Restaurant Final Project IT

Michael Rains, Samousa Fofana, Alec Dudognon

11/21/2021

Introduction

Running a thriving local restaurant is not always as charming as first impressions appear. There are often all sorts of unexpected troubles popping up that could hurt business. One common predicament is that restaurants need to know how many customers to expect each day to effectively purchase ingredients and schedule staff members. This forecast is not easy to make because many unpredictable factors affect restaurant attendance, like weather and local competition. It is even harder for newer restaurants with little historical data. In this task, you are challenged to use reservation and visitation data to predict the total number of visitors to a restaurant for future dates. This information will help restaurants be much more efficient and allow them to focus on creating an enjoyable dining experience for their customers.

Problem Statement

In this task, we are challenged to use reservation and visitation data to predict the total number of visitors to a restaurant for future dates.

We have a certain amount of information about these restaurants like reservations, dates with the number of customers and reservations , the location of the restaurant etc…

Our main objective will be to create different model that we will use to predict, (i.e. linear, decision tree and random forest model), and choose the model that seems the best one.

Analysis

First of all we start read the database from mysql using the library RMysql

library(RMySQL)

mydb <- dbConnect(MySQL(), user="newuser", password="will-20", dbname="ITDAproject", host="127.0.0.1")

dbListTables(mydb)

air_reserve <- dbGetQuery(mydb, " SELECT R.*, RI.* FROM air_reserve R left JOIN restaurant_info RI USING(ID); ")

air_visit <- dbGetQuery(mydb, " SELECT V.*, RI.* FROM air_visit V left JOIN restaurant_info RI USING(ID); ") date_info <- dbGetQuery(mydb," SELECT * FROM date_info; ") restaurant_info <- dbGetQuery(mydb," SELECT * FROM restaurant_info; ")

It's polite to let the database know when you're done

dbDisconnect(mydb)

library(tidyverse)

glimpse(air_reserve) glimpse(air_visit) glimpse(date_info) glimpse(restaurant_info)

We will now write the csv files that we will use in the rest of the analysis

write.csv(air_reserve,"reservation_data.csv") write.csv(air_visit,"visit_data.csv") write.csv(date_info,"date_info.csv") write.csv(restaurant_info,"restaurant_info.csv")

Step 1 - read in the dataset

```
reservations <- read.csv("reservation_data.csv")
visits <- read.csv("visit_data.csv")
date_info <- read.csv("date_info.csv")
restaurant_info <- read.csv("restaurant_info.csv")
```

All the relevant data is read in - we started from a point where we ran SQL queries to get the restaurant information into the reservations and visits tables but need to do some variable manipulation to align the dates to get the date information to a point where we can join as some datasets store the date in different ways. For this we will use the **lubridate** package in R.

Dataset

In this part we will become aware of the differents dataset we have at our disposal :

```
names(reservations)
```

```
##  [1] "X"                "ID"               "visit_datetime"   "reserve_datetime"
##  [5] "reserve_visitors" "ID..5"            "air_genre_name"   "air_area_name"
##  [9] "latitude"         "longitude"
```

First we have a dataset about the reservations in detail, with the number of expected reservations, the number of visitors with the date coupled with the restaurant ID.

```
names(visits)
```

```
## [1] "X"              "ID"             "visit_date"     "visitors"
## [5] "ID..4"          "air_genre_name" "air_area_name"  "latitude"
## [9] "longitude"
```

This table concerns the reservation part of our work. Indeed we find the ID of the restaurants with the corresponding data with the number of visitors.

```
names(date_info)
```

```
## [1] "X"             "calendar_date" "day_of_week"   "holiday_flg"
```

This dataset allows us to have information on the different dates contained in the reservation dataset. We can know the exact day and if it is a vacation.

```
names(restaurant_info)
```

```
## [1] "X"              "ID"             "air_genre_name" "air_area_name"
## [5] "latitude"       "longitude"
```

This dataset concerns the information specific to the restaurants we will analyze. We have an identification number (ID) for each restaurant and information about the type of restaurant and its location.

Load relevant packages for analysis

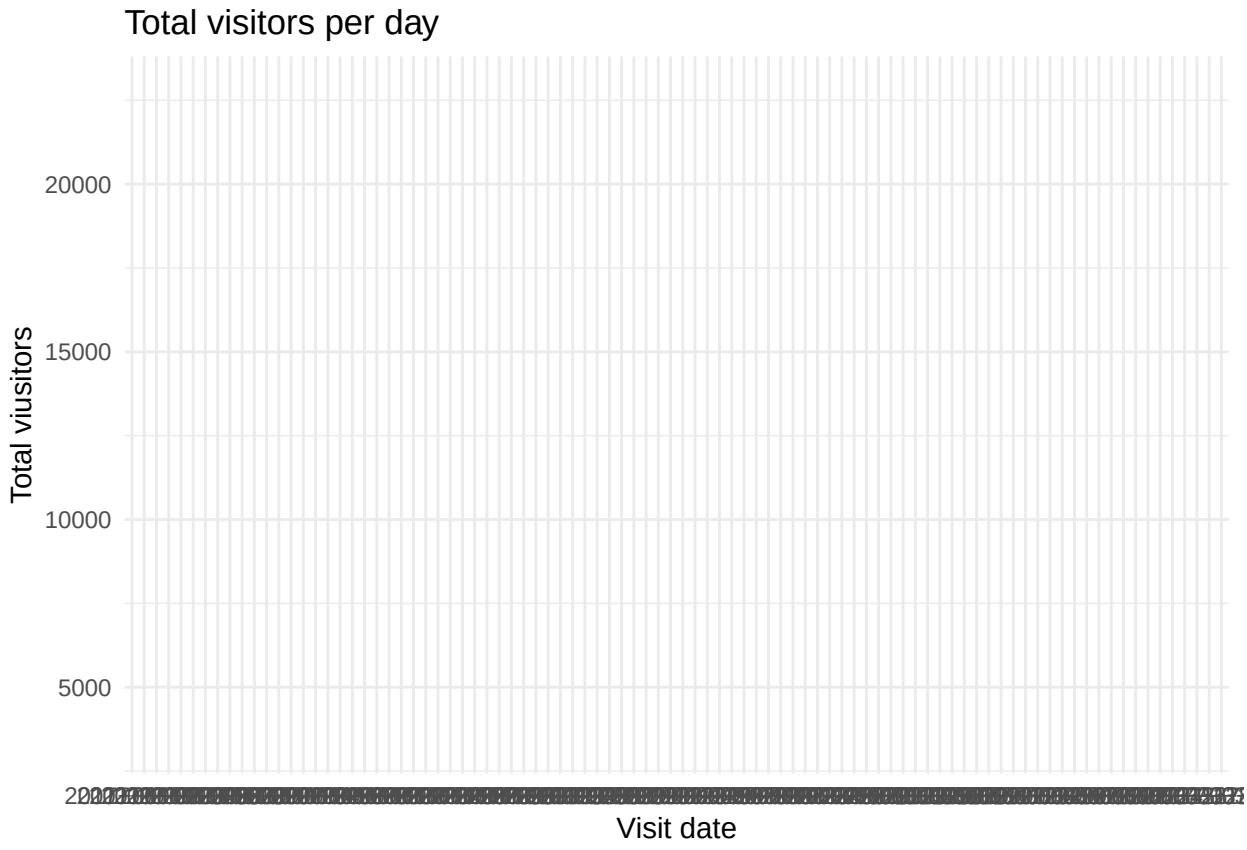
```
library(tidyverse)
library(lubridate)
library(leaflet)
```

