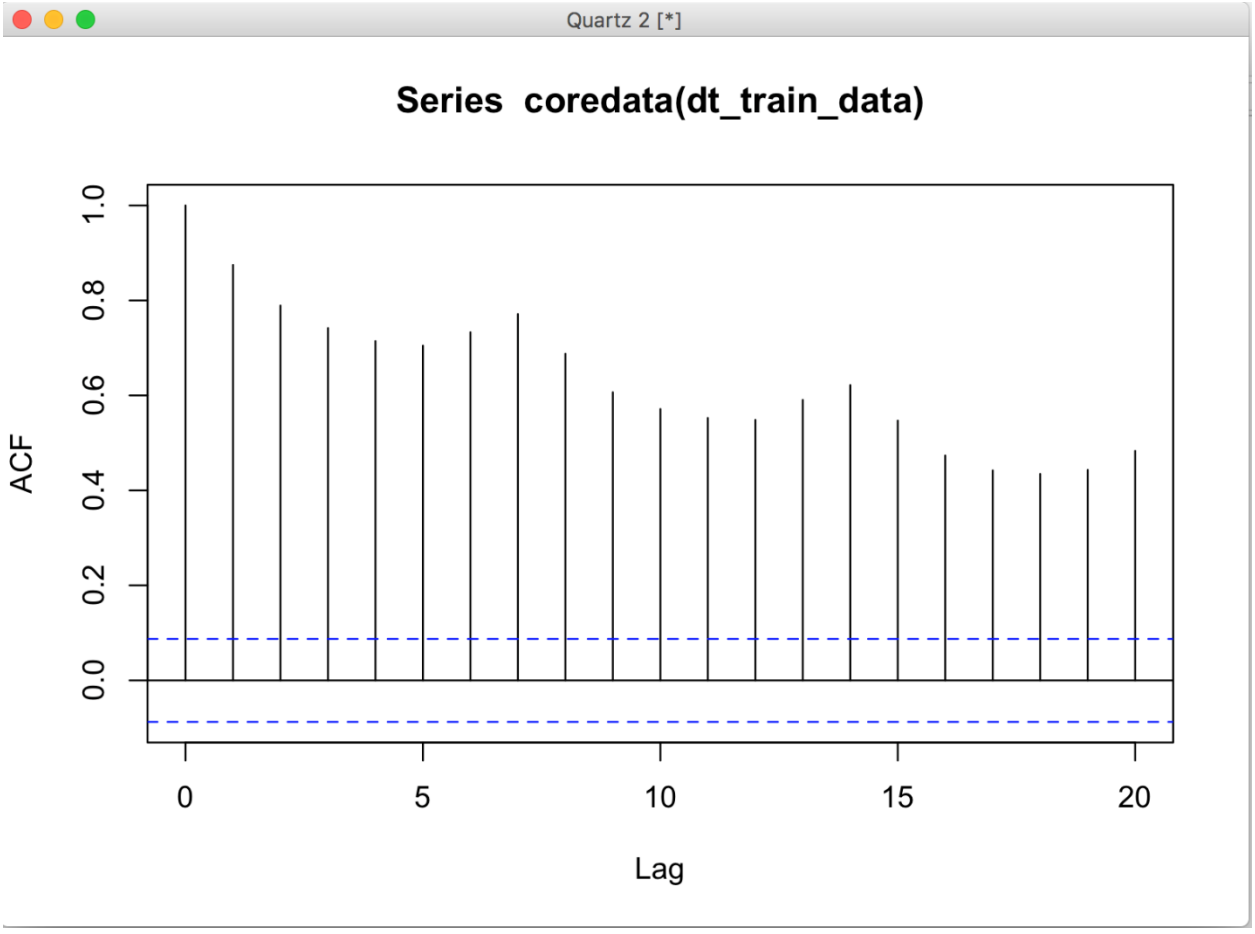
#Create Time Plot
plot(dt_train_data,ylab='Average Number of People rate change')



From the above time plot we can see that the there is seasonal pattern and the data is not stationary.

#Compute ACF,PACF and Plot Correlogram
acf_value=acf(coredata(dt_train_data), plot=FALSE, lag=20)

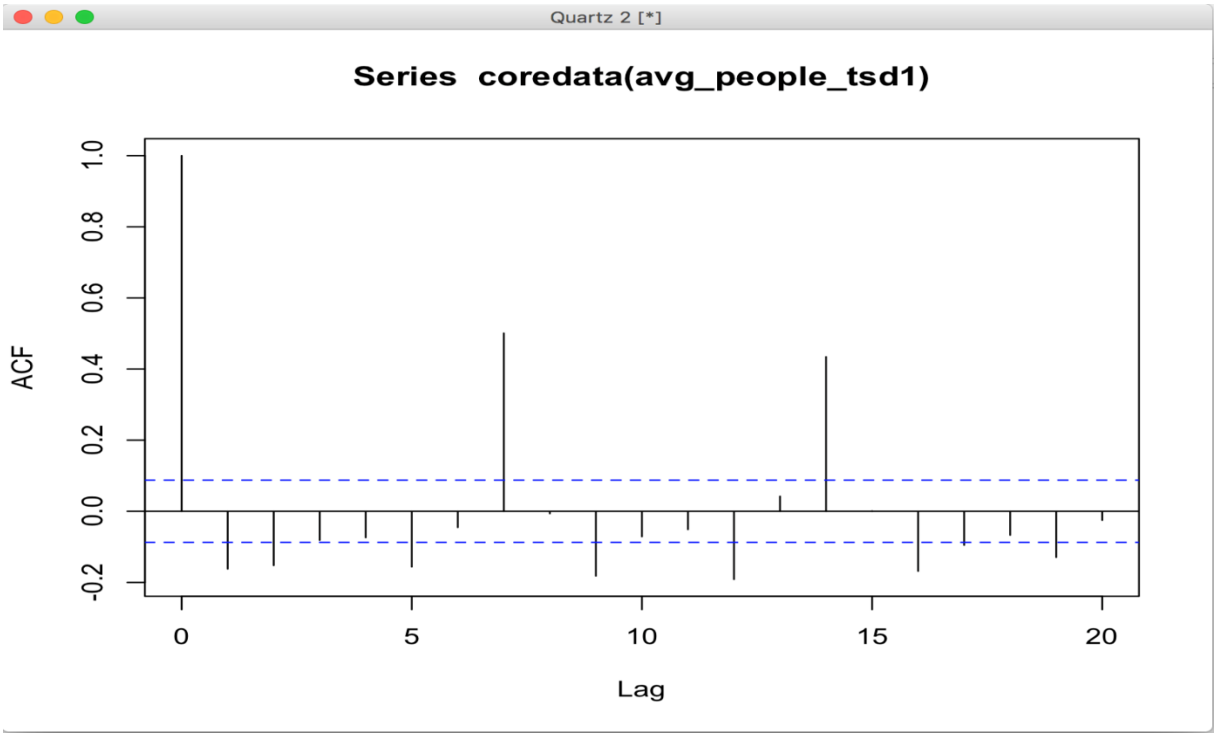
#Plot acf(correlogram)
acf(coredata(dt_train_data), plot=TRUE, lag=20)



