Final Project

Recruit Restaurant Visitor Forecasting

ALY 6015 Intermediate Analytics

Submitted to:

Joseph Manseau

Date :03/27/2020

Submitted by:

Deepak Natarajan (001088182)

Ashlesha Kshirsagar (001082234)

Abstract

Running a thriving local restaurant is a challenging business. There are different problems faced by owners which are unexpected most of the times. One of the important factors that they want to know is the number customers they are expecting everyday based along with exceptional events such as festivals, holidays, weekends. That makes the process effective right from purchase ingredients and schedule staff members. This forecast is not always easy to since many unpredictable factors affects the restaurants attendance such as location, food price, quality. We have come across a Kaggle competition which is Recruit Restaurant Visitor Forecasting and we used the data to predict the visitors based on methods we learn in this course. Since this data is purely a time series data, we have used other set of data to implement other methods.

1.Introduction

1.1 Motivation

We have learned different methods during course. we wanted to work on big and complex dataset and apply different algorithms on it. We picked the Kaggle competition on Recruit Restaurant Visitor Forecasting. The challenge is to understand the data, try to corelated ,try to visualize the data and interpret the result.

1.2 Goal

The aim of this project is to predict the future numbers of restaurant visitors. This makes it a Time Series Forecasting problem. We have used different forecasting methods and decide which model is the best fit for the data given. Also, we used Decision tree and Random forest algorithm on the dataset.

1.3 Dataset

Recruit Holdings has unique access to key datasets that could make automated future customer prediction possible. Recruit Holdings owns Hot Pepper Gourmet (a restaurant review service), AirREGI (a restaurant point of sales service), and Restaurant Board (reservation log management software).

Hot Pepper Gourmet (hpg): similar to Yelp, here users can search restaurants and also make a reservation online

AirREGI / Restaurant Board (air): similar to Square, a reservation control and cash register system

The individual files:

- air_visit_data.csv: historical visit data for the *air* restaurants. This is essentially the main training data set.
- air_reserve.csv / hpg_reserve.csv: reservations made through the *air* / *hpg* systems.
- air_store_info.csv / hpg_store_info.csv: details about the *air* / *hpg* restaurants including genre and location.
- store_id_relation.csv: connects the *air* and *hpg* ids
- date_info.csv: essentially flags the Japanese holidays.

1.4. Required Packages

We have used below packages in the project

```
library(dplyr)
library(ggplot2)
library(forecast)
library(gridExtra)
library(lubridate)
library(gbm)
library(quantmod)
library(rpart)
library(rpart.plot)
library(randomForest)
```

2.Preliminary Data Analysis

Various aspects of the data were explored with an aim to find relationships between number of visitors at given restaurant and factors like: time, day of the week, holidays, peak time, store distribution as well as statistical summaries of visitors at a given restaurant on given day of the week. The number of daily visitors seems to have a strong weekly cycle in all restaurants. However, individual restaurant's daily visitor cycles differ drastically, which indicates many other factors may impact number of visitors at each place. Further exploratory data analysis showed that factors such type of restaurant, holiday, time to visit the restaurant all have a significant effect on daily visits and weekly visits on a restaurant-specific basis.

2.1. Load Libraries

We loaded require libraries in R

2.2. Load Data

Code:

```
air_visit_data <- read.csv('air_visit_data.csv')
air_reserve <- read.csv('air_reserve.csv')
air_store_info <- read.csv('air_store_info.csv') #done
date_info <- read.csv('date_info.csv') # done
sample_submission <- read.csv('sample_submission.csv')
store_id_relation <- read.csv('store_id_relation.csv')
```

2.3. Overview: File structure and content

```
head(air_visit_data)
head(air_reserve)
head(air_store_info)
head(date_info)
```

1.Air Visit Data
Observations: 252,108
Variables: 3

air_store_id	visit_date	visitors
air_ba937bf13d40fb24	2016-01-13	25
air_ba937bf13d40fb24	2016-01-14	32
air_ba937bf13d40fb24	2016-01-15	29
air_ba937bf13d40fb24	2016-01-16	22
air_ba937bf13d40fb24	2016-01-18	6
air_ba937bf13d40fb24	2016-01-19	9

2. Air Reserve
Observations: 92,378
Variables: 4

air_store_id	visit_datetime	reserve_datetime	reserve_visitors
air_877f79706adbfb06	2016-01-01 19:00:00	2016-01-01 16:00:00	1
air_db4b38ebe7a7ceff	2016-01-01 19:00:00	2016-01-01 19:00:00	3
air_db4b38ebe7a7ceff	2016-01-01 19:00:00	2016-01-01 19:00:00	6
air_877f79706adbfb06	2016-01-01 20:00:00	2016-01-01 16:00:00	2
air_db80363d35f10926	2016-01-01 20:00:00	2016-01-01 01:00:00	5
air_db80363d35f10926	2016-01-02 01:00:00	2016-01-01 16:00:00	2

3. Air Store Info
Observations: 829
Variables: 5

air_store_id	air_genre_name	air_area_name
air_0f0cdeee6c9bf3d7	Italian/French	HyÂ□go-ken KÂ□be-shi KumoidÂ□ri
air_7cc17a324ae5c7dc	Italian/French	HyÂ□go-ken KÂ□be-shi KumoidÂ□ri
air_fee8dcf4d619598e	Italian/French	HyÂ□go-ken KÂ□be-shi KumoidÂ□ri
air_a17f0778617c76e2	Italian/French	HyÂ□go-ken KÂ□be-shi KumoidÂ□ri
air_83db5aff8f50478e	Italian/French	TÂ□kyÂ□-to Minato-ku ShibakÂ□en
air_99c3eae84130c1cb	Italian/French	TÂ□kyÂ□-to Minato-ku ShibakÂ□en

4. date_info

calendar_date	day_of_week	holiday_flg
2016-01-01	Friday	1
2016-01-02	Saturday	1
2016-01-03	Sunday	1
2016-01-04	Monday	0
2016-01-05	Tuesday	0
2016-01-06	Wednesday	0

2.4. Overview of the data and summary

```
summary(air_visit_data)
summary(air_reserve)
summary(air_store_info)
summary(date_info)
```

1. Air Visits

```
air_store_id        visit_date           visitors       
Length:252108      Length:252108      Min.   :  1.00  
Class :character   Class :character   1st Qu.:  9.00  
Mode  :character   Mode  :character   Median : 17.00  
                                      Mean   : 20.97  
                                      3rd Qu.: 29.00  
                                      Max.   :877.00  
```

2.Air Reserve

```
 air_store_id       visit_datetime
 Length:92378       Min.   :2016-01-01 19:00:00
 Class :character   1st Qu.:2016-11-15 19:00:00
 Mode  :character   Median :2017-01-05 18:00:00
                    Mean   :2016-12-05 08:18:58
                    3rd Qu.:2017-03-03 19:00:00
                    Max.   :2017-05-31 21:00:00
 reserve_datetime              reserve_visitors
 Min.   :2016-01-01 01:00:00   Min.   :  1.000
 1st Qu.:2016-11-07 17:00:00   1st Qu.:  2.000
 Median :2016-12-27 22:00:00   Median :  3.000
 Mean   :2016-11-27 01:13:07   Mean   :  4.482
 3rd Qu.:2017-02-26 18:00:00   3rd Qu.:  5.000
 Max.   :2017-04-22 23:00:00   Max.   :100.000
```

3.Air Store data

```
air_store_id       air_genre_name     air_area_name         latitude
Length:829         Length:829         Length:829         Min.   :33.21
Class :character   Class :character   Class :character   1st Qu.:34.70
Mode  :character   Mode  :character   Mode  :character   Median :35.66
                                                         Mean   :35.65
                                                         3rd Qu.:35.69
                                                         Max.   :44.02

  longitude
Min.   :130.2
1st Qu.:135.3
Median :139.7
Mean   :137.4
3rd Qu.:139.8
Max.   :144.3
```

4. Holidays

```
 calendar_date        day_of_week         holiday_flg
 Min.   :2016-01-01   Length:517         Min.   :0.0000
 1st Qu.:2016-05-09   Class :character   1st Qu.:0.0000
 Median :2016-09-15   Mode  :character   Median :0.0000
 Mean   :2016-09-15                      Mean   :0.0677
 3rd Qu.:2017-01-22                      3rd Qu.:0.0000
 Max.   :2017-05-31                      Max.   :1.0000
```

2.5. Missing values

There are no missing values in any of the data

2.6. Reforming features

We change the formatting of the date/time features and reformat a few features to logical and factor variables for exploration purposes. Also, we have removed the latitude and longitude variables

2.7. Combined data

In order to predict the visitors, we need a final dataset which will have all the variables, so we combined the data frame with common columns

Code:

```
# combining the data frame with common columns
air_visit_data1 <- left_join(air_visit_data, air_store_info, by = c('air_store_id' = 'air_store_id'))
air_visit_data2 <- left_join(air_visit_data1, date_info, by = c('visit_date' = 'calendar_date'))
air_visit_data2 <- mutate(air_visit_data2, holiday_flg = ifelse(holiday_flg == 0, 'No_holiday', 'Holiday'))
air_visit_data2$holiday_flg <- as.factor(air_visit_data2$holiday_flg)
air_visit_data2$visit_date <- as.Date(air_visit_data2$visit_date)
head(air_visit_data2)
str(air_visit_data2)
summary(air_visit_data2)
head(air_reserve)
str(air_reserve)
summary(air_reserve)
air_visit_data3 <- left_join(air_visit_data2, air_reserve, by = c('air_store_id' = 'air_store_id', 'visit_date' = 'visit_date'))
head(air_visit_data3)
str(air_visit_data3)
summary(air_visit_data3)
air_visit_data4 <- na.omit(air_visit_data3)
air_visit_data4$unreserved_visitors <- (air_visit_data4$visitors -(air_visit_data4$reserve_visitors_air))
air_visit_data4$reserved_visitors <- air_visit_data4$visitors - air_visit_data4$unreserved_visitors
air_visit_data4$time_gap_in_hours <- as.numeric(difftime(air_visit_data4$visit_datetime_air,air_visit_data4$reserve_datetime_air), units = 'hours')
breaks <- hour(hm("00:00", "8:00", "2:00", "18:00", "23:59")) # create breaks
labels <- c("Morning", "Afternoon", "Evening", "Night") # labels for the breaks
air_visit_data4$Time_of_day <- cut(x=hour(air_visit_data4$visit_datetime_air), breaks = breaks, labels = labels, include.lowest= TRUE)
air_visit_data4$reserve_datetime_air <- as.POSIXct(air_visit_data4$reserve_datetime_air)
air_visit_data4$visit_month <-  months.Date(air_visit_data4$visit_date)
air_visit_data4$r_month_air <-  months.Date(air_visit_data4$reserve_datetime_air)
air_visit_data4$reserve_visitors_air <- NULL
head(air_visit_data4)
str(air_visit_data4)
air_visit_data4$air_store_id <- as.factor(air_visit_data4$air_store_id)
air_visit_data4$visit_month <- as.factor(air_visit_data4$visit_month)
air_visit_data4$r_month_air <- as.factor(air_visit_data4$r_month_air)
summary(air_visit_data4)
```