Plot relate the different table

table visits

- The below graph will show us the overall total visitors per day

```
visits %>%
  group_by(visit_date) %>%
  summarise(all_visitors = sum(visitors)) %>%
  ggplot(aes(x=visit_date,y=all_visitors)) +
  geom_line(col = "blue") +
  labs(x = "Visit date", y = "Total viusitors",title = "Total visitors per day")+
  theme_minimal()
```

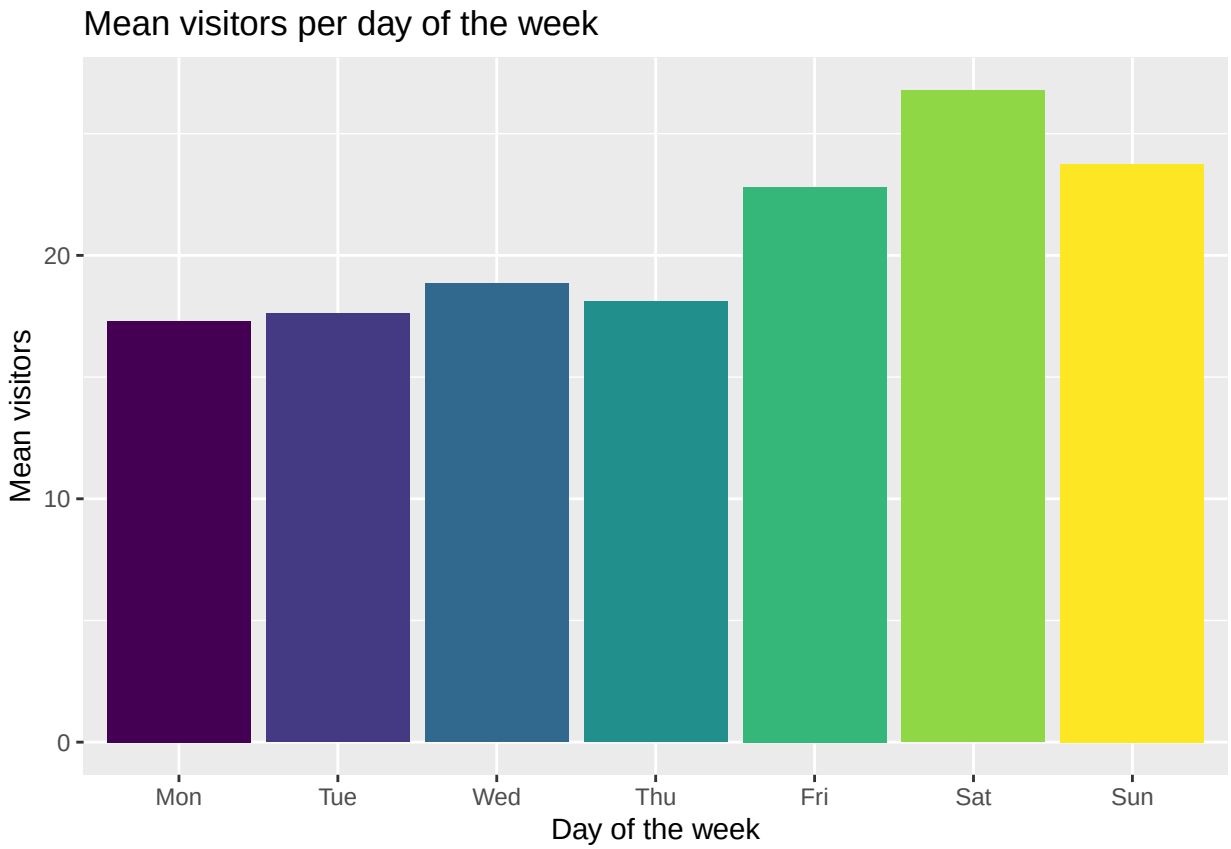


We can easily see that we have an increasing trend during the time, and we can also en-light the fact that during the week we always have a peak which is in correspondence of the weekend.

- The below graph will show the total mean visitors per day of the week

```
visits %>%
  mutate(week_day = wday(visit_date, label = TRUE, week_start = 1)) %>%
  group_by(week_day) %>%
  drop_na()%>%
```

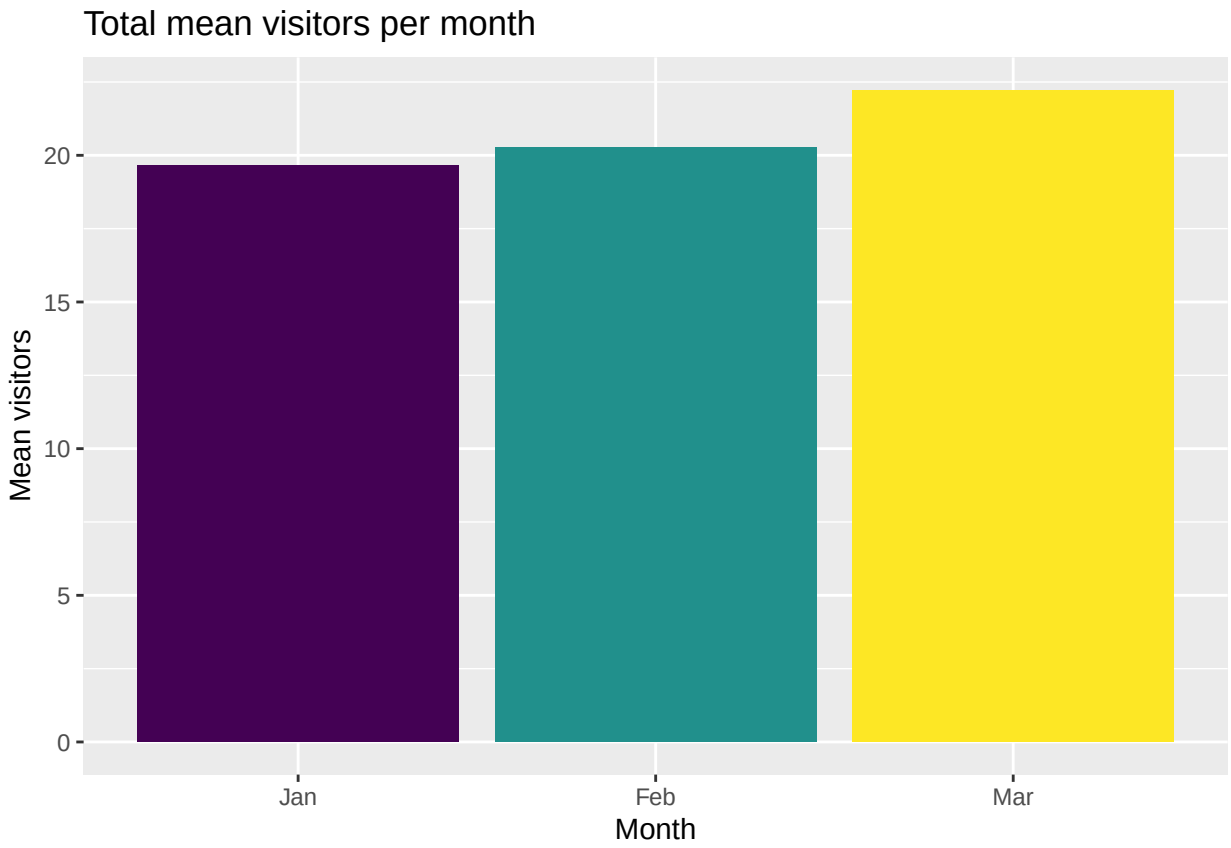
```
  summarise(visits = mean(visitors)) %>%
  ggplot(aes(week_day, visits, fill = week_day)) +
  geom_col() +
  theme(legend.position = "none") +
  labs(x = "Day of the week", y = "Mean visitors", title = "Mean visitors per day of the week")
```



From the result we can easily see that the day with the higher mean fall during the Friday, Saturday and Sunday. While the day with the lower mean are Monday and Tuesday.