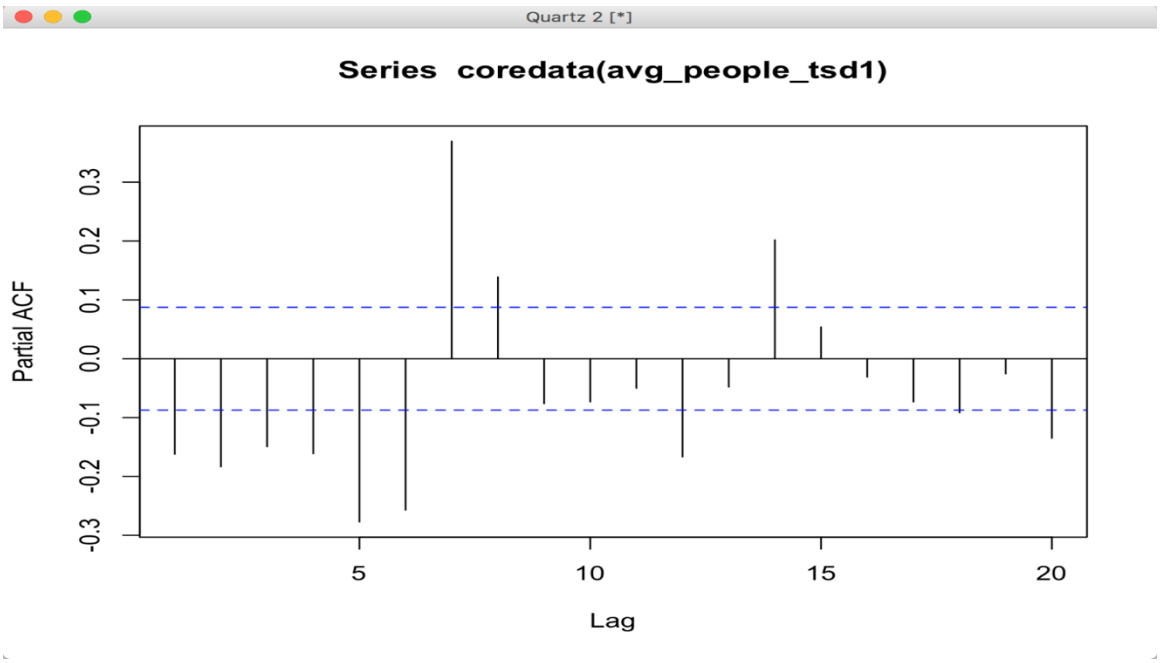
From the above ACF plot, we can observe that the correlation values is decaying slowly hence the data is not stationary, but as there are multiple non-zero co related values, data is serially co related. Now in order to make the data stationary we will apply differencing.

#Apply first differencing

```
avg_people_tsd1=diff(dt_train_data)
acf(coredata(avg_people_tsd1), plot=TRUE, lag=20)
```

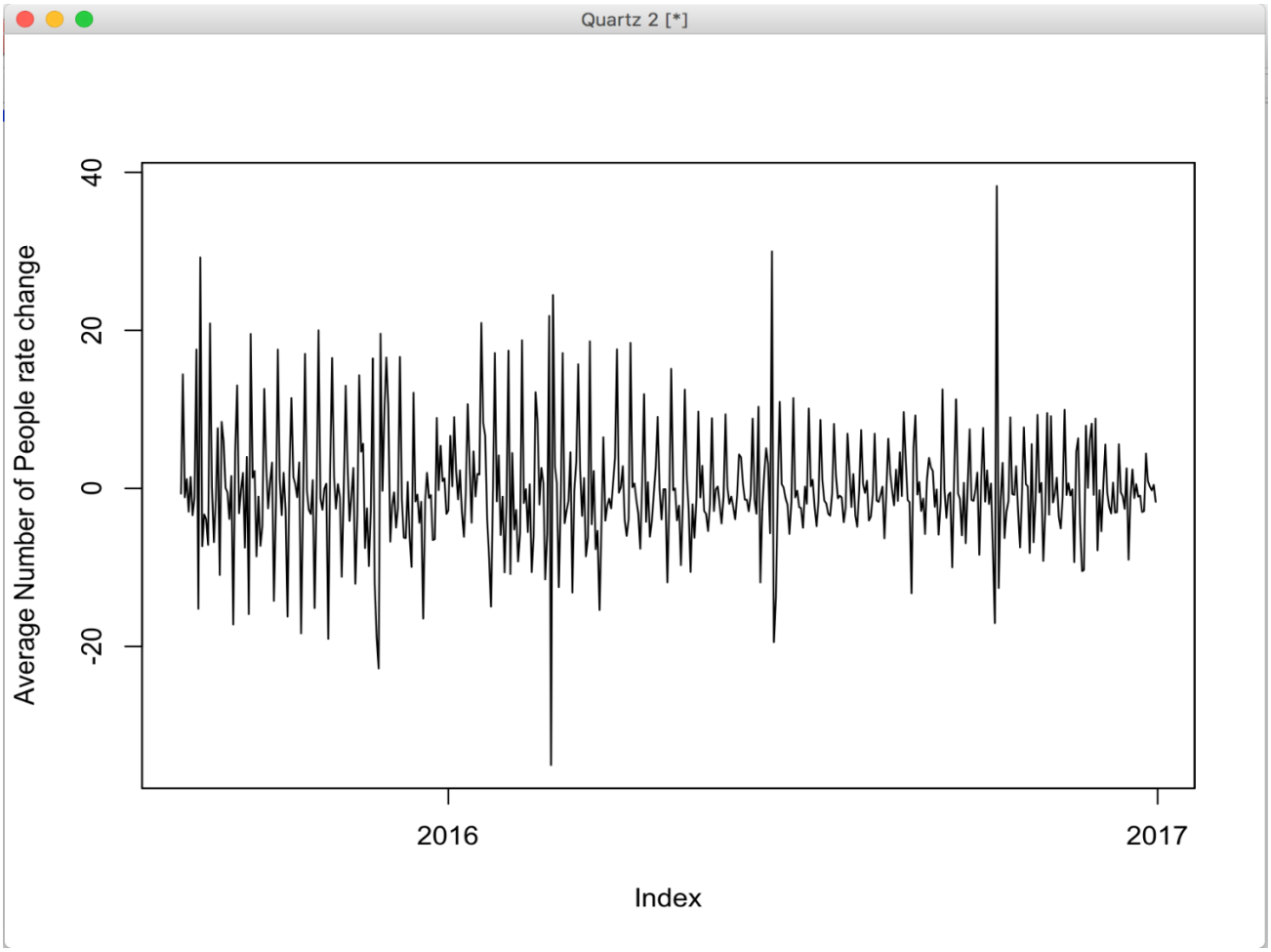



```
pacf(coredata(avg_people_tsd1),lag=20)
```



#Create Time Plot

plot(avg_people_tsd1,ylab='Average Number of People rate change')



So now from the above ACF and time plot we can observe that co relation value is decaying very quickly and hence our data(number of people) is serially correlated.

#Ljung Box Test – To check white noise

Box.test(coredata(avg_people_tsd1),lag=6,type='Ljung')
Box.test(coredata(avg_people_tsd1),lag=12,type='Ljung')

```
~/Documents/DataAnalytics/Project
> Box.test(coredata(avg_people_tsd1),lag=6,type='Ljung')

    Box-Ljung test

data:  coredata(avg_people_tsd1)
X-squared = 44.713, df = 6, p-value = 5.336e-08

> Box.test(coredata(avg_people_tsd1),lag=12,type='Ljung')

    Box-Ljung test

data:  coredata(avg_people_tsd1)
X-squared = 213.24, df = 12, p-value < 2.2e-16
```

From the above graph we can see that p-value is less than 0.05 at 95% confidence level, hence proving there is no white noise present in the data.

#Build model m1 – Identifying p automatically

```
~/Documents/DataAnalytics/Project
> m1=ar(coredata(avg_people_tsd1),method='mle')
> m1

Call:
ar(x = coredata(avg_people_tsd1), method = "mle")

Coefficients:
      1       2       3       4       5       6       7       8       9      10      11      12
-0.3263  -0.1946  -0.1716  -0.1490  -0.1618  -0.1574   0.3207   0.0617  -0.1372  -0.1226  -0.1115  -0.1789

Order selected 12  sigma^2 estimated as  35.97
```

#Build model m2 – ARIMA(5,0,0)

```
> m2=auto.arima(avg_people_tsd1)
> m2
Series: avg_people_tsd1
ARIMA(5,0,0) with zero mean

Coefficients:
          ar1      ar2      ar3      ar4      ar5
      -0.2871  -0.2963  -0.2525  -0.2297  -0.2772
s.e.   0.0428   0.0436   0.0441   0.0435   0.0428

sigma^2 estimated as 48.18:  log likelihood=-1686.77
AIC=3385.54   AICc=3385.71   BIC=3410.88
> 
```

#Build AR model– Manually identifying from PACF plot

The value of p i.e. 7 is identified from above, PACF plot. Since after lag 7 PACF are generally not that significant hence the time series data can be explained by lag-7 data.