Output:

1.We combined the data based on air_visit_data, air_store_info - . air_visit_data

2. **air_visit_data2-** Combined the air_visit_data1, date_info

air_store_id	visit_date	visitors	air_genre_name	air_area_name	latitude	longitude	day_of_week	holiday_flg
air_ba937bf13d40fb24	2016-01-13	25	Dining bar	TÂ□kyÂ□-to Minato-ku ShibakÂ□en	35.65807	139.7516	Wednesday	No_holiday
air_ba937bf13d40fb24	2016-01-14	32	Dining bar	TÂ□kyÂ□-to Minato-ku ShibakÂ□en	35.65807	139.7516	Thursday	No_holiday
air_ba937bf13d40fb24	2016-01-15	29	Dining bar	TÂ□kyÂ□-to Minato-ku ShibakÂ□en	35.65807	139.7516	Friday	No_holiday
air_ba937bf13d40fb24	2016-01-16	22	Dining bar	TÂ□kyÂ□-to Minato-ku ShibakÂ□en	35.65807	139.7516	Saturday	No_holiday
air_ba937bf13d40fb24	2016-01-18	6	Dining bar	TÂ□kyÂ□-to Minato-ku ShibakÂ□en	35.65807	139.7516	Monday	No_holiday
air_ba937bf13d40fb24	2016-01-19	9	Dining bar	TÂ□kyÂ□-to Minato-ku ShibakÂ□en	35.65807	139.7516	Tuesday	No_holiday

3. **air_visit_data3-**

air_visit_data2, air_reserve

air_store_id	visit_date	visitors	air_genre_name	air_area_name	latitude	longitude	day_of_week	holiday_flg	reserve_visitors_air	visit_datetime_air	reserve_datetime_air
air_ba937bf13d40fb24	2016-01-13	25	Dining bar	TÂ□kyÂ□-to Minato-ku ShibakÂ□en	35.65807	139.7516	Wednesday	No_holiday	NA	NA	NA
air_ba937bf13d40fb24	2016-01-14	32	Dining bar	TÂ□kyÂ□-to Minato-ku ShibakÂ□en	35.65807	139.7516	Thursday	No_holiday	NA	NA	NA
air_ba937bf13d40fb24	2016-01-15	29	Dining bar	TÂ□kyÂ□-to Minato-ku ShibakÂ□en	35.65807	139.7516	Friday	No_holiday	NA	NA	NA
air_ba937bf13d40fb24	2016-01-16	22	Dining bar	TÂ□kyÂ□-to Minato-ku ShibakÂ□en	35.65807	139.7516	Saturday	No_holiday	NA	NA	NA
air_ba937bf13d40fb24	2016-01-18	6	Dining bar	TÂ□kyÂ□-to Minato-ku ShibakÂ□en	35.65807	139.7516	Monday	No_holiday	NA	NA	NA
air_ba937bf13d40fb24	2016-01-19	9	Dining bar	TÂ□kyÂ□-to Minato-ku ShibakÂ□en	35.65807	139.7516	Tuesday	No_holiday	NA	NA	NA

air_store_id	visit_date	visitors	air_genre_name	air_area_name	latitude	longitude	day_of_week	holiday_flg	reserve_visitors_air	visit_datetime_air	reserve_datetime_air
air_ba937bf13d40fb24	2016-01-13	25	Dining bar	TÀ□kyÀ□-to Minato-ku ShibakÀ□en	35.65807	139.7516	Wednesday	No_holiday	NA	NA	NA
air_ba937bf13d40fb24	2016-01-14	32	Dining bar	TÀ□kyÀ□-to Minato-ku ShibakÀ□en	35.65807	139.7516	Thursday	No_holiday	NA	NA	NA
air_ba937bf13d40fb24	2016-01-15	29	Dining bar	TÀ□kyÀ□-to Minato-ku ShibakÀ□en	35.65807	139.7516	Friday	No_holiday	NA	NA	NA
air_ba937bf13d40fb24	2016-01-16	22	Dining bar	TÀ□kyÀ□-to Minato-ku ShibakÀ□en	35.65807	139.7516	Saturday	No_holiday	NA	NA	NA
air_ba937bf13d40fb24	2016-01-18	6	Dining bar	TÀ□kyÀ□-to Minato-ku ShibakÀ□en	35.65807	139.7516	Monday	No_holiday	NA	NA	NA
air_ba937bf13d40fb24	2016-01-19	9	Dining bar	TÀ□kyÀ□-to Minato-ku ShibakÀ□en	35.65807	139.7516	Tuesday	No_holiday	NA	NA	NA

4. **air_visit_data4**

	air_store_id	visit_date	visitors	air_genre_name	air_area_name	latitude	longitude	day_of_week	holiday_flg	reserve_visitors_air	visit_datetime_air	reserve_datetime_air	unreserved_visitors	reserved_visitors	time_gap_in_hours	Time_of_day
1007	air_35512c42db0868da	2016-02-27	6	Dining bar	TÀ□kyÀ□-to Musashino-shi MidorichÀ□	35.71778	139.5663	Saturday	No_holiday	12	2016-02-27 22:00:00	2016-02-27 21:00:00	-6	12	0	Night
3745	air_ee3a01f0c71a769f	2016-01-04	61	Cafe/Sweets	Shizuoka-ken Hamamatsu-shi MotoshirochÀ□	34.71090	137.7259	Monday	No_holiday	2	2016-01-04 15:00:00	2016-01-04 11:00:00	59	2	0	Evening
3748	air_ee3a01f0c71a769f	2016-01-08	21	Cafe/Sweets	Shizuoka-ken Hamamatsu-shi MotoshirochÀ□	34.71090	137.7259	Friday	No_holiday	3	2016-01-08 19:00:00	2016-01-06 14:00:00	18	3	48	Night
3749	air_ee3a01f0c71a769f	2016-01-09	57	Cafe/Sweets	Shizuoka-ken Hamamatsu-shi MotoshirochÀ□	34.71090	137.7259	Saturday	No_holiday	1	2016-01-09 11:00:00	2016-01-06 13:00:00	56	1	72	Evening
3750	air_ee3a01f0c71a769f	2016-01-09	57	Cafe/Sweets	Shizuoka-ken Hamamatsu-shi MotoshirochÀ□	34.71090	137.7259	Saturday	No_holiday	2	2016-01-09 11:00:00	2016-01-07 19:00:00	55	2	48	Evening
3751	air_ee3a01f0c71a769f	2016-01-09	57	Cafe/Sweets	Shizuoka-ken Hamamatsu-shi MotoshirochÀ□	34.71090	137.7259	Saturday	No_holiday	3	2016-01-09 11:00:00	2016-01-08 12:00:00	54	3	24	Evening

Interpretation:

We have used the combined data air visit data 4 for building our model. The combine data contain the fields such visit date, visitors, time of the day is being used in forecasting.

3.Individual Feature Visualizations

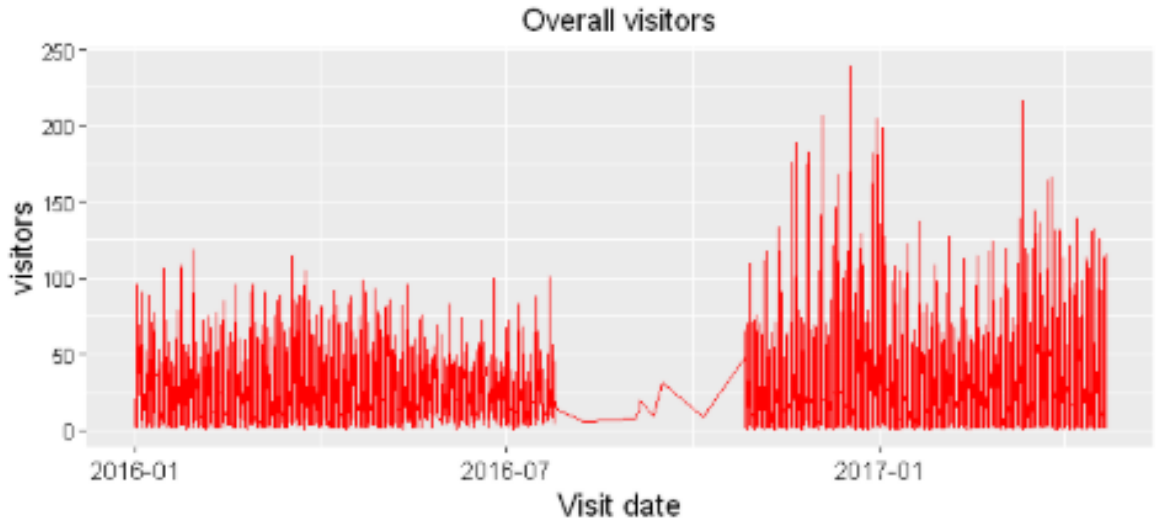
This initial visualization will be foundation on which we will build our analysis.