- The below graph will show the total mean visitors per month

```
visits %>%
  mutate(month = month(visit_date, label = TRUE)) %>%
  group_by(month) %>%
  drop_na()%>%
  summarise(visits = mean(visitors)) %>%
  ggplot(aes(month, visits, fill = month)) +
  geom_col() +
  theme(legend.position = "none") +
  labs(x = "Month", y = "Mean visitors", title= "Total mean visitors per month")
```

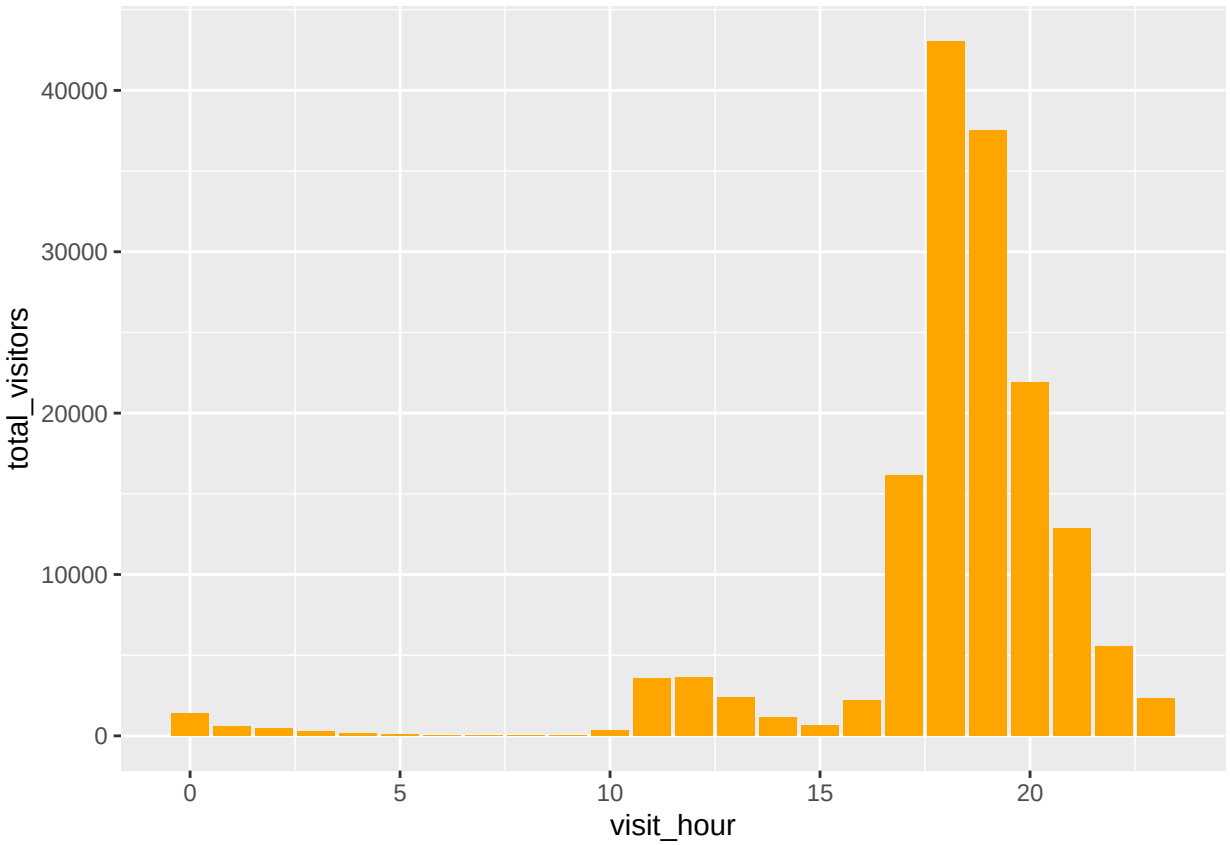


Reservation

```
reservations_graph <- reservations %>%
  mutate(reserve_date = date(reserve_datetime),
         reserve_hour = hour(reserve_datetime),
         reserve_wday = wday(reserve_datetime, label = TRUE, week_start = 1),
         visit_date = date(visit_datetime),
         visit_hour = hour(visit_datetime),
         visit_wday = wday(visit_datetime, label = TRUE, week_start = 1),

         )

reservations_graph %>%
  group_by(visit_hour) %>%
  drop_na() %>%
  summarise(total_visitors = sum(reserve_visitors)) %>%
  ggplot(aes(visit_hour, total_visitors)) +
  geom_col(fill="orange")
```



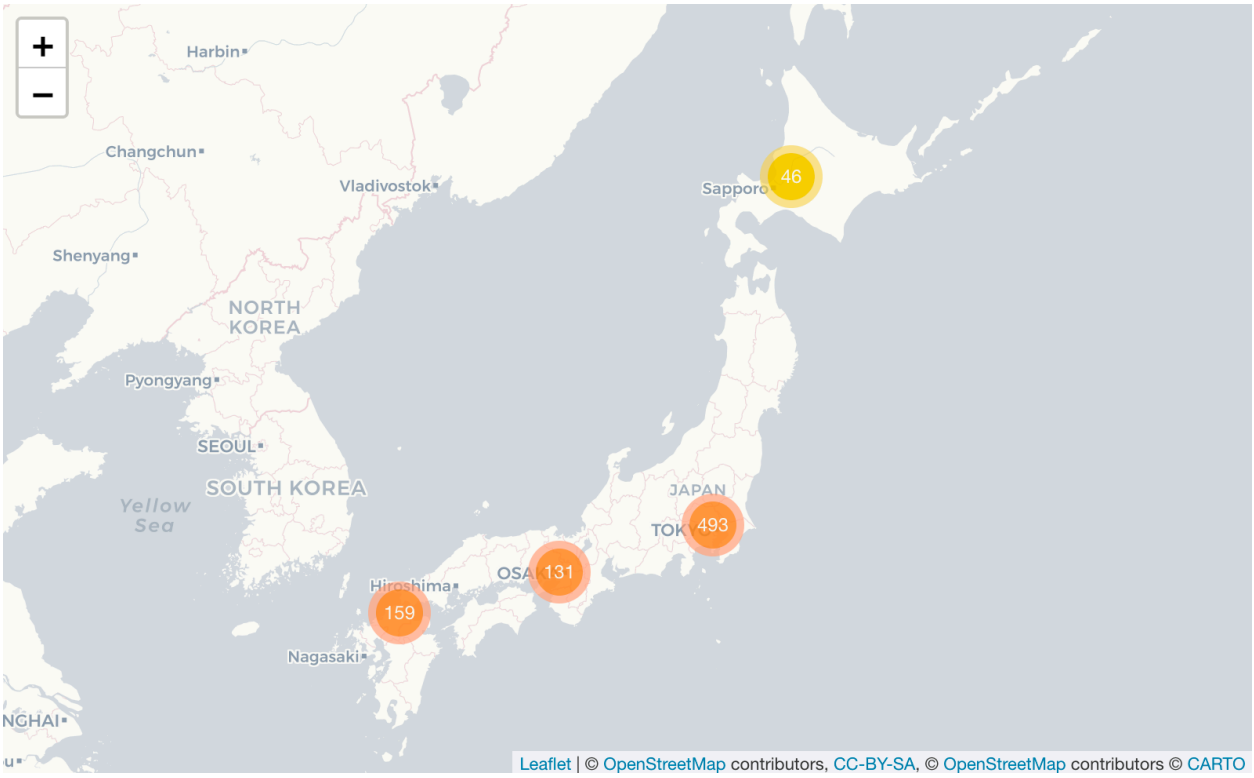
From this graph we can easily see that the majority of the client usually do reservation for the evening. Between hour five and ten we can see that we don't have value principally relate to the fact that restaurant at those time are closed.