m3=arima(avg_people_tsd1,c(7,0,0))

```
> m3=arima(avg_people_tsd1,c(7,0,0))
> m3

Call:
arima(x = avg_people_tsd1, order = c(7, 0, 0))

Coefficients:
          ar1      ar2      ar3      ar4      ar5      ar6     ar7  intercept
      -0.2617  -0.2242  -0.2036  -0.1877  -0.2202  -0.1271  0.3712    -0.0547
s.e.   0.0413   0.0427   0.0427   0.0428   0.0425   0.0425  0.0413     0.1492

sigma^2 estimated as 38.26:  log likelihood = -1631.54,  aic = 3281.08
```

#Build MA model

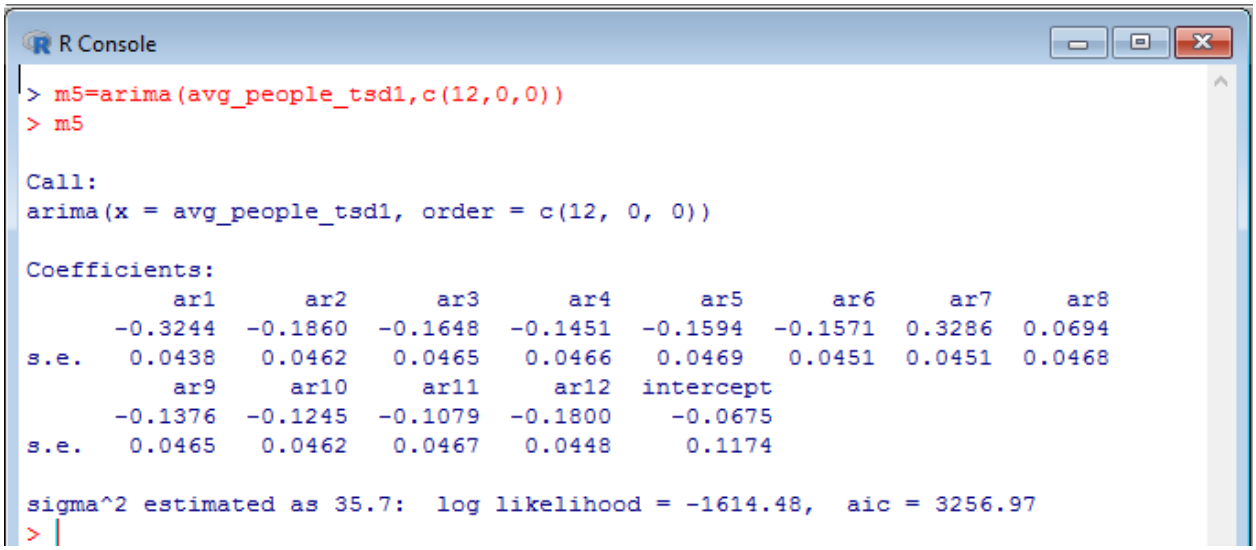
The value of q i.e. 7 is identified from above, ACF plot. Since after lag 7 ACF are generally not that significant hence the time series data can be explained by lag-7 data.

m4=arima(avg_people_tsd1,c(0,0,7))

```
> m4=arima(avg_people_tsd1,c(0,0,7))
> m4

Call:
arima(x = avg_people_tsd1, order = c(0, 0, 7))

Coefficients:
          ma1      ma2      ma3      ma4      ma5      ma6     ma7  intercept
      -0.2315  -0.0943  -0.1016  -0.0838  -0.0957  -0.0982  0.3645    -0.0440
s.e.   0.0408   0.0428   0.0454   0.0461   0.0453   0.0521  0.0407     0.1937

sigma^2 estimated as 43.44:  log likelihood = -1662.94,  aic = 3343.89
>
```

#Build AR(12) model

m5=arima(avg_people_tsd1,c(12,0,0))



Model selection and evaluation is done in the next phase.

Box plot

#Install the library needs

```
library(needs)
```

#Load the library

```
needs(ggplot2,lubridate,rpart.plot,corrplot,dplyr,caret,caretEnsemble,Metrics,doMC)
```

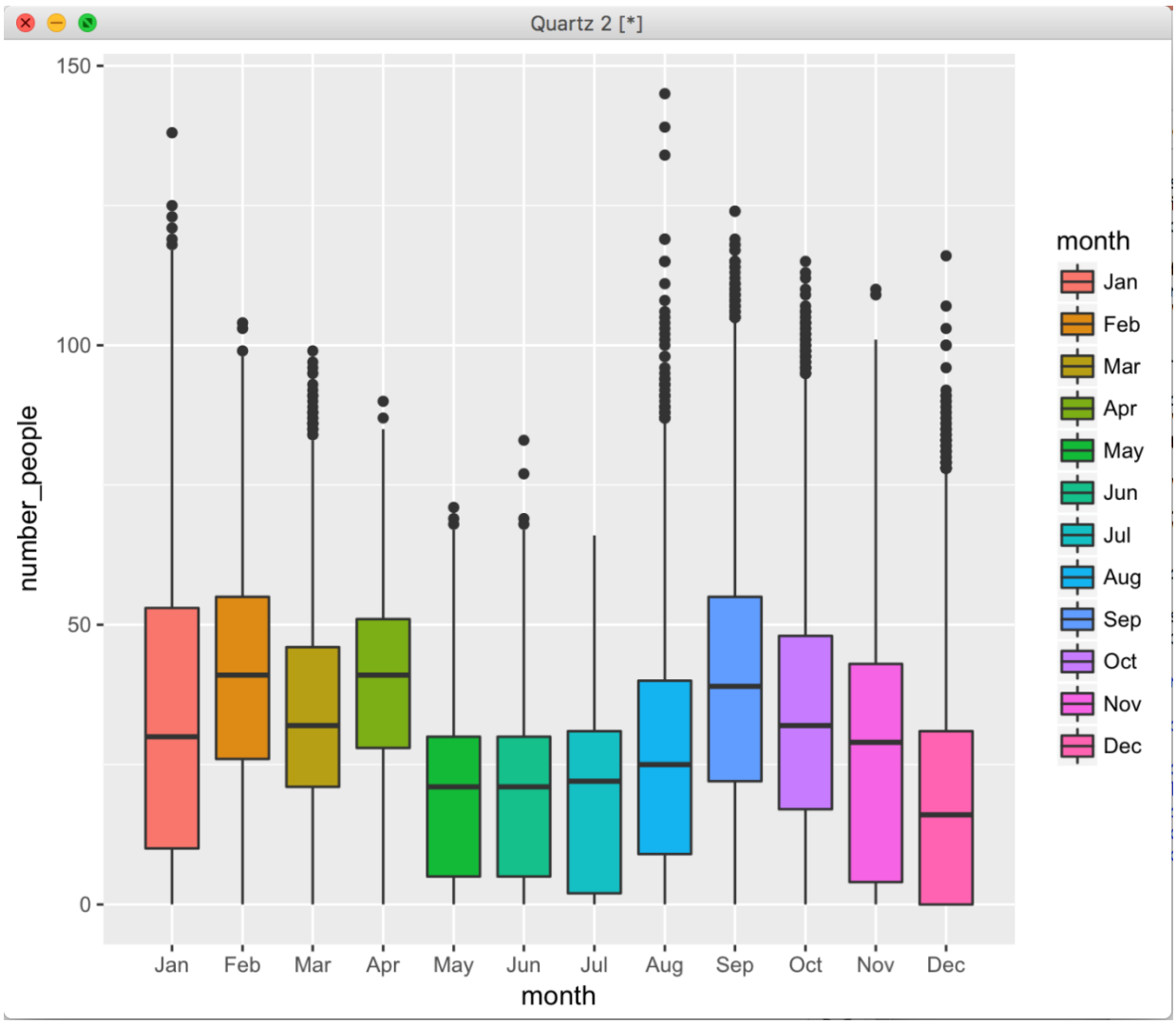
#Read and Load the data into R

```
mydata=read.table('data.csv', header=TRUE, sep=',')
```