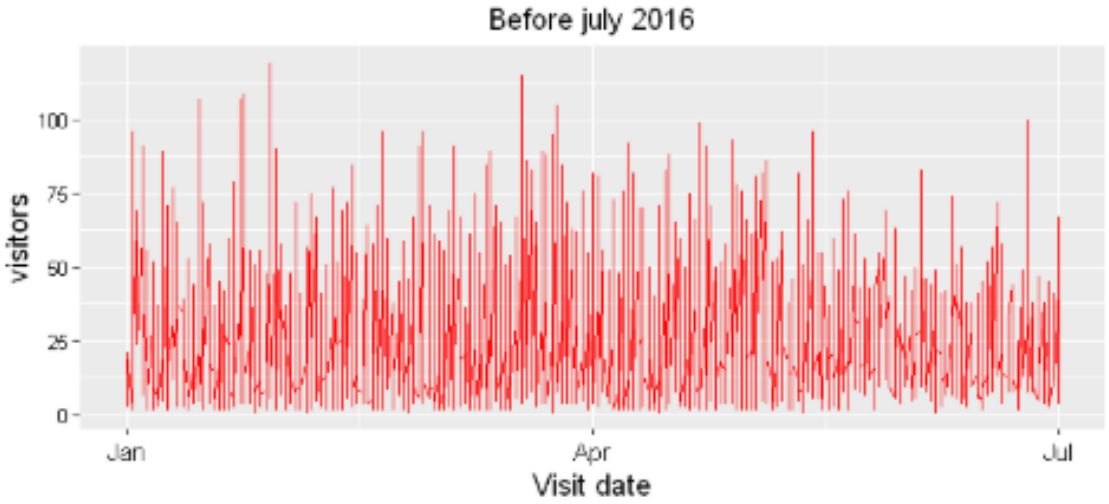
1.Overall Visitors

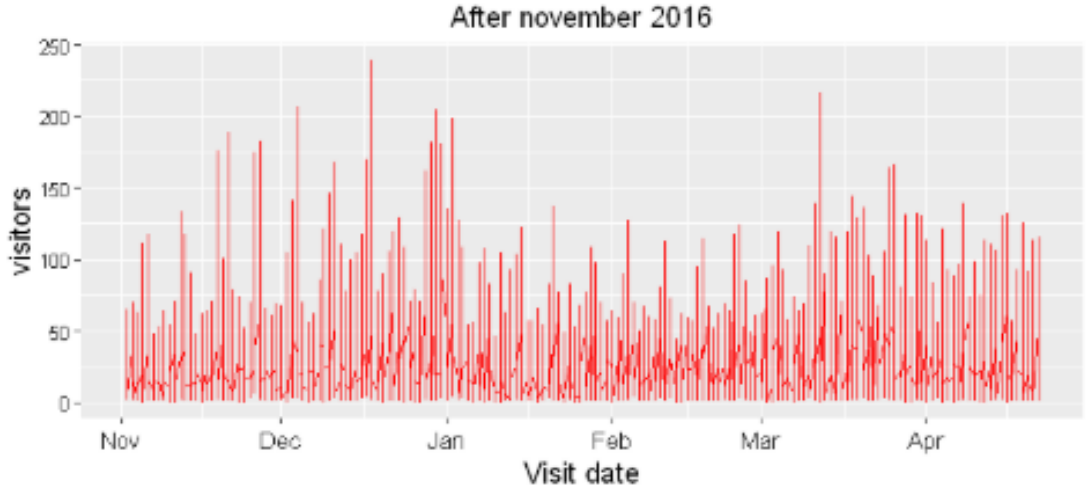
Code:

```
options(repr.plot.width = 13, repr.plot.height = 6)
plot1 <- ggplot(air_visit_data4, aes(x = visit_date, y= visitors)) +
         geom_line(col = 'red') +
         labs( x = 'Visit date', y = 'visitors', title = 'Overall visitors') +
         theme(plot.title = element_text(hjust = 0.5, size = 12),
         axis.title = element_text(size = 12),
         axis.text.x = element_text(size = 10),
         axis.text.y = element_text(size = 8),
         axis.text = element_text(colour = 'black'))
# before september 20, 2016
plot2 <- ggplot(air_visit_data4 %>% filter(visit_date <= '2016-07-01'),
                aes(x = visit_date, y= visitors)) +
         geom_line(col = 'red') +
         labs( x = 'Visit date', y = 'visitors', title = 'Before july 2016') +
         theme(plot.title = element_text(hjust = 0.5, size = 12),
         axis.title = element_text(size = 12),
         axis.text.x = element_text(size = 10),
         axis.text.y = element_text(size = 8),
         axis.text = element_text(colour = 'black'))

# after september 20, 2016
plot3 <- ggplot(air_visit_data4 %>% filter(visit_date > '2016-11-01') ,
aes(x = visit_date, y= visitors)) +
         geom_line(col = 'red') +
         labs( x = 'Visit date', y = 'visitors', title = 'After november 2016') +
         theme(plot.title = element_text(hjust = 0.5, size = 12),
         axis.title = element_text(size = 12),
         axis.text.x = element_text(size = 10),
         axis.text.y = element_text(size = 8),
         axis.text = element_text(colour = 'black'))
```

Output:







Interpretation:

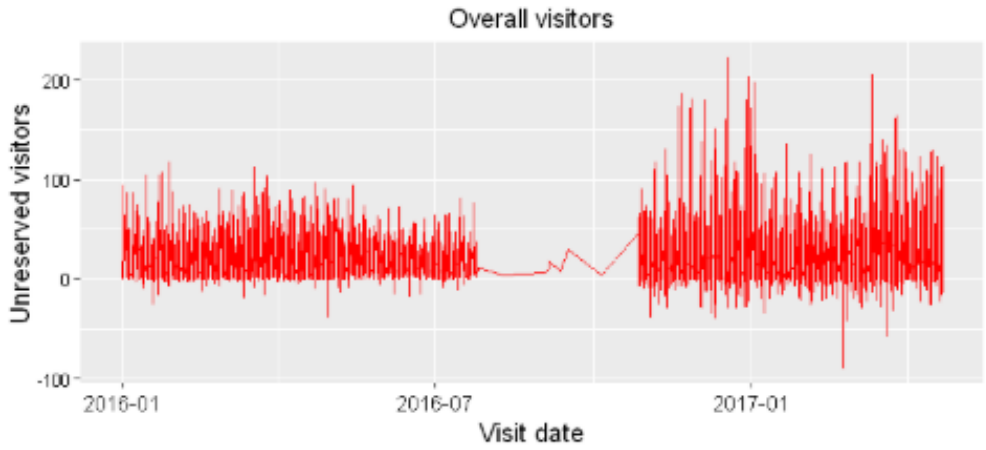
We see the irregularities in the visitors graph after july 2016 to November 2016. This might be related to new restaurants being added to the data base. In addition, we see a periodic pattern that most likely corresponds to a weekly cycle. The number of visitors increased after November 2016.

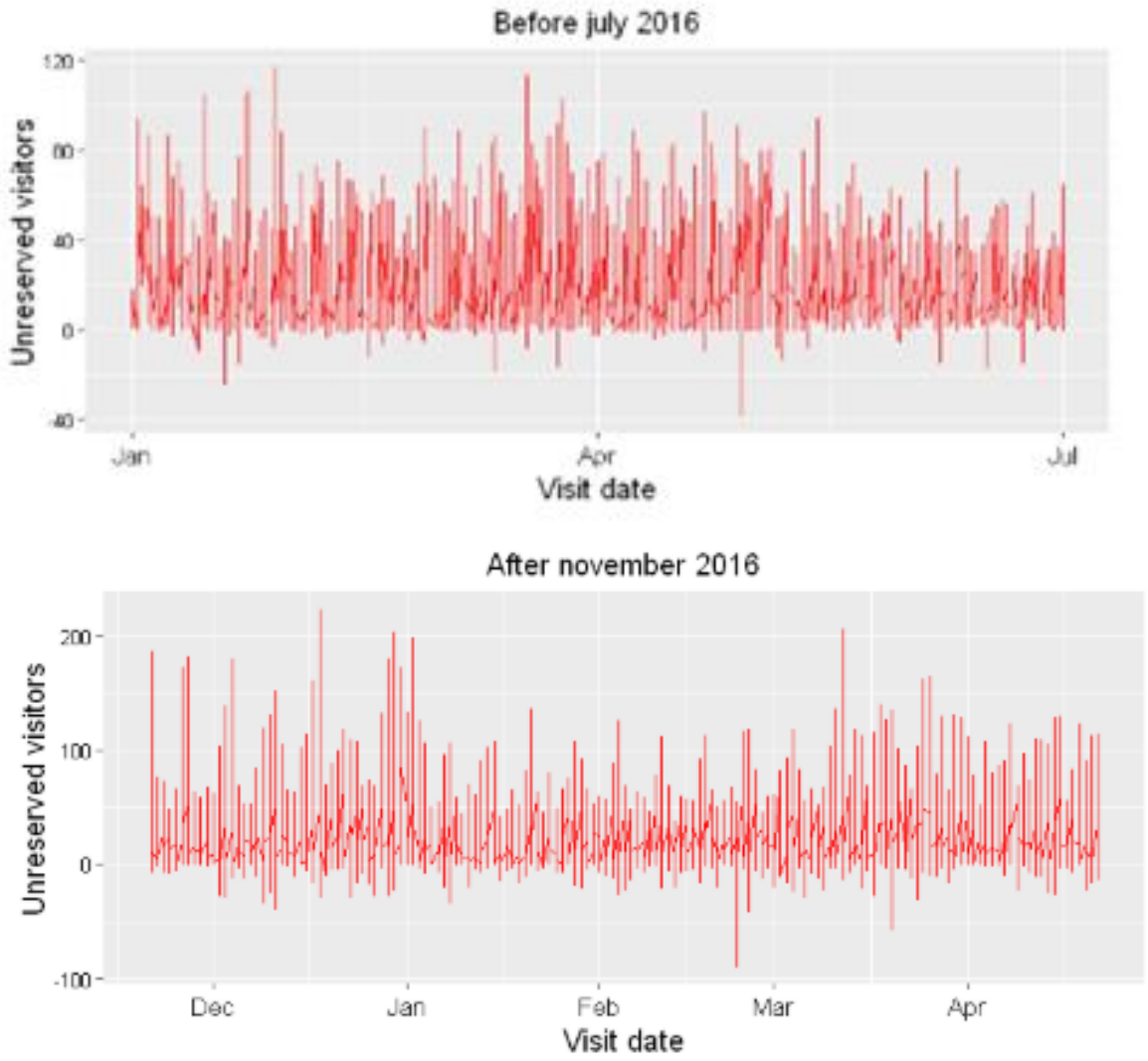
2.Unreserved Visitors

Code:

```
options(repr.plot.width = 13, repr.plot.height = 6)
plot1 <- ggplot(air_visit_data4, aes(x = visit_date, y= unreserved_visitors)) +
         geom_line(col = 'red') +
         labs( x = 'Visit date', y = 'Unreserved visitors', title = 'Overall visitors') +
         theme(plot.title = element_text(hjust = 0.5, size = 12),
         axis.title = element_text(size = 12),
         axis.text.x = element_text(size = 10),
         axis.text.y = element_text(size = 8),
         axis.text = element_text(colour = 'black'))
# before september 20, 2016
plot2 <- ggplot(air_visit_data4 %>% filter(visit_date <= '2016-07-01'),
            aes(x = visit_date, y= unreserved_visitors)) +
         geom_line(col = 'red') +
         labs( x = 'Visit date', y = 'Unreserved visitors', title = 'Before july 2016') +
         theme(plot.title = element_text(hjust = 0.5, size = 12),
         axis.title = element_text(size = 12),
         axis.text.x = element_text(size = 10),
         axis.text.y = element_text(size = 8),
         axis.text = element_text(colour = 'black'))
# after september 20, 2016
plot3 <- ggplot(air_visit_data4 %>% filter(visit_date > '2016-11-20') ,
        aes(x = visit_date, y= unreserved_visitors)) +
         geom_line(col = 'red') +
         labs( x = 'Visit date', y = 'Unreserved visitors', title = 'After november 2016') +
         theme(plot.title = element_text(hjust = 0.5, size = 12),
         axis.title = element_text(size = 12),
         axis.text.x = element_text(size = 10),
         axis.text.y = element_text(size = 8),
         axis.text = element_text(colour = 'black'))
grid.arrange(plot1, plot2,plot3, nrow = 2, ncol = 2)
```

Output





Interpretation:

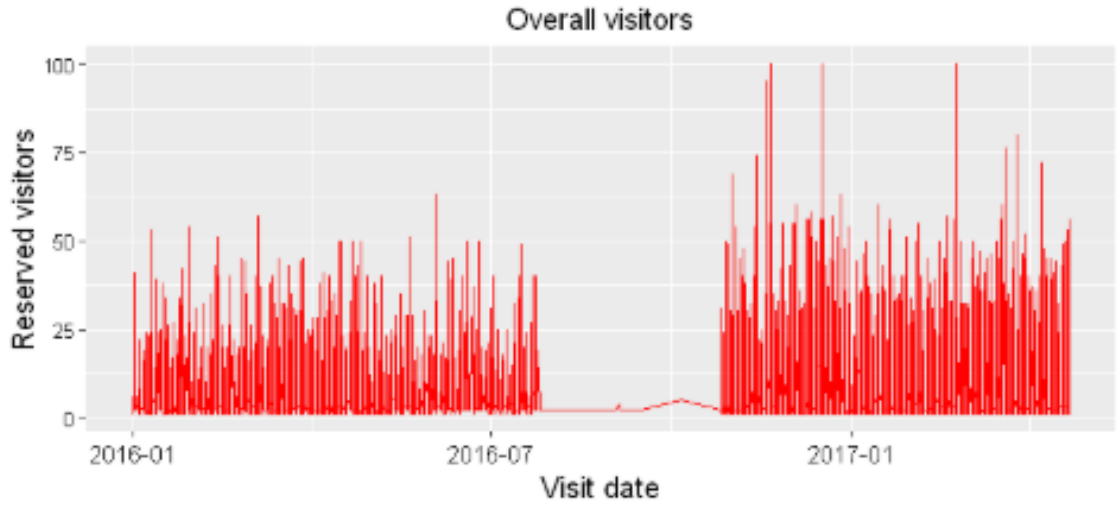
From above figure shows the plot unreserved visitors. Before July and after November 2016. Negative values in the unreserved visitor's column indicates people not showing up to restaurant after reserving the place. Also, number of unreserved visitors are more in before July 2016.

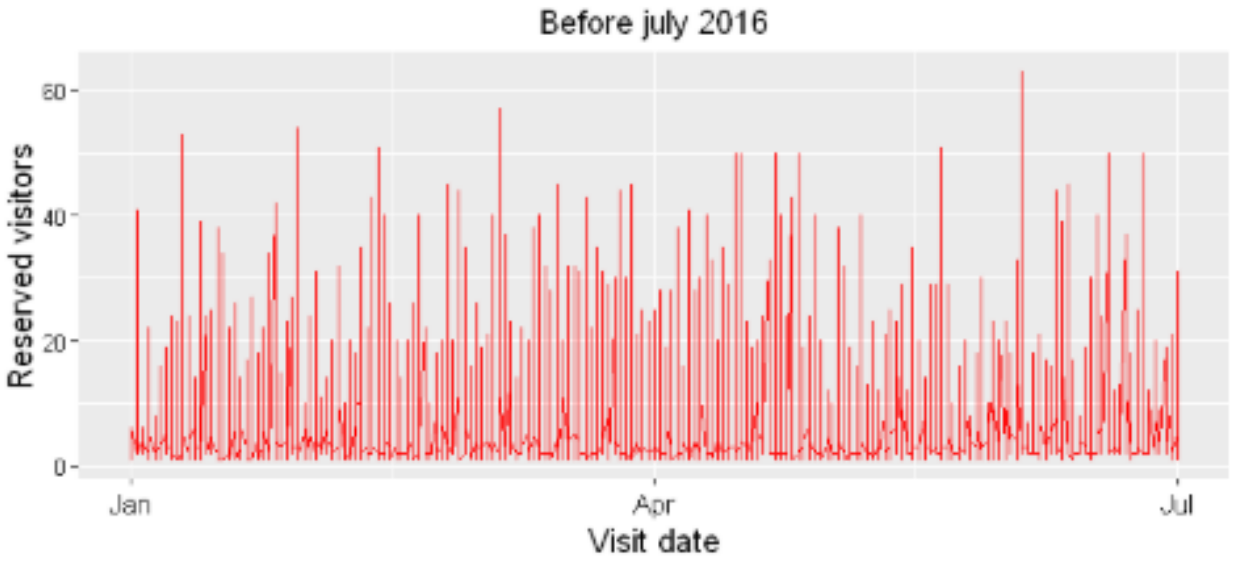
3. Reserved Visitors

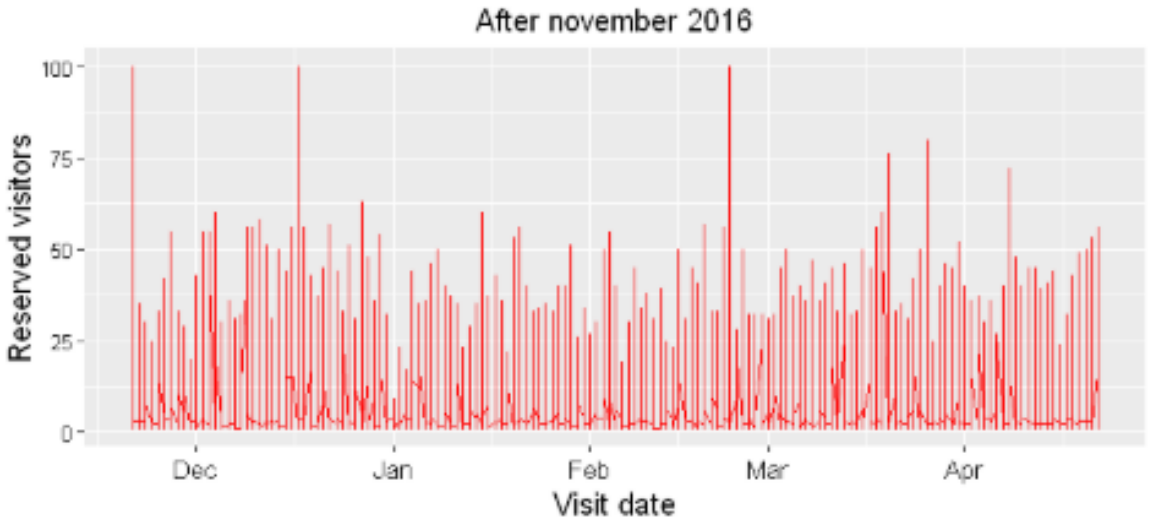
Code

```
options(repr.plot.width = 13, repr.plot.height = 6)
plot1 <- ggplot(air_visit_data4, aes(x = visit_date, y= reserved_visitors)) +
         geom_line(col = 'red') +
         labs( x = 'Visit date', y = 'Reserved visitors', title = 'Overall visitors') +
         theme(plot.title = element_text(hjust = 0.5, size = 12),
         axis.title = element_text(size = 12),
         axis.text.x = element_text(size = 10),
         axis.text.y = element_text(size = 8),
         axis.text = element_text(colour = 'black'))
# before september 20, 2016
plot2 <- ggplot(air_visit_data4 %>% filter(visit_date <= '2016-07-01'),
        aes(x = visit_date, y= reserved_visitors)) +
         geom_line(col = 'red') +
         labs( x = 'Visit date', y = 'Reserved visitors', title = 'Before july 2016') +
         theme(plot.title = element_text(hjust = 0.5, size = 12),
         axis.title = element_text(size = 12),
         axis.text.x = element_text(size = 10),
         axis.text.y = element_text(size = 8),
         axis.text = element_text(colour = 'black'))
# after september 20, 2016
plot3 <- ggplot(air_visit_data4 %>% filter(visit_date > '2016-11-20') ,
        aes(x = visit_date, y= reserved_visitors)) +
         geom_line(col = 'red') +
         labs( x = 'Visit date', y = 'Reserved visitors', title = 'After november 2016') +
         theme(plot.title = element_text(hjust = 0.5, size = 12),
         axis.title = element_text(size = 12),
         axis.text.x = element_text(size = 10),
         axis.text.y = element_text(size = 8),
         axis.text = element_text(colour = 'black'))

grid.arrange(plot1, plot2,plot3, nrow = 2, ncol = 2)
```

Output







Interpretation:

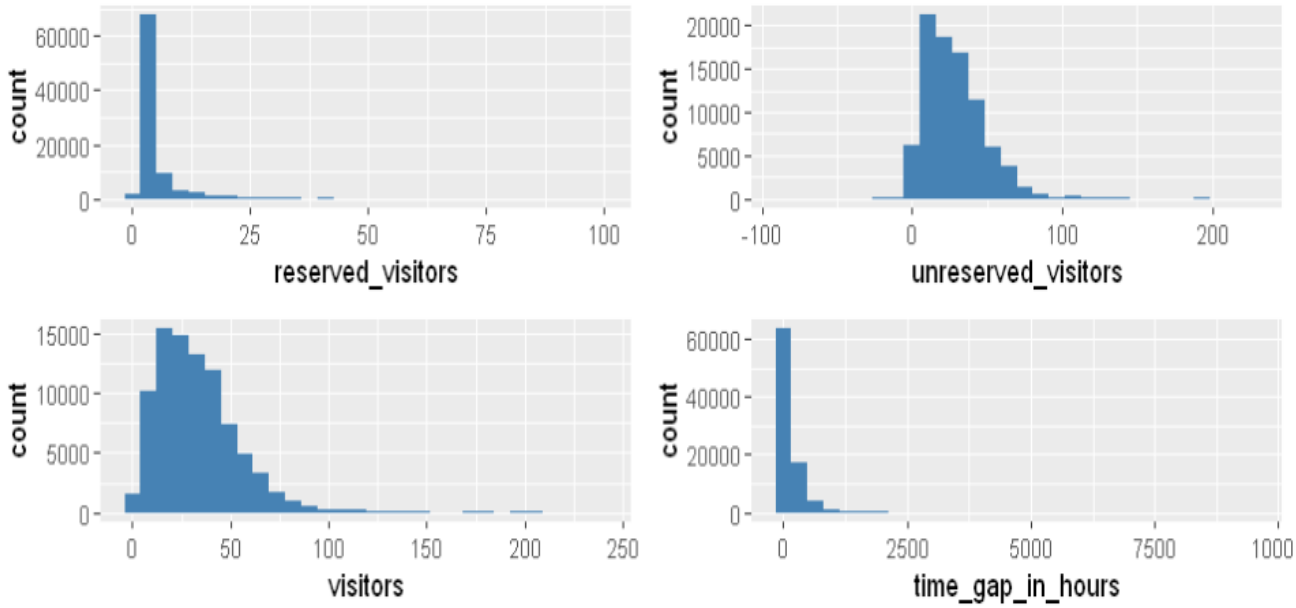
There were much fewer reservations made in 2016 through the **air** system the volume only increased during the end of that year. In 2017 the visitor numbers stayed strong.