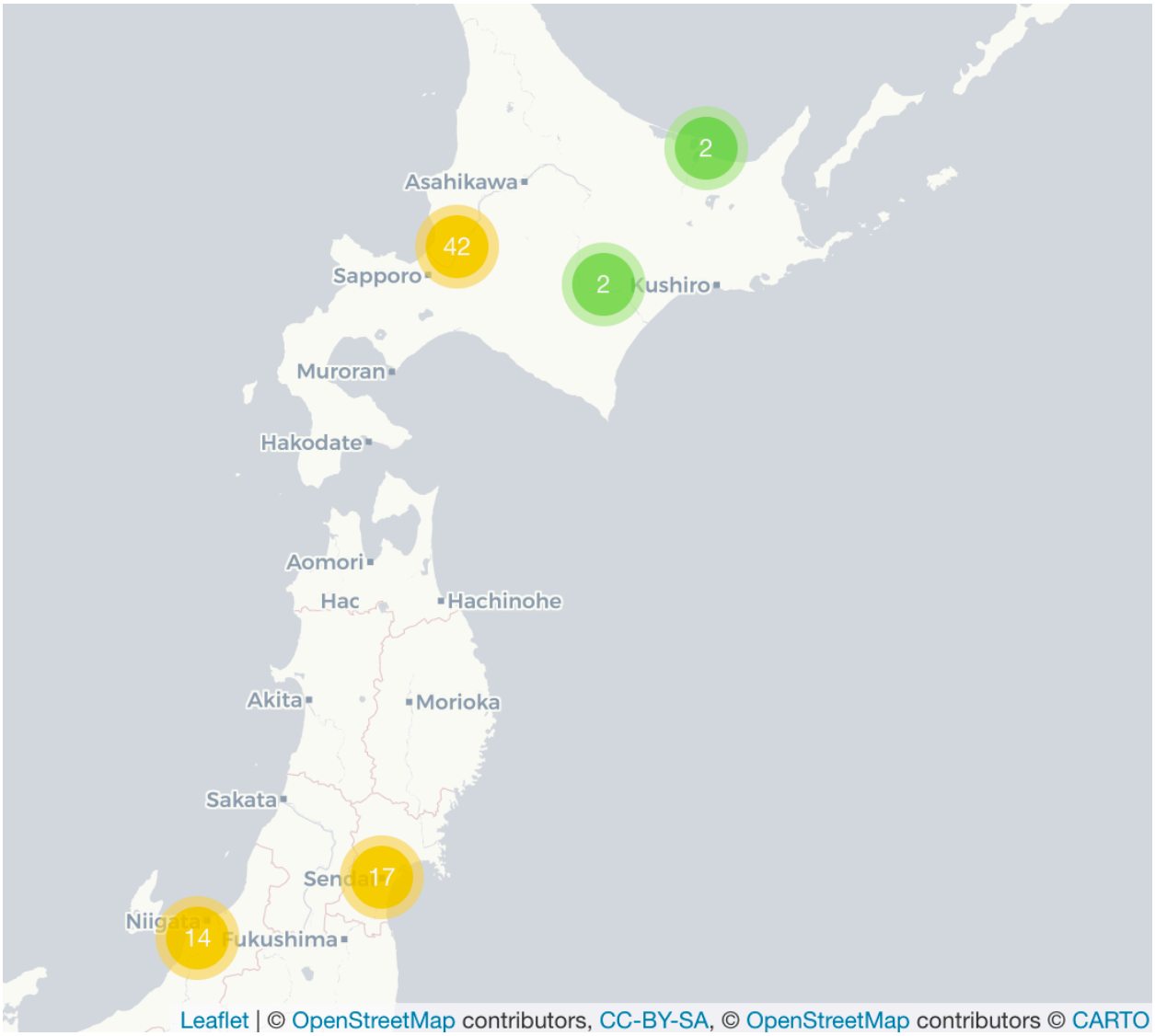
Restaurant Info

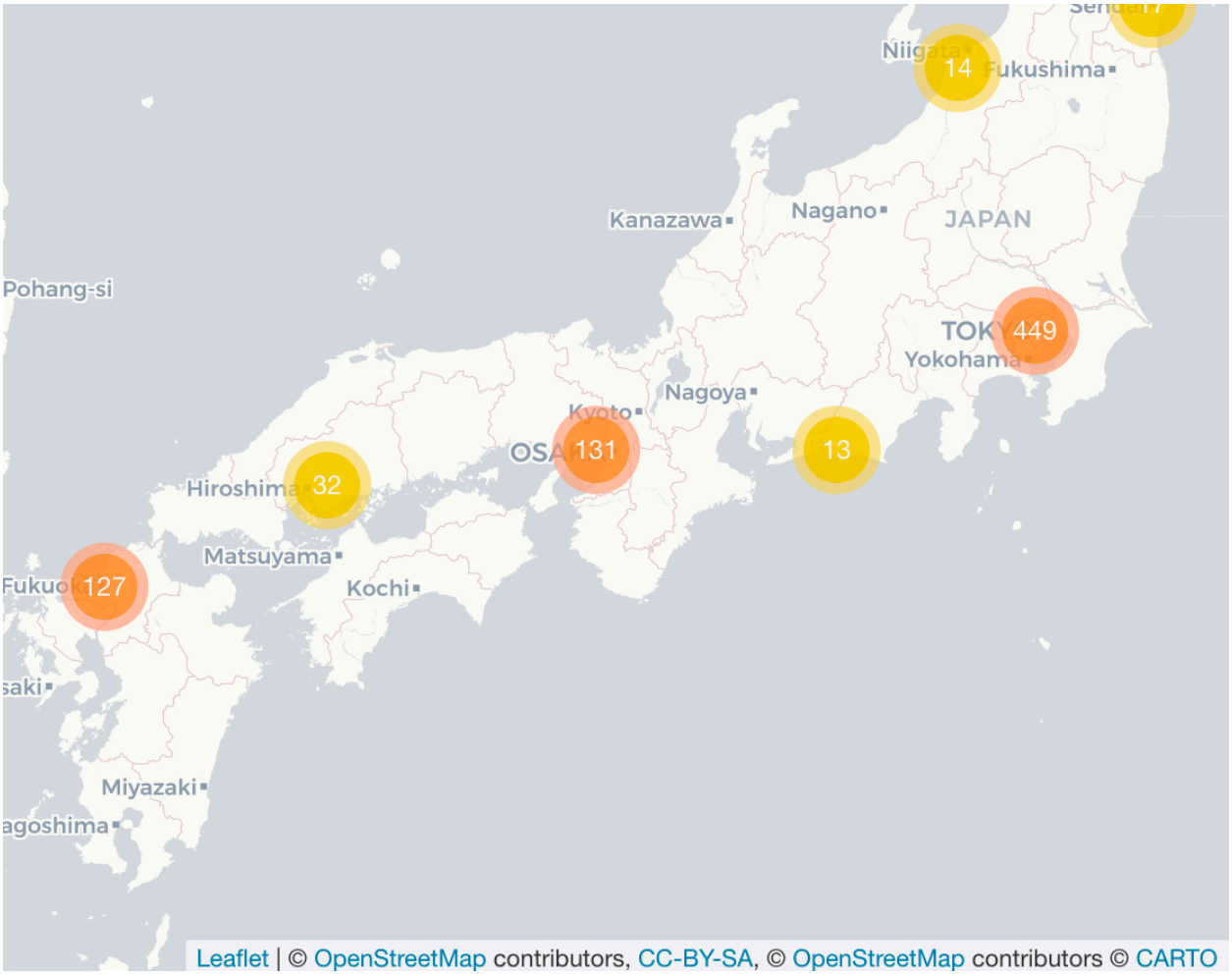
Plotting the position of the different restaurant

```
# leaflet(restaurant_info) %>%
#   addTiles() %>%
#   addProviderTiles("CartoDB.Positron") %>%
#   addMarkers(~longitude, ~latitude,
#              popup = ~ID, label = ~air_genre_name,
#              clusterOptions = markerClusterOptions())
```

The leaflet package only outputs HTML formats thus we will insert images from the interactive map we have generated


Harbin
Changchun
Vladivostok
Sapporo
46
Shenyang
NORTH KOREA
Pyongyang
SEOUL
SOUTH KOREA
Yellow Sea
JAPAN
TOK
493
OSA
131
Hiroshima
159
Nagasaki
NGHAI