#Plot the Boxplot

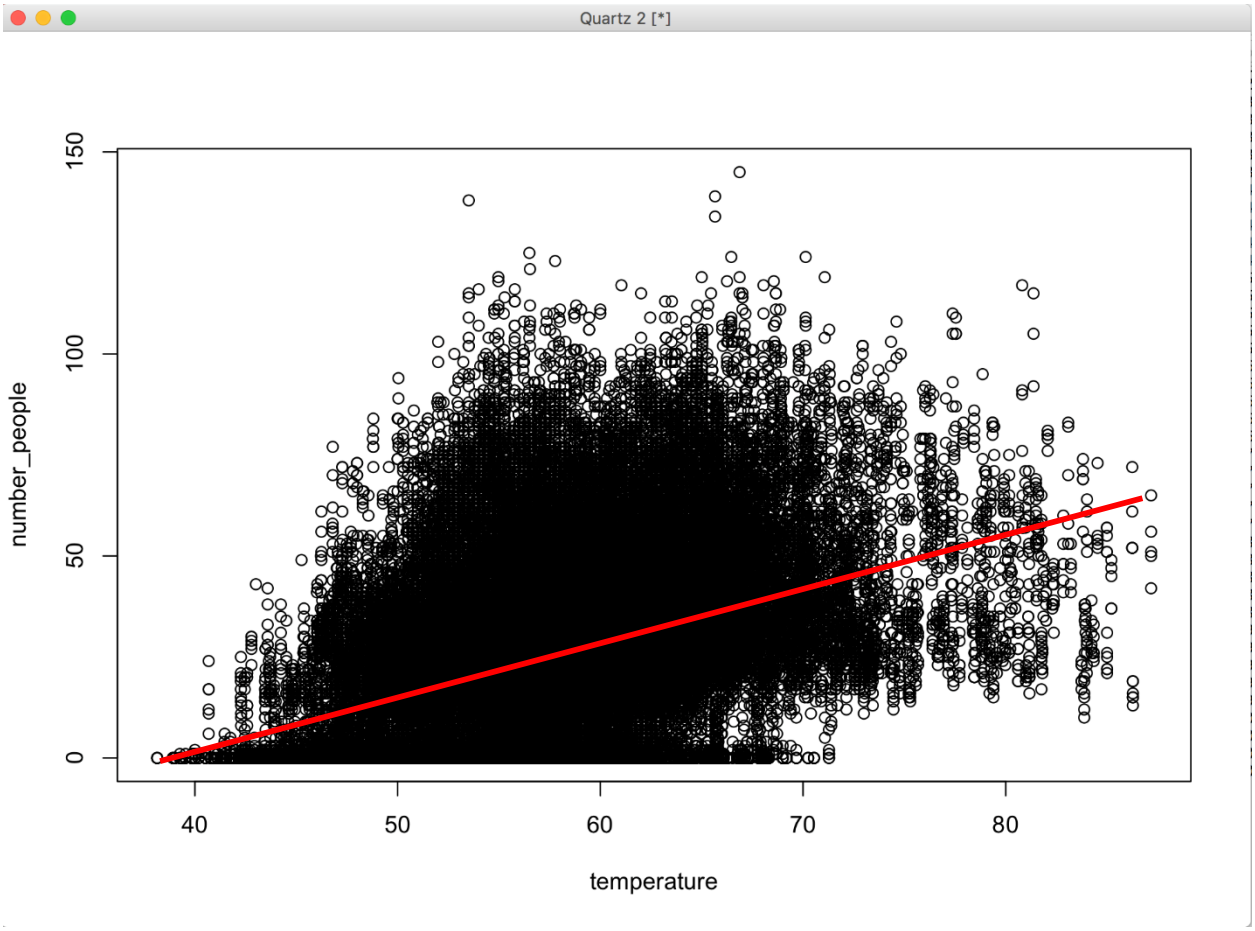
```
ggplot(mydata, aes(x=month, y=number_people,  fill = as.factor(month))) + geom_boxplot() +
scale_fill_discrete(name="month",labels=month.abb[1:12]) +
scale_x_discrete(limits=1:12, labels=month.abb[1:12])
```

From the below graph, we can see that the number of people going to gym in the month of May, June and July are less compared to other months. i.e. Since it's a university gym mostly all the students go back home during summer and people would like to run outside rather than inside the gym.


Quartz 2 [*]
150
100
50
0
number_people
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
month
month
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

#Number_people vs temperature

plot(temperature, number_people)



From the graph it is clear that number of people in the gym increases with the temperature.

Bar Graph

#Install the library needs

```
library(needs)
```

#Load the library

```
needs(ggplot2,lubridate,rpart.plot,corrplot,dplyr,caret,caretEnsemble,Metrics,doMC)
```

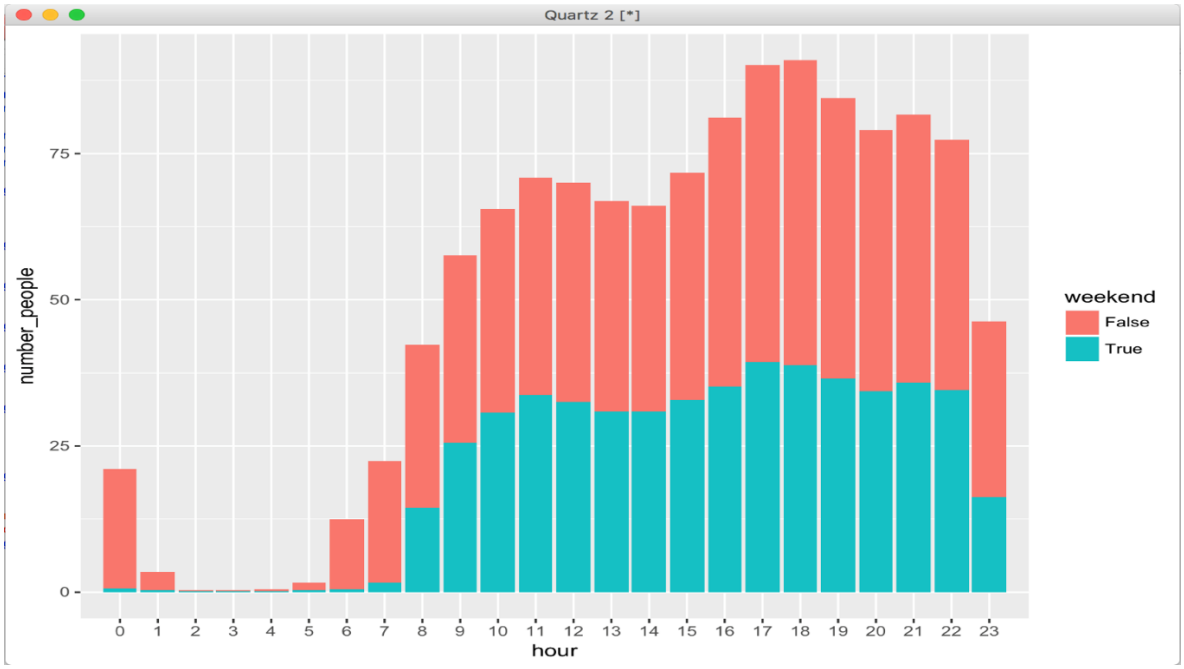
#Read and Load the data into R

```
mydata=read.table('data.csv', header=TRUE, sep=',')
```

#Plot the bar graph

```
ggplot(mydata, aes(x=hour, y=number_people, fill=factor(is_weekend))) + geom_bar(stat = "summary", fun.y = "mean") + scale_fill_discrete(name="weekend", labels=c("False", "True")) + scale_x_discrete(limits=0:23, labels=0:23)
```

From the below graph, we can say that number of people working out in the evening are more compared to that in the night, and number of people working out in weekend are little less compared to that in weekend.



5. Evaluations and Results

5.1. Evaluation Methods

For time series analysis, we spilt the data into two parts, train.data and test.data, as shown below:

```
~/Documents/DataAnalytics/Project
> dt_train_data=window(avg_people_ts,start=as.Date("2015-8-15"),end=("2016-12-31"))
> dt_test_data=window(avg_people_ts,start=as.Date("2017-1-1"),end=("2017-3-18"))
>
```

i.e. we used data from date range 2015 till 2016 as train.data and the remaining data i.e. data from 2017 was used for testing purpose as test.data.