4 Distribution of Visitors data

Code:

```
plot1 <- ggplot(air_visit_data4, aes(reserved_visitors)) +
         geom_histogram(fill = 'steelblue')
plot2 <- ggplot(air_visit_data4, aes(unreserved_visitors)) +
         geom_histogram(fill = 'steelblue')
plot3 <- ggplot(air_visit_data4, aes(visitors)) +
         geom_histogram(fill = 'steelblue')
plot4 <- ggplot(air_visit_data4, aes(time_gap_in_hours)) +
         geom_histogram(fill = 'steelblue')
```

Output



Interpretation

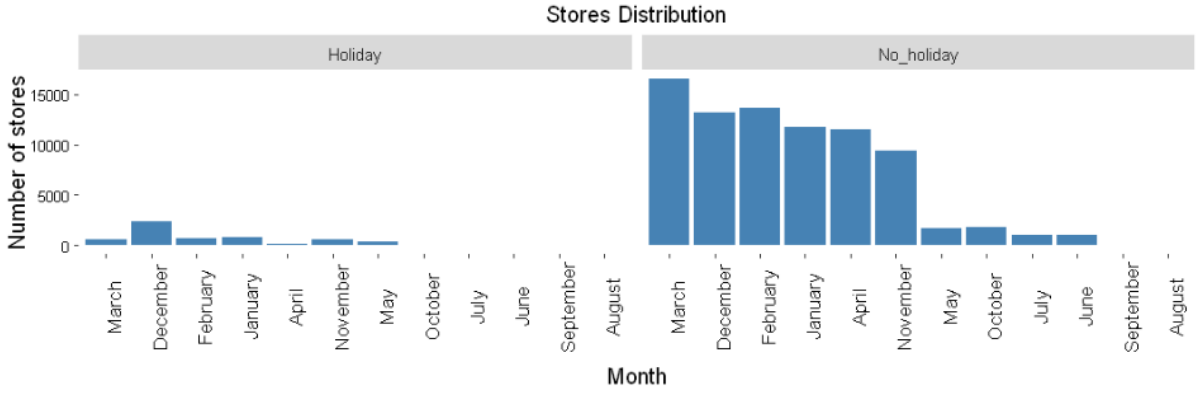
From Above figure we get the distribution of visitor's data. This helps us to understand the many visitors visits the restaurants without reservation.

5.Store Distribution

Code:

```
options(repr.plot.width = 9, repr.plot.height = 3)
g3 <- air_visit_data4 %>% group_by(Month = months.Date(visit_date))
ggplot(g3, aes(reorder(Month,Month,function(x)-length(x)))) +
geom_bar(fill = 'steelblue') +
labs( x = 'Month', y = 'Number of stores', title = 'Stores Distribution') +
theme(plot.title = element_text(hjust = 0.5, size = 12),
axis.title = element_text(size = 12),
axis.text.x = element_text(angle = 90, size = 10),
axis.text.y = element_text(size = 8),
panel.grid = element_blank(),
panel.background = element_blank(),
axis.text = element_text(colour = 'black')) +
facet_wrap(~holiday_flg)
```

Output:



Interpretation:

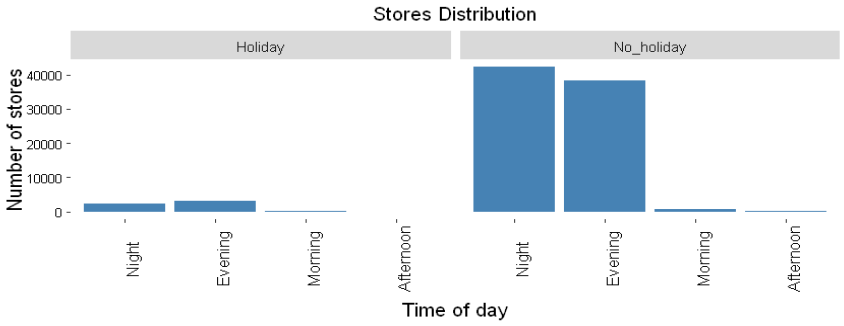
Store distribution graph shows that the number of stores get impacted by holiday during the month of December. However, without holiday, the number of stores are operated are more in the month of march 2016 to November 2017.

6.Store Distribution based on time of the day

Code

```
options(repr.plot.width = 7, repr.plot.height = 3)
g1 <- air_visit_data4 %>% group_by(visit_date)
ggplot(g1, aes(reorder(Time_of_day,Time_of_day,function(x)-length(x)))) +
geom_bar(fill = 'steelblue') +
labs( x = 'Time of day', y = 'Number of stores', title = 'Stores Distribution') +
theme(plot.title = element_text(hjust = 0.5, size = 12),
axis.title = element_text(size = 12),
axis.text.x = element_text(angle = 90, size = 10),
axis.text.y = element_text(size = 8),
panel.grid = element_blank(),
panel.background = element_blank(),
axis.text = element_text(colour = 'black')) +
facet_wrap(~holiday_flg)
```

Output



Interpretation

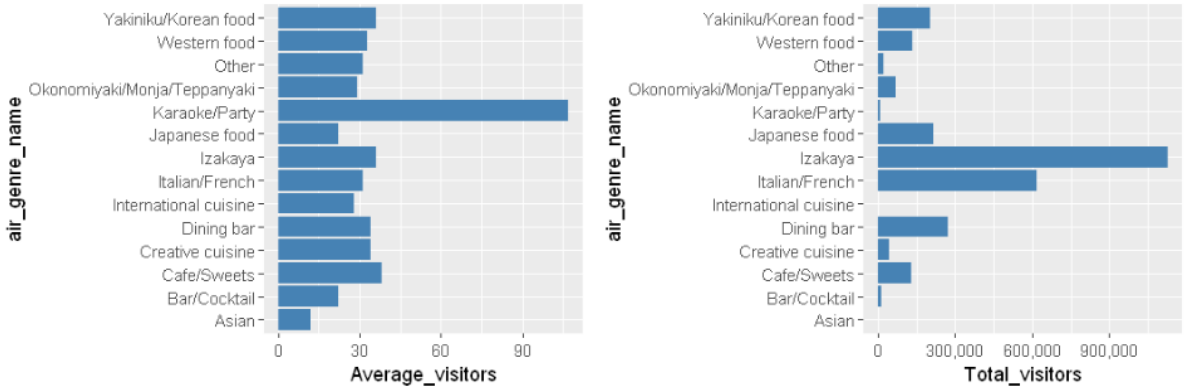
Above plot shows the distribution of stores based on the time of the day during holiday and non-holiday situation. The greater number of stores operated during the Night and the evening time.

7. Average Visitors based on Genere

Code

```
options(repr.plot.width = 9, repr.plot.height = 3)

plot1 <- ggplot(air_visit_data4 %>% group_by(air_genre_name) %>%
        summarise(Average_visitors = round(mean(visitors))) ,
                aes(air_genre_name, Average_visitors)) +
         geom_bar(stat = "identity",fill = 'steelblue') +
         coord_flip()

plot2 <- ggplot(air_visit_data4 %>% group_by(air_genre_name) %>%
        summarise(Total_visitors = sum(visitors)), aes(air_genre_name, Total_visitors)) +
         geom_bar(stat = "identity",fill = 'steelblue') +
         scale_y_continuous(labels = scales::comma) +
         coord_flip()

grid.arrange(plot1, plot2, nrow = 1, ncol = 2)

options(repr.plot.width = 9, repr.plot.height = 3)
```

Output



Interpretation:

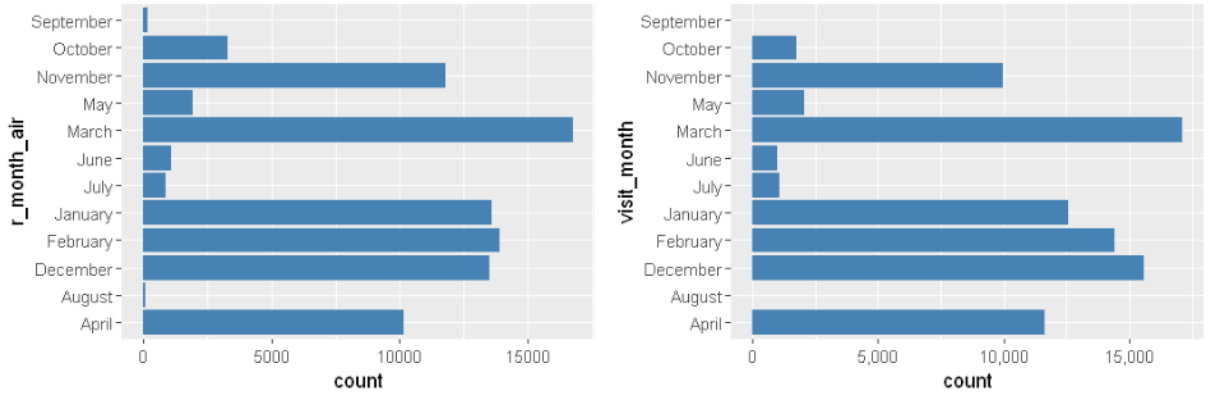
The above graph shows that the average number of visitors per genre. We can see that the during holiday number of visitors tend to be more for Karaoke/party than the other genere. Also, during non-Holiday time, Visitors prefer Izakaya.

8.Visitors count based on Month

Code:

```
plot1 <- ggplot(air_visit_data4, aes(r_month_air)) +
         geom_bar(fill = 'steelblue') +
         coord_flip()
plot2 <- ggplot(air_visit_data4, aes(visit_month)) +
         geom_bar(fill = 'steelblue') +
         scale_y_continuous(labels = scales::comma) +
         coord_flip()

grid.arrange(plot1, plot2, nrow = 1, ncol = 2)
```

Output



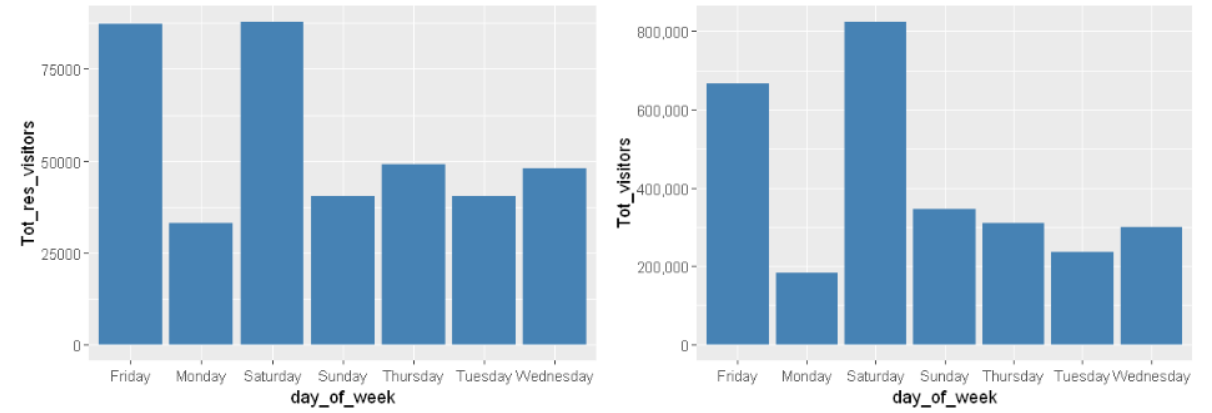
Interpretation:

The graph shows that during the year there is a certain amount of variation. Dec appears to be the most popular month for restaurant visits. The period of Mar - May is consistently busy

9.Number of visitors based on day of the week
Code

```
plot1 <- ggplot(air_visit_data4 %>% group_by(day_of_week) %>% summarise(Tot_res_visitors = sum(reserved_visitors)),
                aes(day_of_week, Tot_res_visitors)) +
         geom_bar(stat = 'identity' ,fill = 'steelblue')
plot2 <- ggplot(air_visit_data4 %>% group_by(day_of_week) %>% summarise(Tot_visitors = sum(visitors)),
                aes(day_of_week, Tot_visitors)) +
         geom_bar(stat = 'identity' ,fill = 'steelblue') +
         scale_y_continuous(labels = scales::comma)
```

Output



Interpretation

Friday and the weekend appear to be the most popular days; which is to be expected. Monday and Tuesday have the lowest numbers of average visitors.