Leaflet | © OpenStreetMap contributors, CC-BY-SA, © OpenStreetMap contributors © CARTO

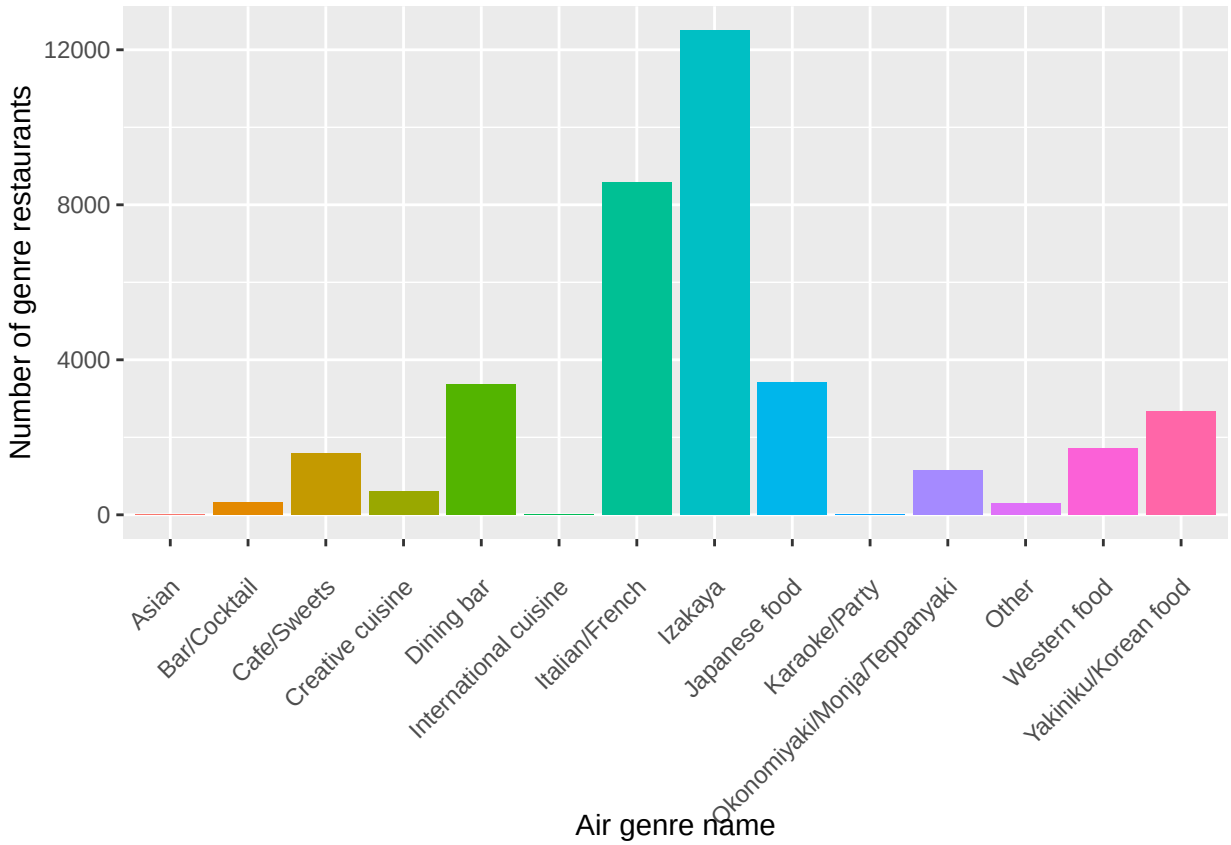

2
Asahikawa
42
Sapporo
2
Kushiro
Muroran
Hakodate
Aomori
Hac
Hachinohe
Akita
Morioka
Sakata
17
14
Fukushima
Leaflet | © OpenStreetMap contributors, CC-BY-SA, © OpenStreetMap contributors © CARTO



The map show us that many restaurants share common coordinates, since those coordinates refer to the area of the restaurant. By clicking on the clusters they will break into smaller clusters and at the end into the individual restaurants, which are labelled by their genre name . If we click on the single markers we will able to visualize the id of each restaurant in the cluster.