We used AIC for model selection and MAE to evaluate the model as shown below:

Model	**AIC**
ARIMA(5,0,0)	3385.54
AR(7)	3281.08
AR(12)	3256.97
MA(7)	3343.89

After careful observation we select AR(12) model since it has the lowest AIC value.

Model Evaluation:


```
> pr2=predict(m2,n.ahead=33)
> pr3=predict(m3,n.ahead=33)
> pr4=predict(m4,n.ahead=33)
> pr5=predict(m5,n.ahead=33)
>
> diff2=cumsum(c(4.215277778,pr2$pred))
> pred2=diff2[2:34]
> diff3=cumsum(c(4.215277778,pr3$pred))
> pred3=diff3[2:34]
> diff4=cumsum(c(4.215277778,pr4$pred))
> pred4=diff4[2:34]
> diff5=cumsum(c(4.215277778,pr5$pred))
> pred5=diff5[2:34]
> observed=coredata(dt_test_data)
> mae2=mean(abs(observed-pred2))
> mae3=mean(abs(observed-pred3))
> mae4=mean(abs(observed-pred4))
> mae5=mean(abs(observed-pred5))
> mae2
[1] 18.51985
> mae3
[1] 19.44326
> mae4
[1] 19.84554
> mae5
[1] 19.23435
>
```

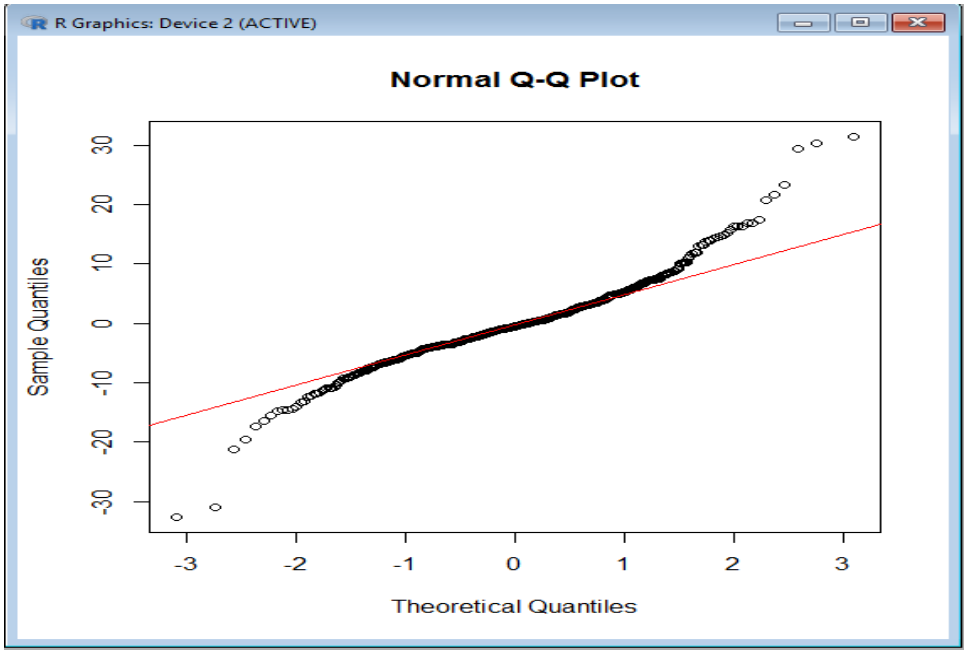
Model	**MAE**
ARIMA(5,0,0)	18.51985
AR(7)	19.44326
AR(12)	19.23435
MA(7)	19.84554

From MAE metric we can observe that almost all the models have same MAE values. Since AIC value for AR(12) is smaller we choose AR(12) as our best model.

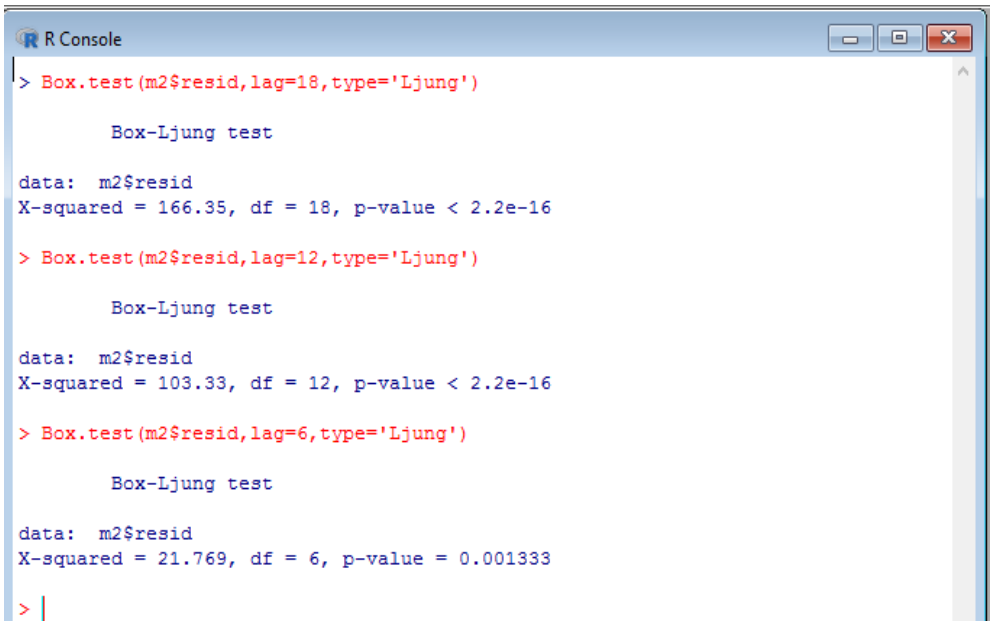
#Residual Analysis for all the models

For ARIMA model (5,0,0) – validate normality

```
qqnorm(m2$residuals)
qqline(m2$residual,col=2)
```