Summary

1.Friday and the weekend appear to be the most popular days, which is to be expected. Monday and Tuesday have the lowest numbers of average visitors

2.Reservations are made typically for the dinner hours in the evening.

3.In the top 15 area we find again Tokyo and Osaka to be prominently present.

4.The same days were holidays in late Apr / May in 2016 as in 2017.

5.Visitors count increase from the month of March- May and it is highest in the month of December

4.Data Modelling

4.1Train-Test Split

Since there is a random fluctuation after July to November, we are going to use the data after September 2016 for forecasting. We are going to forecast average visitors

```
avg_visitors_data <- air_visit_data4 %>% filter(visit_date > '2016-11-01')
                     %>% group_by(visit_date) %>%
          summarise(avg_visitors = round(mean(visitors)))
train1 <- avg_visitors_data %>% filter(visit_date <= '2017-03-01')
test1 <- avg_visitors_data %>% filter(visit_date > '2017-03-01')

str(test)
```

4.2Data preparation for Tree models

Code:

```
#average visitors
ts <- xts(avg_visitors_data[,2], order.by = avg_visitors_data$visit_date) # time series
target <- dailyReturn(ts)
predictor <- lag(target, k = 1)
avg_data <- cbind(target,predictor)
colnames(avg_data) <- c('target', 'predictor')
avg_data <- na.exclude(avg_data)
head(avg_data,3)
str(avg_data)
#average visitors
train_data <- window(avg_data, end = '2017-03-01')
test_data <- window(avg_data, start = '2017-03-02')
head(train_data,3)
head(test_data,3)
```

Output:

```
               target   predictor
2016-11-03 -0.10000000  0.00000000
2016-11-04  0.03703704 -0.10000000
2016-11-05  0.28571429  0.03703704

An 'xts' object on 2016-11-03/2017-04-22 containing:
  Data: num [1:171, 1:2] -0.1 0.037 0.2857 -0.0278 -0.5143 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:2] "target" "predictor"
  Indexed by objects of class: [Date] TZ: UTC
  xts Attributes:
List of 1
 $ na.action: 'exclude' num 1
```

5. Implementation of Methods

5.1.ARIMA

As our objective is to predict Visitors, we will try the best method followed by analyst i.e. **ARIMA Model**. ARIMA is basically a short form commonly used for Autoregressive Integrated Moving Average where:

Auto Regressive (AR) – The lags of the differenced series

Moving Average (MA) - The lags of errors

I - The number of differences used to make the time series stationary

Assumption for a successful ARIMA Model

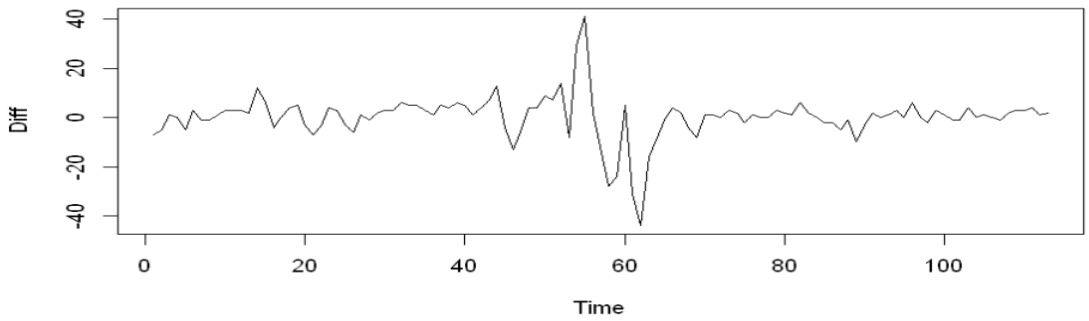
1. Data should be stationary
2. Data should be univariate

We have ARIMA model with a seasonal component because we have evidence of a seasonal trend in our data. We examined the differenced data when we have seasonality. Seasonality usually causes the series to be. When we run adf.test, we are testing to see if our data is stationary. Our null hypothesis is that the data is not stationary, and the alternative hypothesis is that the data is stationary.

Code

```
Diff <- diff(train1$avg_visitors, lag = 7)
plot.ts(Diff)
adf.test(Diff)
```

Output




```
        Augmented Dickey-Fuller Test

data:  Diff
Dickey-Fuller = -3.2957, Lag order = 4, p-value = 0.07545
alternative hypothesis: stationary
```

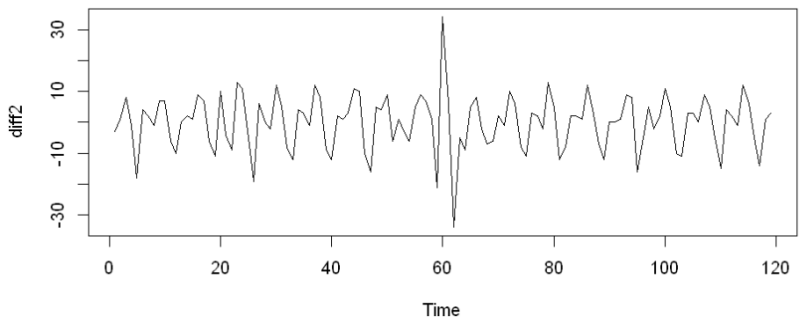
Interpretation

Looking at the plot and the output of our test, we have a p-value of .07545, which is higher than our significance level of .05. and we cannot reject the null hypothesis, Which means our data is not stationary and we still have a trend present. Therefore, we performed non-seasonal differencing.

Code:

```
diff2 <- diff(train1$avg_visitors, lag = 1)
plot.ts(diff2)
adf.test(diff2)
```

Output:



```
        Augmented Dickey-Fuller Test

data:  diff2
Dickey-Fuller = -10.499, Lag order = 4, p-value = 0.01
alternative hypothesis: stationary
```

Interpretation

We now have a p-value of .01, so we can reject the null and say that our data is stationary.

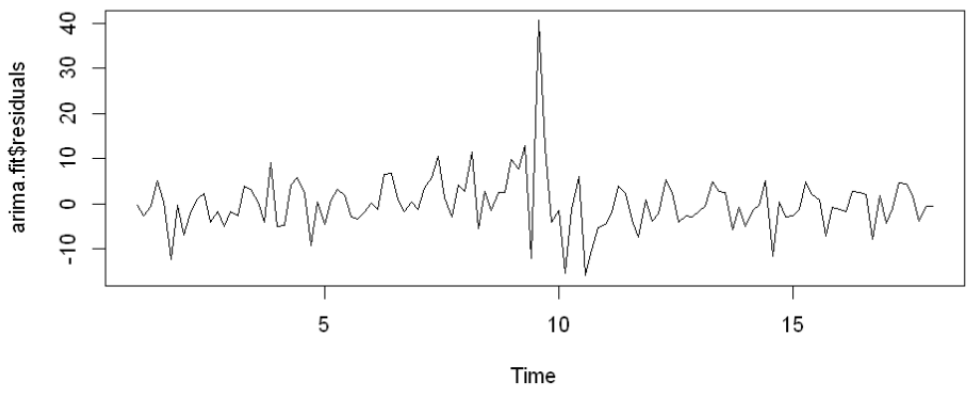
Code

```
arima.fit <- auto.arima(tsclean(ts(train1$avg_visitors, frequency = 7)),
                        stepwise = FALSE, approximation = FALSE)
AIC(arima.fit)
summary(arima.fit)
Box.test(arima.fit$residuals, type = "Ljung")
plot(arima.fit$residuals)
par(mfrow=c(2,1))
plot(forecast(arima.fit, h = 21), main = "auto.arima")
forecast(arima.fit, h = 21)
```

Output

```
Series: tsclean(ts(train1$avg_visitors, frequency = 7))
ARIMA(0,0,1)(1,0,0)[7] with non-zero mean

Coefficients:
         ma1    sar1     mean
      0.6962  0.4405  30.4162
s.e.  0.0598  0.0832   1.6753

sigma^2 estimated as 41.27:  log likelihood=-393.09
AIC=794.18   AICc=794.53   BIC=805.33

Training set error measures:
                     ME     RMSE      MAE       MPE     MAPE      MASE
Training set 0.04002186 6.343709 4.22554 -3.767292 13.94488 0.8134344
                    ACF1
Training set 0.05449304
```

```
        Box-Ljung test

data:  arima.fit$residuals
X-squared = 0.36532, df = 1, p-value = 0.5456
```