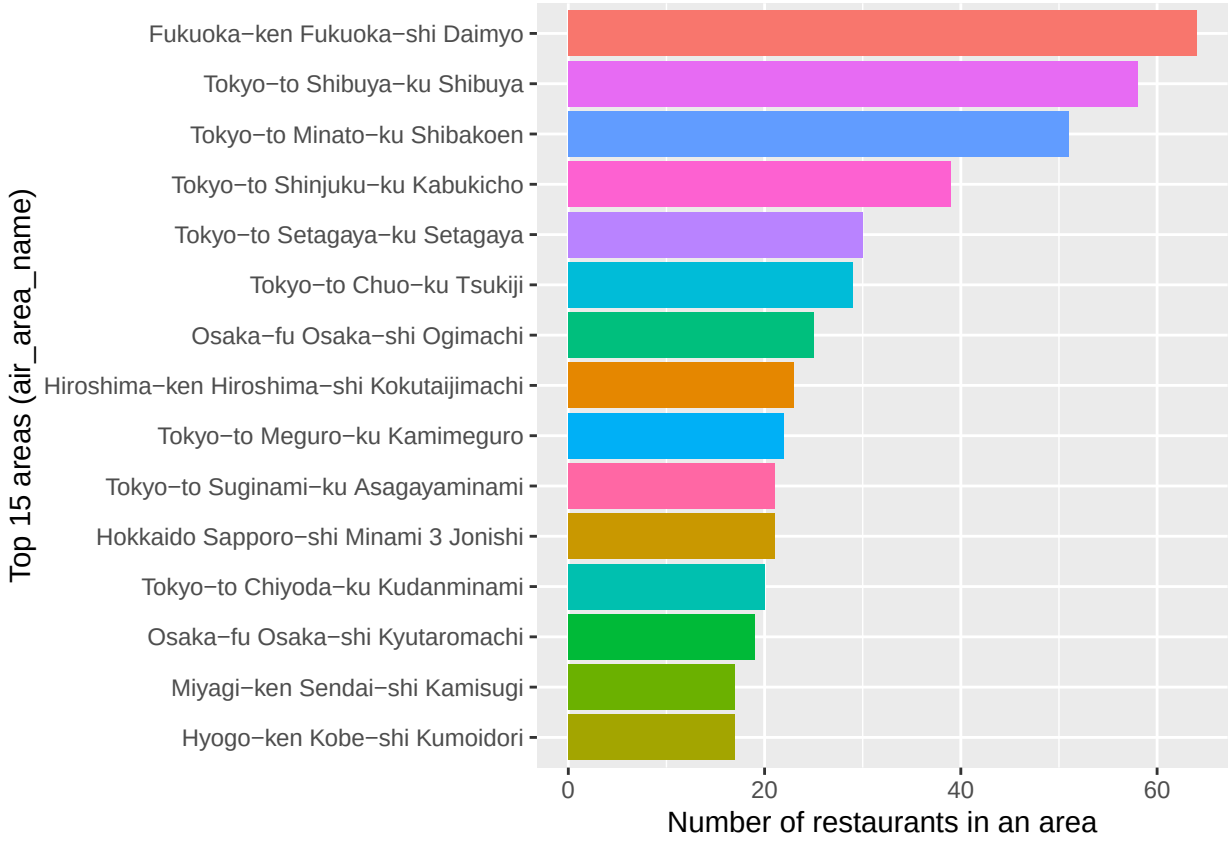
- Let's have a look on the genre name frequency

```
reservations %>%
  group_by(air_genre_name) %>%
  count() %>%
  ggplot(aes(air_genre_name, n, fill = air_genre_name)) +
  geom_col() +
  theme(legend.position = "none",axis.text.x  = element_text(angle=45, hjust=1, vjust=0.9))+
  labs(x = "Air genre name", y = "Number of genre restaurants")
```



- Let's have a look on the frequency of restaurant in the different area

```
restaurant_info %>%
  group_by(air_area_name) %>%
  count() %>%
  ungroup() %>%
  top_n(15,n) %>%
  ggplot(aes(reorder(air_area_name, n, FUN = min) ,n, fill = air_area_name)) +
  geom_col() +
  theme(legend.position = "none") +
  coord_flip() +
  labs(x = "Top 15 areas (air_area_name)", y = "Number of restaurants in an area")
```



Examine the variables in the datasets

```
for(data in list(reservations,visits,date_info,restaurant_info)) {

  print(skimr::skim(data))
  print("--------------------------------------")
}
```

```
## -- Data Summary ------------------------
##                            Values
## Name                       data
## Number of rows             36259
## Number of columns          10
## _________________________
## Column type frequency:
##   character                6
##   numeric                  4
## _________________________
## Group variables            None
##
## -- Variable type: character -------------------------------------------------------
##   skim_variable    n_missing complete_rate   min   max empty n_unique whitespace
## 1 ID                       0             1    13    15     0      275          0
## 2 visit_datetime           0             1    19    19     0     1617          0
## 3 reserve_datetime         0             1    19    19     0     1892          0
## 4 ID..5                    0             1    13    15     0      275          0
```

```
## 5 air_genre_name           0             1     5    28     0       14          0
## 6 air_area_name            0             1    21    45     0       64          0
##
## -- Variable type: numeric -----------------------------------------------------
##   skim_variable    n_missing complete_rate     mean       sd    p0    p25
## 1 X                        0             1 18130    10467.     1   9066.
## 2 reserve_visitors         0             1     4.31     4.56   1      2
## 3 latitude                 0             1    36.4      3.10  33.2   34.7
## 4 longitude                0             1   137.       3.78 130.   135.
##       p50     p75    p100 hist
## 1 18130   27194.  36259
## 2     3       5     100
## 3    35.6    35.7    44.0
## 4   140.    140.    144.
## [1] "--------------------------------------"
## -- Data Summary ------------------------
##                            Values
## Name                       data
## Number of rows             61803
## Number of columns          9
## ________________________
## Column type frequency:
##   character                5
##   numeric                  4
## ________________________
## Group variables            None
##
## -- Variable type: character ---------------------------------------------------
##   skim_variable  n_missing complete_rate   min   max empty n_unique whitespace
## 1 ID                     0             1    13    15     0      825          0
## 2 visit_date             0             1    10    10     0       90          0
## 3 ID..4                  0             1    13    15     0      825          0
## 4 air_genre_name         0             1     5    28     0       14          0
## 5 air_area_name          0             1    21    45     0      103          0
##
## -- Variable type: numeric -----------------------------------------------------
##   skim_variable n_missing complete_rate    mean       sd    p0     p25     p50
## 1 X                     0             1 30902   17841.     1   15452.  30902
## 2 visitors              0             1    20.8    16.7    1       9      17
## 3 latitude              0             1    35.6     2.06  33.2    34.7    35.7
## 4 longitude             0             1   137.      3.66 130.    135.    140.
##       p75    p100 hist
## 1 46352.  61803
## 2    29     877
## 3    35.7    44.0
## 4   140.    144.
## [1] "--------------------------------------"
## -- Data Summary ------------------------
##                            Values
## Name                       data
## Number of rows             517
## Number of columns          4
## ________________________
## Column type frequency:
```

```
##   character                2
##   numeric                  2
## ________________________
## Group variables            None
##
## -- Variable type: character -----------------------------------------------------
##   skim_variable n_missing complete_rate   min   max empty n_unique whitespace
## 1 calendar_date         0             1    10    10     0      517          0
## 2 day_of_week           0             1     6     9     0        7          0
##
## -- Variable type: numeric -------------------------------------------------------
##   skim_variable n_missing complete_rate     mean      sd    p0   p25   p50   p75
## 1 X                     0             1 259      149.        1   130   259   388
## 2 holiday_flg           0             1   0.0677   0.251     0     0     0     0
##    p100 hist
## 1   517
## 2     1
## [1] "------------------------------------"
## -- Data Summary ------------------------
##                           Values
## Name                      data
## Number of rows            829
## Number of columns         6
## ________________________
## Column type frequency:
##   character               3
##   numeric                 3
## ________________________
## Group variables           None
##
## -- Variable type: character -----------------------------------------------------
##   skim_variable  n_missing complete_rate   min   max empty n_unique whitespace
## 1 ID                     0             1    13    15     0      829          0
## 2 air_genre_name         0             1     5    28     0       14          0
## 3 air_area_name          0             1    21    45     0      103          0
##
## -- Variable type: numeric -------------------------------------------------------
##   skim_variable n_missing complete_rate  mean     sd    p0   p25   p50   p75
## 1 X                     0             1 415   239.     1   208   415   622
## 2 latitude              0             1  35.6   2.08  33.2  34.7  35.7  35.7
## 3 longitude             0             1 137.    3.65 130.  135.  140.  140.
##    p100 hist
## 1 829
## 2  44.0
## 3 144.
## [1] "------------------------------------"
```