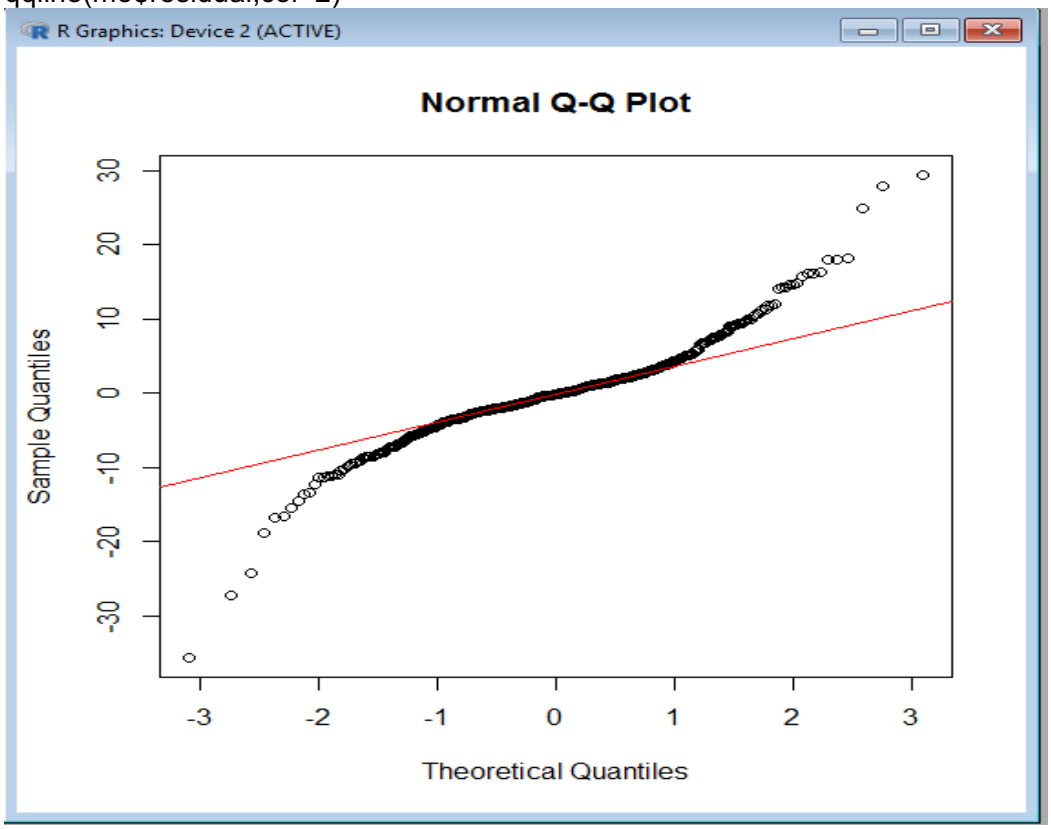


#Ljung Box Test



For AR(7) model– validate normality

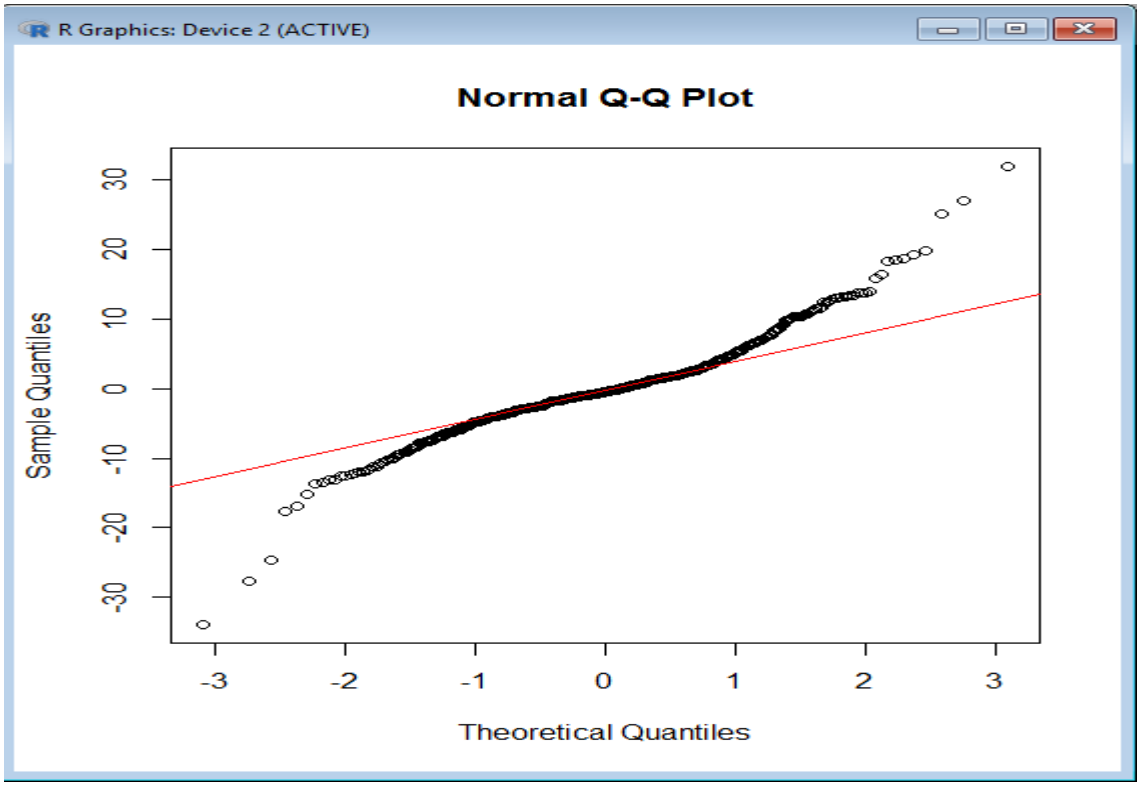
```
qqnorm(m3$residuals)
qqline(m3$residual,col=2)
```



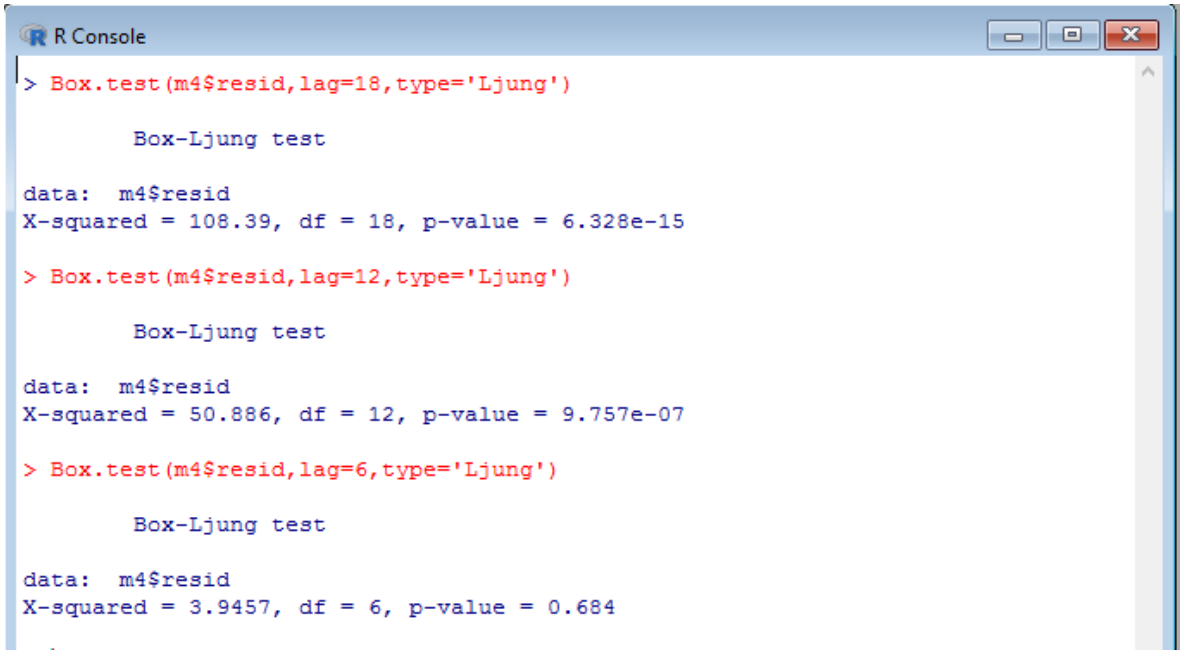

```
R Console
> Box.test(m3$resid,lag=18,type='Ljung')

        Box-Ljung test

data:  m3$resid
X-squared = 36.707, df = 18, p-value = 0.005721

> Box.test(m3$resid,lag=12,type='Ljung')

        Box-Ljung test

data:  m3$resid
X-squared = 27.149, df = 12, p-value = 0.007355

> Box.test(m3$resid,lag=6,type='Ljung')

        Box-Ljung test

data:  m3$resid
X-squared = 5.8671, df = 6, p-value = 0.4382
```

For MA(7) model– validate normality

```
qqnorm(m4$residuals)
qqline(m4$residual,col=2)
```