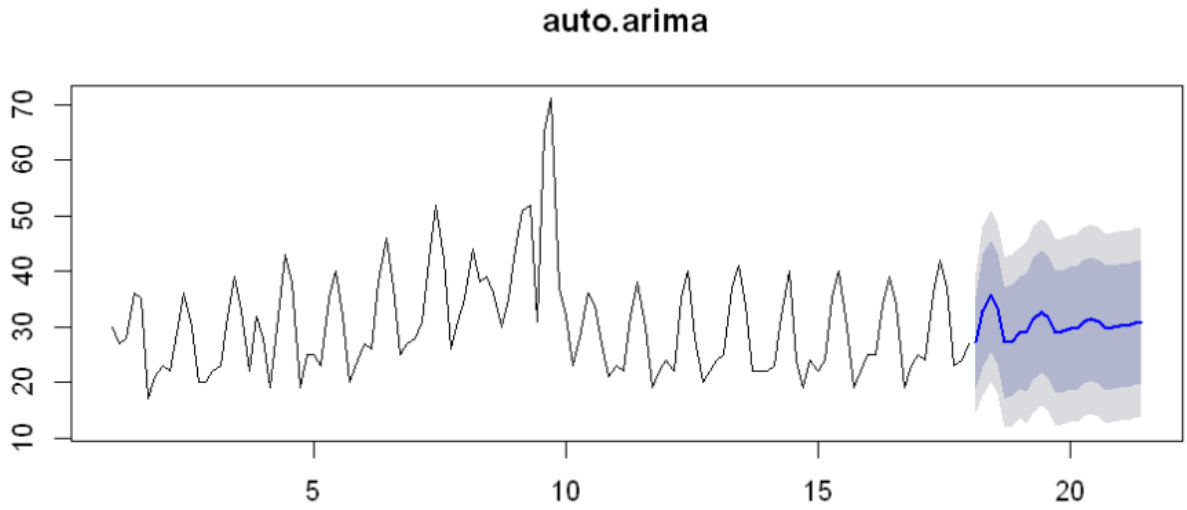

```
AIC(arima.fit)
```

794.178535356149

Forecast Output

Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95	
18.14286	27.15222	18.91886	35.38558	14.56038	39.74406
18.28571	32.87612	22.84397	42.90826	17.53328	48.21896
18.42857	35.51938	25.48724	45.55153	20.17654	50.86223
18.57143	33.31666	23.28452	43.34881	17.97382	48.65951
18.71429	27.14905	17.11690	37.18119	11.80620	42.49189
18.85714	27.58959	17.55744	37.62174	12.24675	42.93243
19.00000	28.91122	18.87908	38.94337	13.56838	44.25407
19.14286	28.97828	18.31056	39.64600	12.66341	45.29315
19.28571	31.49991	20.53739	42.46243	14.73419	48.26564
19.42857	32.66439	21.70187	43.62690	15.89866	49.43011
19.57143	31.69399	20.73147	42.65651	14.92826	48.45972
19.71429	28.97688	18.01436	39.93940	12.21116	45.74261
19.85714	29.17096	18.20844	40.13348	12.40524	45.93669
20.00000	29.75320	18.79068	40.71572	12.98747	46.51893
20.14286	29.78274	18.70438	40.86111	12.83984	46.72564
20.28571	30.89363	19.75955	42.02771	13.86552	47.92174
20.42857	31.40663	20.27255	42.54071	14.37853	48.43474
20.57143	30.97913	19.84505	42.11321	13.95103	48.00724
20.71429	29.78213	18.64805	40.91621	12.75402	46.81023
20.85714	29.86763	18.73355	41.00171	12.83952	46.89573
21.00000	30.12413	18.99005	41.25821	13.09602	47.15223
21.14286	30.13714	18.98083	41.29346	13.07504	47.19925
21.28571	30.62654	19.45947	41.79361	13.54798	47.70510
21.42857	30.85254	19.68547	42.01961	13.77398	47.93110

Visitors Forecast based on ARIMA



Interpretation

We run Box Ljung test to make sure our model is adequate. The p-value for Box Ljung test on the automated model is .5456, so we accept our null hypothesis and say that the model is adequate. This gives us point estimates and 80 and 95 percent confidence intervals for the average visitors per week

5.2. Exponential Smoothing

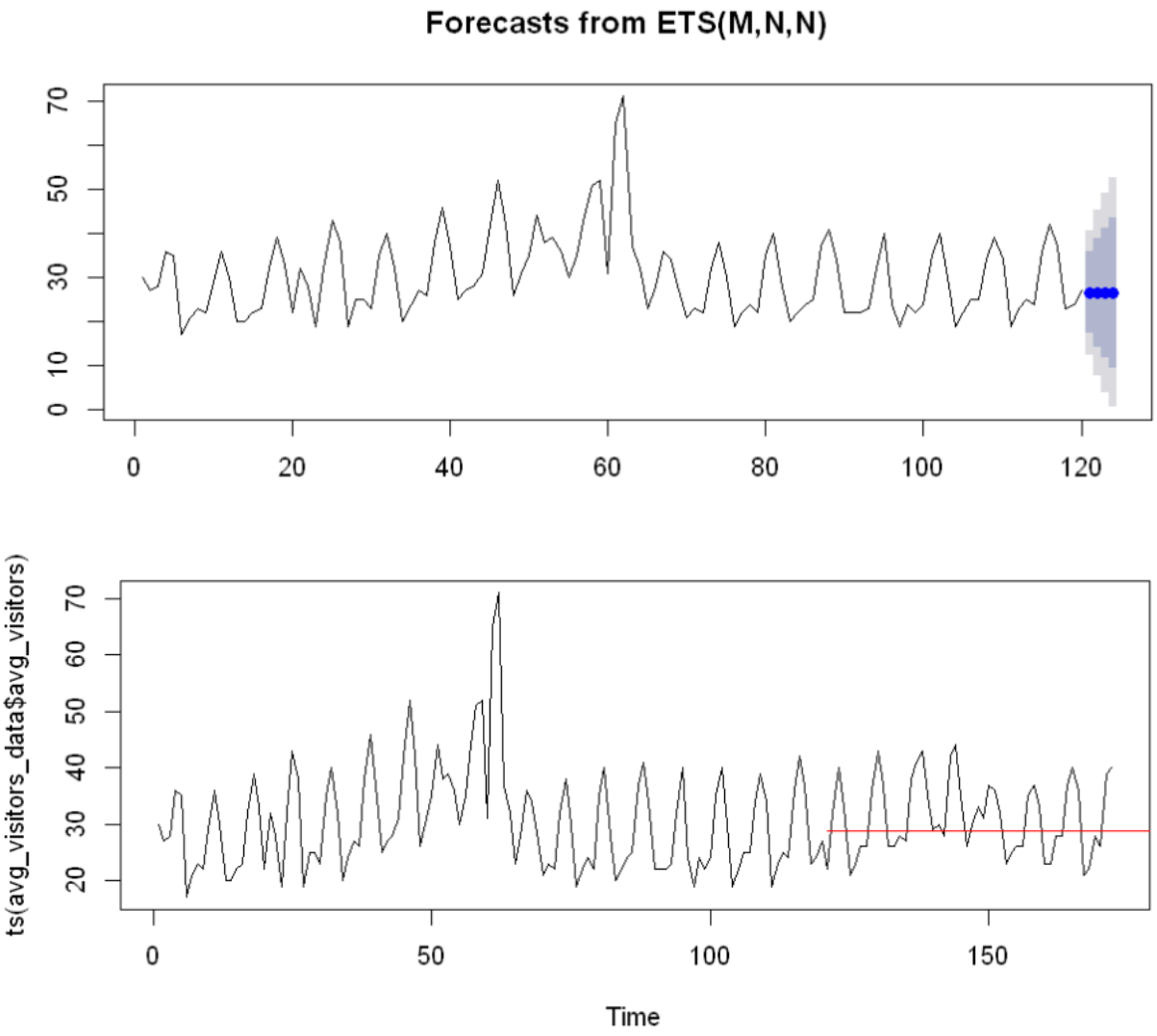
The ETS models with seasonality or non-damped trend or both have two-unit roots (i.e., they need two levels of differencing to make them stationary). All other ETS models have one-unit root (they need one level of differencing to make them stationary).

Code:

```
#ETS Modelling
ses_model1 <- ets(train1$avg_visitors)
summary(ses_model1)
accuracy(ses_model1)
#Forecasting
fcast_ses_model1 <- forecast(ses_model1, h = 4)
summary(accuracy(fcast_ses_model1))
plot(fcast_ses_model1)
accuracy(fcast_ses_model1)
```

Output

```
Call:
 ets(y = train1$avg_visitors)

  Smoothing parameters:
    alpha = 0.8378

  Initial states:
    l = 27.5771

  sigma:  0.2721

     AIC     AICc      BIC
1078.209 1078.416 1086.572

Training set error measures:
                     ME     RMSE      MAE       MPE     MAPE      MASE
Training set -0.01022291 8.931624 6.791438 -4.786338 23.11651 0.9965242
                    ACF1
Training set 0.1158078
```



	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	0.0288018	8.83002	7.213293	-6.889319	24.45353	1.058424	0.4014699

Interpretation:

A great advantage of the ETS statistical framework is that information criteria can be used for model selection.

Here the RMSE value is 8.83 which sugeest that we can use this model for prediction.

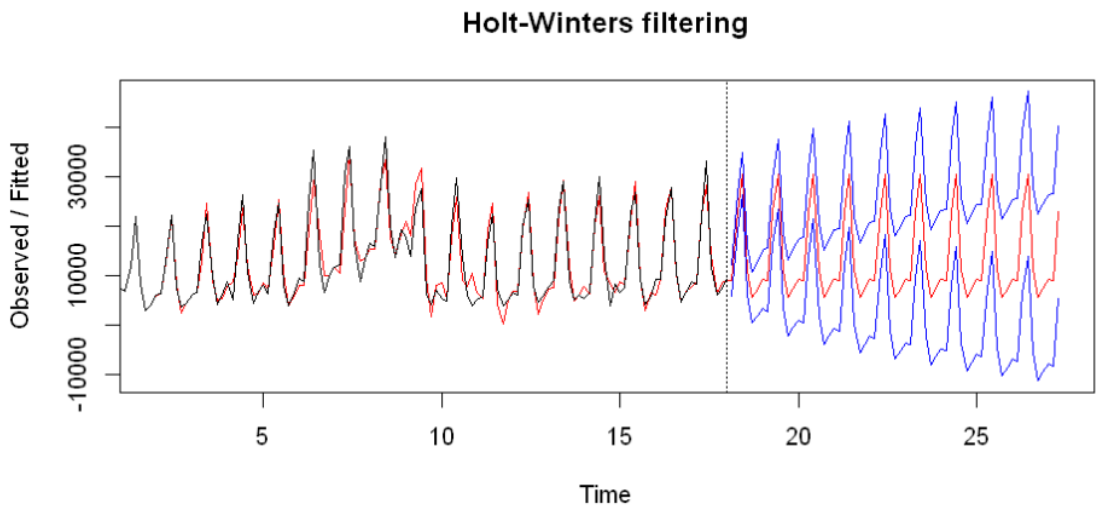
3. Holt-Winters method

The Holt-Winters forecasting algorithm allows users to smooth a time series and use that data to forecast.This method is used to capture seasonality.

Code

```
hw2 <- HoltWinters(tsclean(ts(train1$avg_visitors, frequency = 7)))
pred1 <- predict(hw2, n.ahead = 50, prediction.interval = T, level = 0.95)
plot(hw2, pred1)
sse2 <- hw2$SSE
mse2 <- sse2/nrow(train1)
rmse2 <- sqrt(mse2)
cat('Root mean squared error is ', rmse2)
```

Output




```
Root mean squared error is  6.63887
```

Interpretation

The Holt-Winters method can also be used for daily type of data, where the seasonal period is m= 7 and the appropriate unit of time for h= 7 is in days. Here, we generate daily forecasts. The model has identified the weekly seasonal pattern and the increasing trend at the end of the data, and the forecasts are a close match to the test data. Also, the seasonal variation in the data increases as the level of the series increases.