Convert the date variables with lubridate

```
reservations$visit_datetime <- ymd_hms(reservations$visit_datetime)
reservations$reserve_datetime <- ymd_hms(reservations$reserve_datetime)

visits$visit_date <- ymd(visits$visit_date)
```

```
date_info$calendar_date <- ymd(date_info$calendar_date)
```

Examine the dates in each dataset

```
tribble(
  ~"variable",                      ~"Reservation",          ~"Visits", ~"Date-Info",
  "visit min", min(reservations$visit_datetime), min(visits$visit_date), NA,
  "visit max", max(reservations$visit_datetime), max(visits$visit_date), NA,
  "reserve min", min(reservations$reserve_datetime), NA, NA,
  "reserve max", max(reservations$reserve_datetime), NA, NA,
  "Other min", NA                                  , NA, min(date_info$calendar_date),
  "Other max", NA                                  , NA, max(date_info$calendar_date)

)
```

```
## # A tibble: 6 x 4
##   variable    Reservation         Visits     `Date-Info`
##   <chr>       <dttm>              <date>     <date>
## 1 visit min   2017-01-01 10:00:00 2017-01-01 NA
## 2 visit max   2017-05-28 23:00:00 2017-03-31 NA
## 3 reserve min 2017-01-01 00:00:00 NA         NA
## 4 reserve max 2017-03-31 23:00:00 NA         NA
## 5 Other min   NA                  NA         2016-01-01
## 6 Other max   NA                  NA         2017-05-31
```

From this output we can see the date ranges of the datasets.

Now we will summarize by day.

```
reserve_data_day_summed <- reservations %>% mutate(reserveDay = day(reserve_datetime))

reservations<- reservations%>%mutate(day_reservation = day(reservations$reserve_datetime))
reservations<- reservations%>%mutate(month_reservation = month(reservations$reserve_datetime))
reservations<- reservations%>%mutate(year_reservation = 2017)
reservations<- reservations%>%mutate(date = '')

reservations$date <- as.Date(with(reservations, paste(year_reservation, month_reservation, day_reservati

reservation_per_day <- reservations %>%
  mutate(day_mon_rest=paste(day_reservation,month_reservation,ID))%>%
  group_by(ID,day_mon_rest,date) %>%
  summarise(Tot_visit_day_from_Reservations = sum(reserve_visitors),
            count_of_reservations = n())

head(reservation_per_day)
```

```
## # A tibble: 6 x 5
## # Groups:   ID, day_mon_rest [6]
##   ID            day_mon_rest       date       Tot_visit_day_fr~ count_of_reserv~
##   <chr>         <chr>              <date>                 <int>            <int>
## 1 restaurant_ 1 1 2 restaurant_ 1  2017-02-01                14                5
## 2 restaurant_ 1 1 3 restaurant_ 1  2017-03-01                63                7
## 3 restaurant_ 1 10 2 restaurant_ 1 2017-02-10                 2                1
```

```
## 4 restaurant_ 1 10 3 restaurant_ 1 2017-03-10                30                1
## 5 restaurant_ 1 11 1 restaurant_ 1 2017-01-11                 4                1
## 6 restaurant_ 1 11 3 restaurant_ 1 2017-03-11                20                7
```

Forming modeling dataset

Now we've created a summarized version of the reservation dataset and we need to join this into the visits dataset to form our total dataset with all of the variables we will use

```
# visits <- visits %>% add_row(ID="restaurant_ 292",air_genre_name="Cafe/Sweets",air_area_name="T?ky?-t

date_info$calendar_date=ymd(date_info$calendar_date)

combinedDataset <- visits %>% mutate(date = visit_date) %>%  left_join(reservation_per_day,by = c('date
  mutate(calendar_date=date) %>% left_join(date_info,by='calendar_date')

combinedDataset <- combinedDataset %>% left_join(
  y=(combinedDataset %>% group_by(air_genre_name) %>% summarise(avgDailyVisitsGenre = mean(visitors))),
by="air_genre_name")
```

Create Training and test

```
trainset <- subset(combinedDataset,visit_date <= as.Date("2017-03-10"))
testset <- subset(combinedDataset,visit_date > as.Date("2017-03-10"))
```

Now we wwill begin to fit with linear regression models to start

This is a simpler modeling technique and we want to start here to see if by chance it is the best fitting type of model.

Model 1

```
# mod1<-lm(visitors ~ date + air_genre_name+latitude+longitude+count_of_reservations, data=trainset)
#
# summary(mod1)
```

Model 2

```
# mod2<- lm( visitors ~ date + Tot_visit_day_from_Reservations + day_of_week + holiday_flg + air_genre_
#
# summary(mod2)
```

Model 3

```
# mod3<- lm( visitors ~ visit_date + Tot_visit_day_from_Reservations + day_of_week + holiday_flg + air_
# summary(mod3)
# max rsq achieved to this point
print(.252)
```

```
## [1] 0.252
```

Post Model 3 Conclusions To this point we have tried a few different iterations to fairly poor results. After this we will include ID as we think that it might be important.

Ultimately in this sort of modeling situation, the restaurant themselves will have an effect on the final visitor total. Think of this similar to a sports team or predicting attendance for La Liga, the team being predicted will no doubt be one of the largest predictors.