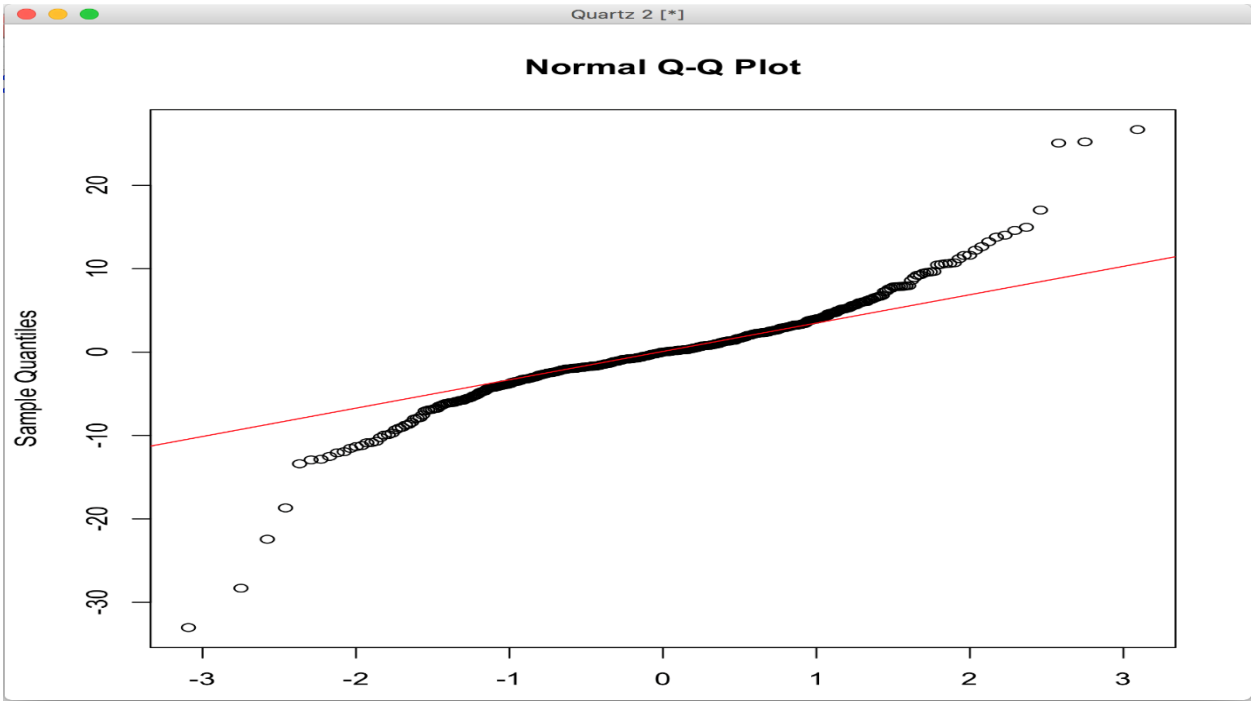


#Ljung Box test

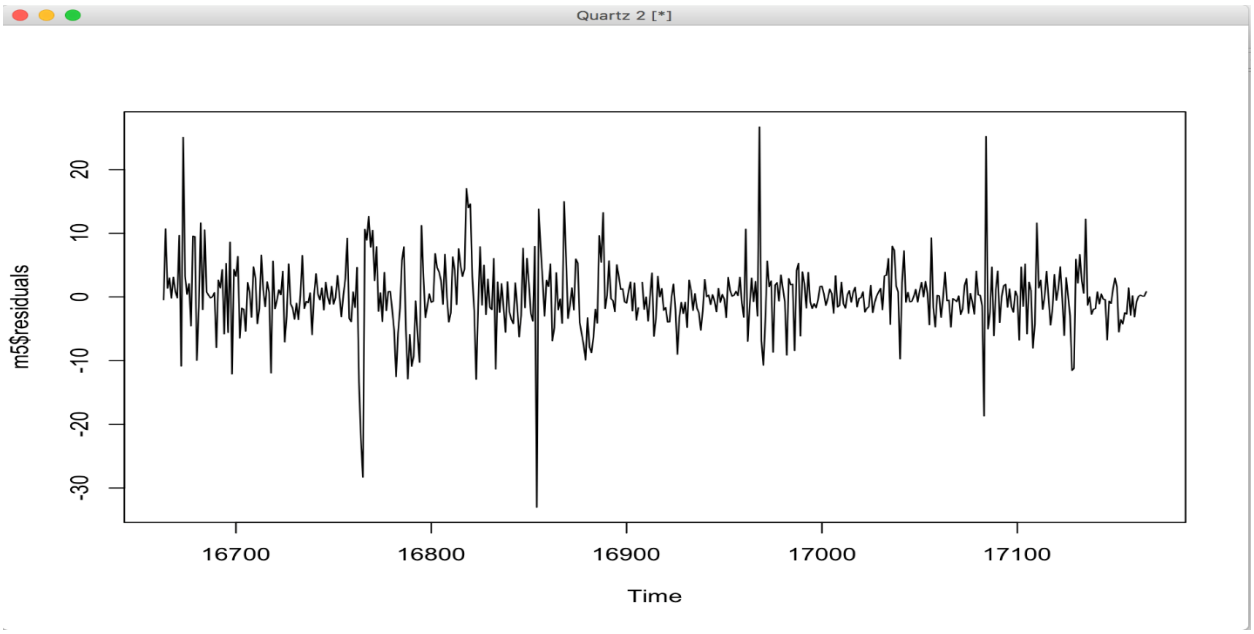


For AR(12) model– Validate Normality

```
qqnorm(m5$residuals)
qqline(m5$residual,col=2)
```




```
plot(m5$residuals,type='l')
```



From the above normality tests, we can say that the residuals are normally distributed.

2) Check white noise

```
> Box.test(m5$resid,lag=18,type='Ljung')

        Box-Ljung test

data:  m5$resid
X-squared = 20.224, df = 18, p-value = 0.3203

> Box.test(m5$resid,lag=12,type='Ljung')

        Box-Ljung test

data:  m5$resid
X-squared = 13.949, df = 12, p-value = 0.304

> Box.test(m5$resid,lag=6,type='Ljung')

        Box-Ljung test

data:  m5$resid
X-squared = 1.2638, df = 6, p-value = 0.9736
```

From the above noise test, we can see that p-value is greater than 0.05 at 95% confidence level hence our assumption of presence of white noise is true.

For Multinomial Ordinal regression analysis, we split the data into 80:20 ratio as shown below:

```
~/Documents/DataAnalytics/Project
> library(MASS)
> mydata=read.csv("ordinal.csv",header=TRUE, sep=',')
> mydata=mydata[sample(nrow(mydata)),]
> select.data= sample (1:nrow(mydata), 0.8*nrow(mydata))
> train.data= mydata[select.data,]
> test.data= mydata[-select.data,]
```

We used AIC for model selection and used accuracy and confusion matrix to evaluate each model.

```
~/Documents/DataAnalytics/Project
> pred=predict(m1,test.data)
> observed=test.data$number_people
> accuracy=mean(pred==observed)
> accuracy
[1] 0.4070113
>
```

```
~/Documents/DataAnalytics/Project
> library(e1071)
> confusionMatrix(data=pred,reference=observed)
Confusion Matrix and Statistics

          Reference
Prediction    1    2    3    4
         1 1711 1131  386  145
         2 1184 1178 1068  432
         3  385 1122 1593  940
         4   12  102  468  580

Overall Statistics

               Accuracy : 0.407
                 95% CI : (0.3984, 0.4157)
    No Information Rate : 0.2841
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.1904
 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity            0.5197  0.33343   0.4532  0.27659
Specificity            0.8183  0.69856   0.7257  0.94371
Pos Pred Value         0.5073  0.30502   0.3943  0.49914
Neg Pred Value         0.8256  0.72536   0.7711  0.86545
Prevalence             0.2647  0.28407   0.2826  0.16861
Detection Rate         0.1376  0.09472   0.1281  0.04664
Detection Prevalence   0.2712  0.31053   0.3248  0.09343
Balanced Accuracy      0.6690  0.51600   0.5895  0.61015
>
```

For easy understanding, we interpret the probability of the first few rows of test data as shown below:

```
> predct = predict(m1,test.data[1:5,],type='prob')
> predct
              1         2          3          4
47460 0.16104619 0.3210078 0.35330194 0.16464404
34630 0.16343760 0.3230107 0.35131332 0.16223839
1969  0.20067120 0.3483075 0.32005388 0.13096743
57984 0.60534048 0.2761287 0.09445796 0.02407283
16650 0.06060034 0.1776510 0.39207017 0.36967853
> test.data[1:5,c(2,6,7,8,9,10,11,12,13)]
      is_weekend morning afternoon evening low_temp medium_temp summer fall spring
47460          0       0         0       0        0           1      0    1      0
34630          1       0         0       1        0           1      1    0      0
1969           0       1         0       0        0           1      0    1      0
57984          1       0         0       0        1           0      0    0      0
16650          0       0         1       0        0           1      0    0      0
>
```

From the table we can see that(check row 57984), if it's a weekend and if the temperature is low(all other variables are set to 0), we can find that the probability of people being very low is more, hence can conclude there will be less number of people in gym during weekend and low temperature.

5.2. Results and Findings

Using OLR we found the probability of people at given time and given season and given temperature.