5.Decision tree

Decision tree regression is a supervised machine learning technique that is used for predicting the target variable by optimal recursive binary splitting of output target and input predictor variables into an incrementally smaller node. Tree pruning and time series cross-validation are used for lowering variance error generated by greater model complexity.

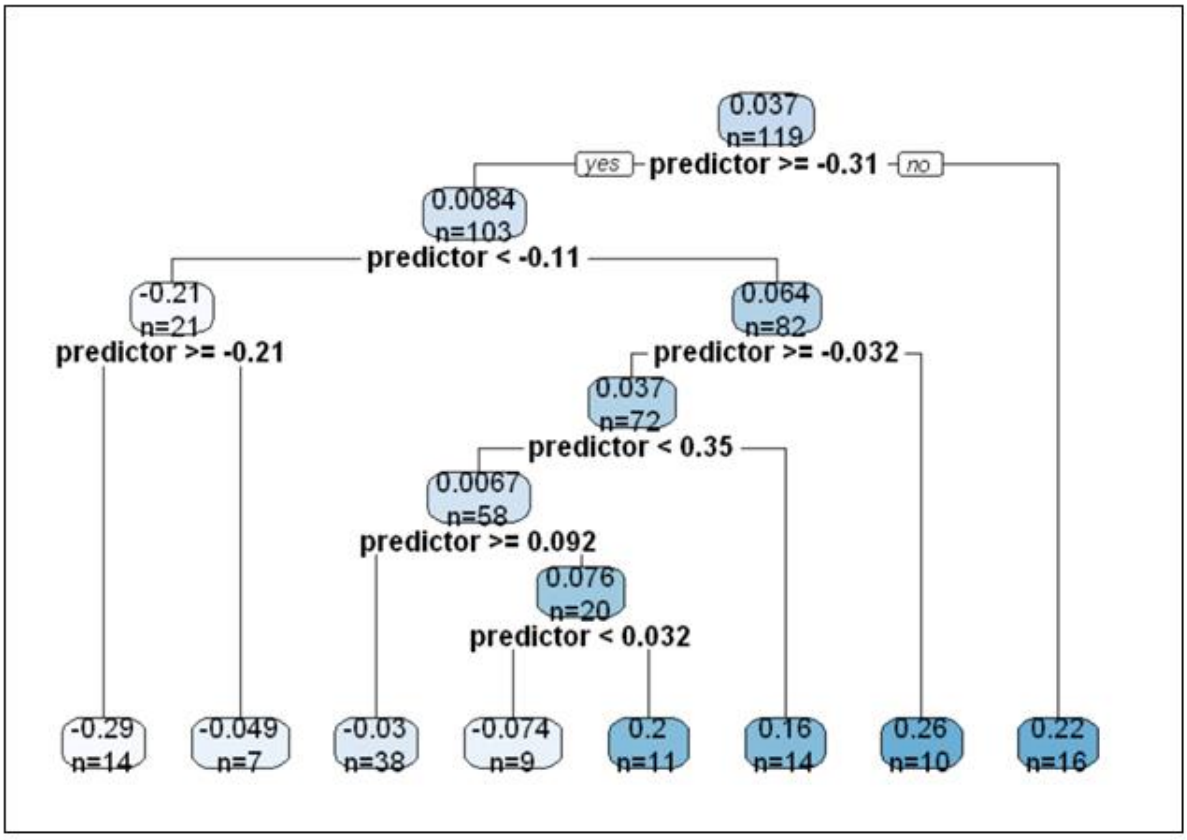
Model Building:

```
avg_d_tree <- rpart(target ~ predictor, train_data) #average visitors model
summary(avg_d_tree)
```

```
Call:
rpart(formula = target ~ predictor, data = train_data)
  n= 119

          CP nsplit rel error    xerror      xstd
1 0.10318271      0 1.0000000 1.0086312 0.1552718
2 0.04866632      2 0.7936346 0.9437847 0.1530680
3 0.03016781      3 0.7449682 0.9285473 0.1417213
4 0.02995541      4 0.7148004 0.9046057 0.1396684
5 0.02882080      5 0.6848450 0.9046057 0.1396684
6 0.01000000      7 0.6272034 0.9149270 0.1440258

Variable importance
predictor
      100
```

```
#avg visitors
options(repr.plot.width = 7, repr.plot.height = 4.5)
rpart.plot(avg_d_tree,extra = 1)
```

Graph:



Prediction:

```
# prediction
avg_test_pred = predict(avg_d_tree, test_data)
sse = sum((avg_test_pred - test_data$target)^2)
mse = sse/nrow(train_data)
rmse = sqrt(mse)
cat('Root mean squared error is',round(rmse,3))
```

```
Root mean squared error is 0.165
```

Interpretation:

Decision tree Regressor is the regression equivalent of the Decision tree algorithm.

Decision tree algorithm comes under **rpart** package in R. We first changed the data to time series using **xts** function from **quantmod** package. We build a decision tree model to predict the average visitors and got the root mean squared error of **1.801** in test data for next 52 days from November 2016 in Japanese restaurants.

4.Random Forest

Random forest regression is a ensemble learning method for predicting the output target variable average by bootstrap aggregation or bagging of independently build decision trees. Bagging is used for lowering variance error of the decision trees.

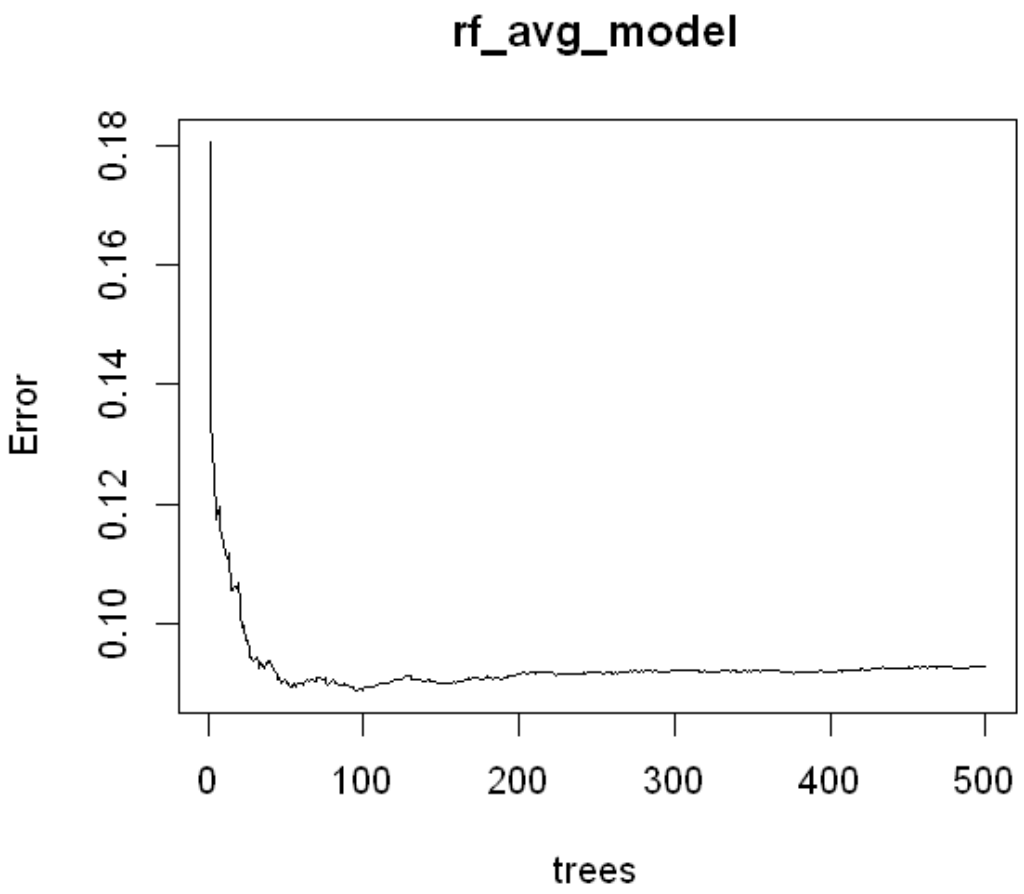
Model Building:

Code:

```
#average visitors model
rf_avg_model <- randomForest(target ~ predictor, train_data, ntree = 500, nodesize = 2, replace = T)
rf_avg_model$predicted[1:10] # top 10 predicted value from train data
options(repr.plot.width = 5, repr.plot.height = 4.5)
plot1 <- plot(rf_avg_model)
# mean squared error of first ten trees
rf_avg_model$mse[1:10]
# average visitors
avg_test_pred1 = predict(rf_avg_model, test_data)
sse5 = sum((avg_test_pred1 - test_data$target)^2)
mse5 = sse5/nrow(train_data)
rmse5 = sqrt(mse5)
cat('Root mean squared error is', round(rmse5,3))
```

Output:

2016-11-03	0.077411066400577
2016-11-04	-0.252353916820986
2016-11-05	0.216159343744875
2016-11-06	-0.0459786346860211
2016-11-07	0.450473344687199
2016-11-08	0.266070062618258
2016-11-09	-0.19643007037996
2016-11-10	-0.129542423981554
2016-11-11	0.43296046142346
2016-11-12	-0.0291942441123203




```
0.180581407488539  0.135636281260707  0.128190321768595  0.125591129386966  0.117331282072351  0.118743013326027
0.119708532769102  0.116104335885168  0.114919603251264  0.113883121260436

Root mean squared error is 0.178
```

Interpretation:

From the fig we can see that mean squared error decrease as we increase the number of decision trees and reaches the lowest point on the range of 70 to 80 trees. We used this random forest model to predict the average visitors demand in the Japanese restaurants and got the root mean squared error of 1.947 in the test data which forecasting for next 52 days from November 2016.

6.Conclusion

Model	Average Vistors RMSE
Decision tree	0.165
Random Forest	0.179
ARIMA	6.345
Ets	8.931
HoltWinter	6.639

From the above table we can observed that our time series models, and other machine learning models are performed well in predicting the average visitors demand on the test data for Japanese restaurants that has next 52 days demand from November 2016. Decision tree and Random Forest performed better with Root mean squared error of **0.165** and **0.179** respectively than the classic time series models like ARIMA, Exponential smoothing and Holt winter method which has the RMSE of **6.345, 8.931** and **6.639** respectively.

7.Reference

1.Gardner, E. S. (1985). Exponential smoothing: The state of the art. Journal of Forecasting, 4(1), 1–28. https://doi.org/10.1002/for.3980040103

2. Time Series Analysis - an overview | ScienceDirect Topics. (2020). Retrieved 14 February 2020, from https://www.sciencedirect.com/topics/medicine-and-dentistry/time-series-analysis

3.Autoregressive Integrated Moving Average Models (ARIMA).(2020). Retrieved 14 February 2020, from http://forecastingsolutions.com/arima.html

4. Tanizaki, Takashi & Shimmura, Takeshi & Takenaka, Takeshi. (2018). Demand forecasting in restaurants using machine learning and statistical analysis.