Model 4

```
# mod4<- lm( visitors ~  ID + visit_date + Tot_visit_day_from_Reservations + day_of_week + holiday_flg
# summary(mod4)
```

Model 5

Another few iterations - we realized that visit date wouldn't be of any use for the dataset we are predicting (and would make the model not usable), thus we've swapped it for **day__of__week**.

```
# mod5<- lm( visitors ~  ID + day_of_week + holiday_flg + air_genre_name + air_area_name, data=trainset
# summary(mod5)
```

validation of the model

Now let's explore decision tree fits

We will now use a very powerful supervised learning algorithm **random forest** to try and fit to our model and see how it performs. It is an extrapolation of a decision tree algorithm.

```
library(tidymodels)

combinedDataset=combinedDataset %>% left_join(combinedDataset%>%group_by(ID,day_of_week)%>%summarise(av



set.seed(42)

combinedDataset= combinedDataset%>%select(-ID..4,-X.y)%>%rename(X=X.x)

data_split <- initial_time_split(combinedDataset, prop = 0.75)

tm_train <- training(data_split)
tm_test <- testing(data_split)

tm_rec <- tm_train %>%
  recipe(visitors ~ . ) %>%
  step_normalize(all_predictors())

# Show the result of our recipe
tm_rec
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor         16
##
## Operations:
```

```
##
## Centering and scaling for all_predictors()
```

```
rf_spec <- rand_forest(mode = "regression") %>%
  set_engine("ranger")

rf_spec
```

```
## Random Forest Model Specification (regression)
##
## Computational engine: ranger
```

```
## Random Forest Model Specification (regression)
##
## Computational engine: ranger
rf_fit <- rf_spec %>%
  fit(visitors ~ .,
    data = tm_train
  )

print("RF R-Squared")
```

```
## [1] "RF R-Squared"
```

```
rf_fit$fit$r.squared
```

```
## [1] 0.6433334
```

Creation of the regression tree with training set

We invoke 3 new packages to be able to run the decision tree method

```
library(rpart)
library(rpart.plot)
library(caTools)

set.seed(122)

combinedDataset$ID=as.factor(combinedDataset$ID)
combinedDataset$air_genre_name=as.factor(combinedDataset$air_genre_name)
combinedDataset$air_area_name=as.factor(combinedDataset$air_area_name)
combinedDataset=combinedDataset%>%select(-day_mon_rest)

combinedDataset$count_of_reservations[is.na(combinedDataset$count_of_reservations)] <- 0
combinedDataset$Tot_visit_day_from_Reservations[is.na(combinedDataset$Tot_visit_day_from_Reservations)]

trainset2 <- subset(combinedDataset,visit_date <= as.Date("2017-03-10"))
testset2 <- subset(combinedDataset,visit_date > as.Date("2017-03-10"))



firsttree <- rpart(
  formula = visitors ~ ID + air_area_name + latitude + longitude + air_genre_name+ avgDailyVisitsGenre +
  data    = trainset2,
  method  = "anova"
  )
```

```
firsttree
```

```
## n= 46763
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 46763 11971780.0 20.044030
##    2) avg_of_day_id< 25.60769 32958  2674783.0 13.341860
##      4) avg_of_day_id< 14.83974 18881   664618.8  8.901753
##        8) avg_of_day_id< 8.630682 8283   127685.3  5.881806 *
##        9) avg_of_day_id>=8.630682 10598   402351.3 11.262030 *
##      5) avg_of_day_id>=14.83974 14077  1138674.0 19.297220 *
##    3) avg_of_day_id>=25.60769 13805  4282150.0 36.044770
##      6) avg_of_day_id< 41.96154 10340  1558884.0 31.300580
##       12) avg_of_day_id< 32.64103 5742   726847.9 27.976840 *
##       13) avg_of_day_id>=32.64103 4598   689386.6 35.451280 *
##      7) avg_of_day_id>=41.96154 3465  1796059.0 50.202020
##       14) avg_of_day_id< 56.96154 2725   635130.6 46.458720 *
##       15) avg_of_day_id>=56.96154 740   982135.9 63.986490 *
```

#Validation with the test set

Now we will consider again LM with all of the variables that we've now developed

Considering visit date time instead of reserve date time to not lose information relate to the future

```
mod6=lm( visitors ~  ID + day_of_week + holiday_flg + air_genre_name + air_area_name + count_of_reserva
# summary(mod6)
#We will print the rsq instead of printing the summary to conciseness of the report

predmod6train <- predict(firsttree, data=trainset2)
#rsq
rsqUPC(trainset2,predmod6train)
```

```
## [1] 0.6072254
```

test the new fits with rsq on the test set as the above rsq was on the train set

```
#mod 6
predictTest2<-predict(mod6, newdata = testset2)
rsqUPC(testset2,predictTest2)
```

```
## [1] 0.6252271
```

Here we see quite a close result to the original meaning that we can expect a similar performance in the wild if you will. As a part of the next steps we will retrain the model inclusive of the test set data as we would otherwise lose some levels otherwise based on our setup.

Refitting LM model with full dataset

Now that we've tested with a train test split we want to reload the best LM model with the full dataset so that we don't drop levels when we predict the submission dataset.

We also undergo a series of steps here to get the variables that we need into the submission dataset to achieve an accurate prediction. We also ensure that we maintain the original sequence which seems entirely random.

```
submission2 <- read.csv("project_submission (1).csv")

submission2<-submission2%>%mutate(date1 = str_sub(ID,-10,-1))
submission2<-submission2%>%mutate(id = str_sub(ID,1,-13))
submission2$date1<-ymd(submission2$date1)
submission2$id <- str_trim(submission2$id)

submission2<-rename(submission2,id=ID,date=date1,ID=id)

submission2$counter <- seq(from=1,to=15770,by=1)

submission2$ID=as.factor(submission2$ID)

submission2 <- submission2 %>% left_join(restaurant_info%>%select(-X), by="ID") %>% left_join(date_info

reservations<- reservations%>%mutate(day_reservation = day(reservations$visit_datetime))
reservations<- reservations%>%mutate(month_reservation = month(reservations$visit_datetime))
reservations<- reservations%>%mutate(year_reservation = 2017)
reservations<- reservations%>%mutate(date = '')

reservations$date <- as.Date(with(reservations, paste(year_reservation, month_reservation, day_reservati

reservation_per_day <- reservations %>%
  group_by(ID,date) %>%
  summarise(Tot_visit_day_from_Reservations = sum(reserve_visitors),
            count_of_reservations = n())

reservation_per_day$ID=as.factor(reservation_per_day$ID)

mod6<- lm( visitors ~  ID + day_of_week + holiday_flg + air_genre_name + air_area_name + count_of_reserv
predictTest2<-predict(mod6, newdata = combinedDataset)
##quick rsq calculation on model
rsqUPC(combinedDataset,predictTest2)
```

```
## [1] 0.6479203
```

```
submission2 <- submission2 %>% select(-date)%>%mutate(date=ymd(calendar_date)) %>% select(-calendar_date

submission2$count_of_reservations[is.na(submission2$count_of_reservations)] <- 0
submission2$Tot_visit_day_from_Reservations[is.na(submission2$Tot_visit_day_from_Reservations)] <- 0
submission2$avg_of_day_id[is.na(submission2$avg_of_day_id)] <- 0

submission_zeroes2 <- submission2 %>% filter(ID=="restaurant_ 292"|ID=="restaurant_ 325")

submission2 <- submission2 %>% filter(ID!="restaurant_ 292"&ID!="restaurant_ 325")


submission2$visitors <- predict(mod6, newdata = submission2)
submission2$visitors <- round(submission2$visitors,0)
submission_zeroes2 <- submission_zeroes2 %>% select(id,ID,visitors,counter)
submission2 <- submission2 %>% select(id,ID,visitors,counter)

submission2 <- bind_rows(submission2,submission_zeroes2) %>%arrange(counter)%>%select(-counter)
```

```r
submission2$visitors <- if_else(submission2$visitors<0,0,submission2$visitors)

submission2=submission2%>%select(-ID)
submission2 <-rename(submission2,ID=id)

submission2 %>% write_csv("project6_data_submission.csv")
```