We can forecast the data using time series analysis using the below equation:

AR(12) model equation:

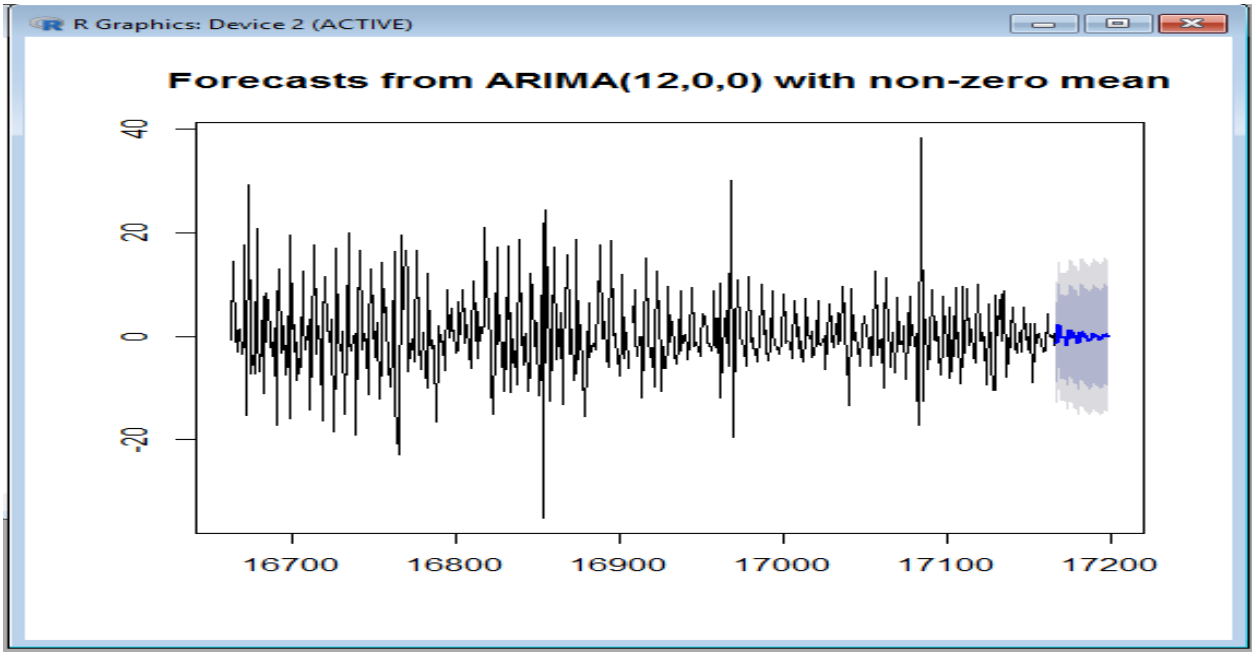
X_t–intercept = $\beta_1(X_{t-1}$-intercept) + $\beta_2(X_{t-2}$-intercept) + $\beta_3(X_{t-3}$-intercept) + $\beta_4(X_{t-4}$-intercept) + $\beta_5(X_{t-5}$-intercept) + $\beta_6(X_{t-6}$-intercept) + $\beta_7(X_{t-7}$-intercept) + $\beta_8(X_{t-8}$-intercept) + $\beta_9(X_{t-9}$-intercept) + $\beta_{10}(X_{t-10}$-intercept) + $\beta_{11}(X_{t-11}$-intercept) + $\beta_{12}(X_{t-12}$-intercept) +a_t

X_t+ 0.0675 = -0.3244(X_{t-1}+0.0675) –0.1860(X_{t-2}+0.0675) - 0.1648(X_{t-3}+0.0675) - 0.1451(X_{t-4}+0.0675) - 0.1594(X_{t-5}+0.0675) - 0.1571(X_{t-6}+0.0675) + 0.3286(X_{t-7}+0.0675) + 0.0694(X_{t-8}+0.0675) - 0.1376(X_{t-9}+0.0675) - 0.1245(X_{t-10}+0.0675) - 0.1079(X_{t-11}+0.0675) - 0.1800(X_{t-12}+0.0675) +a_t

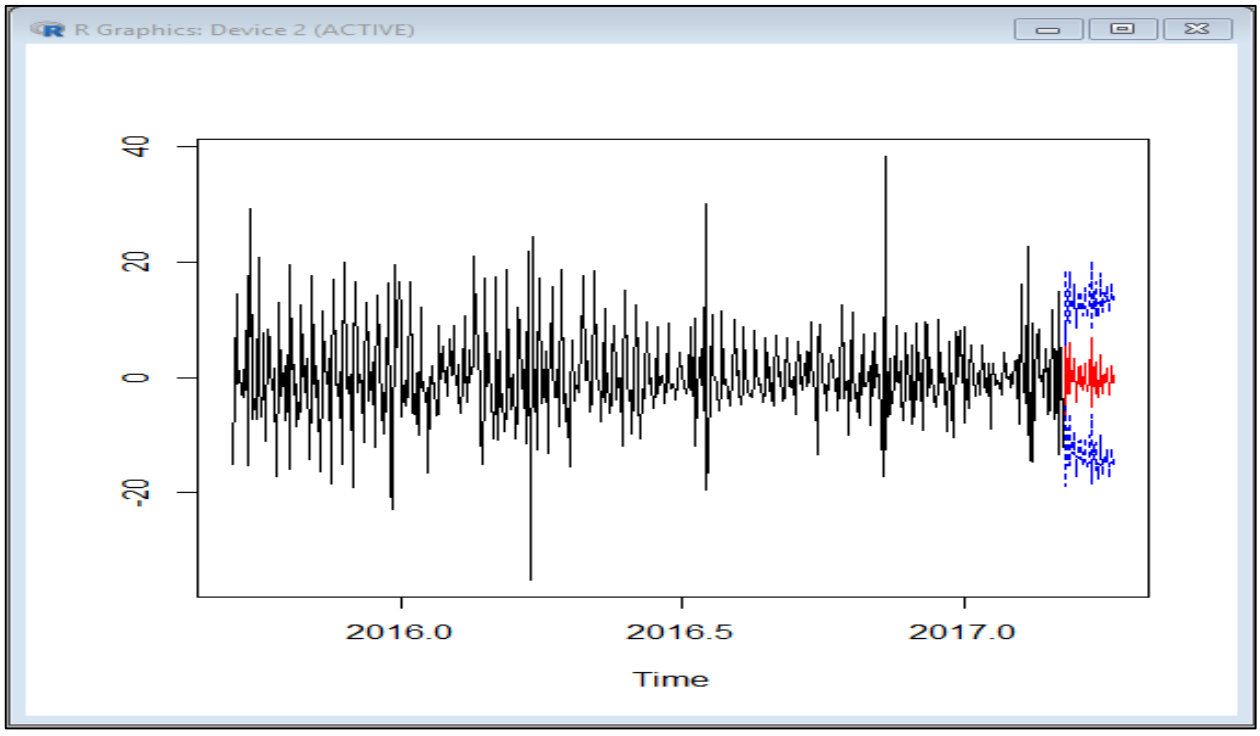
Where a_t is residual error term

#Predictions with best model using time series analysis

plot(forecast(m5,h=33)) – zoo function



plot(forecast(m5,h=33)) – ts function



6. Conclusions and Future Work

6.1. Conclusions

We can conclude that using the above two models, we will be successfully able to predict range of number of people present at a time in the gym.

Few important conclusions we can make after analyzing the data are as shown below :

a) Number of people going to gym during summer are lesser, compared to that of the people going during other months.
b) Number of people going to gym on weekend are lesser compared to that of people going to gym on weekday.
c) Number of people going to gym increases with increase in temperature.
d) Number of people going to gym during start of semester is more compared to end of semester.
e) Number of people going to gym during semester are more compared to semester break or holidays.

6.2. Limitations

Using OLR we are not able to tell exact number of people at a specific time, instead we predict the range of people at any given time and using time series we are not able to tell exact number of people at a specific time, instead we predict the average number of people for a given day.

6.3. Potential Improvements or Future Work

Model can be improved by using algorithms like Xgboost **,** Support Vector Machines and Trees so that the accuracy of OLR model increases.

For time series analysis, we can do predictions on weekly basis, along with daily basis as we have done above.