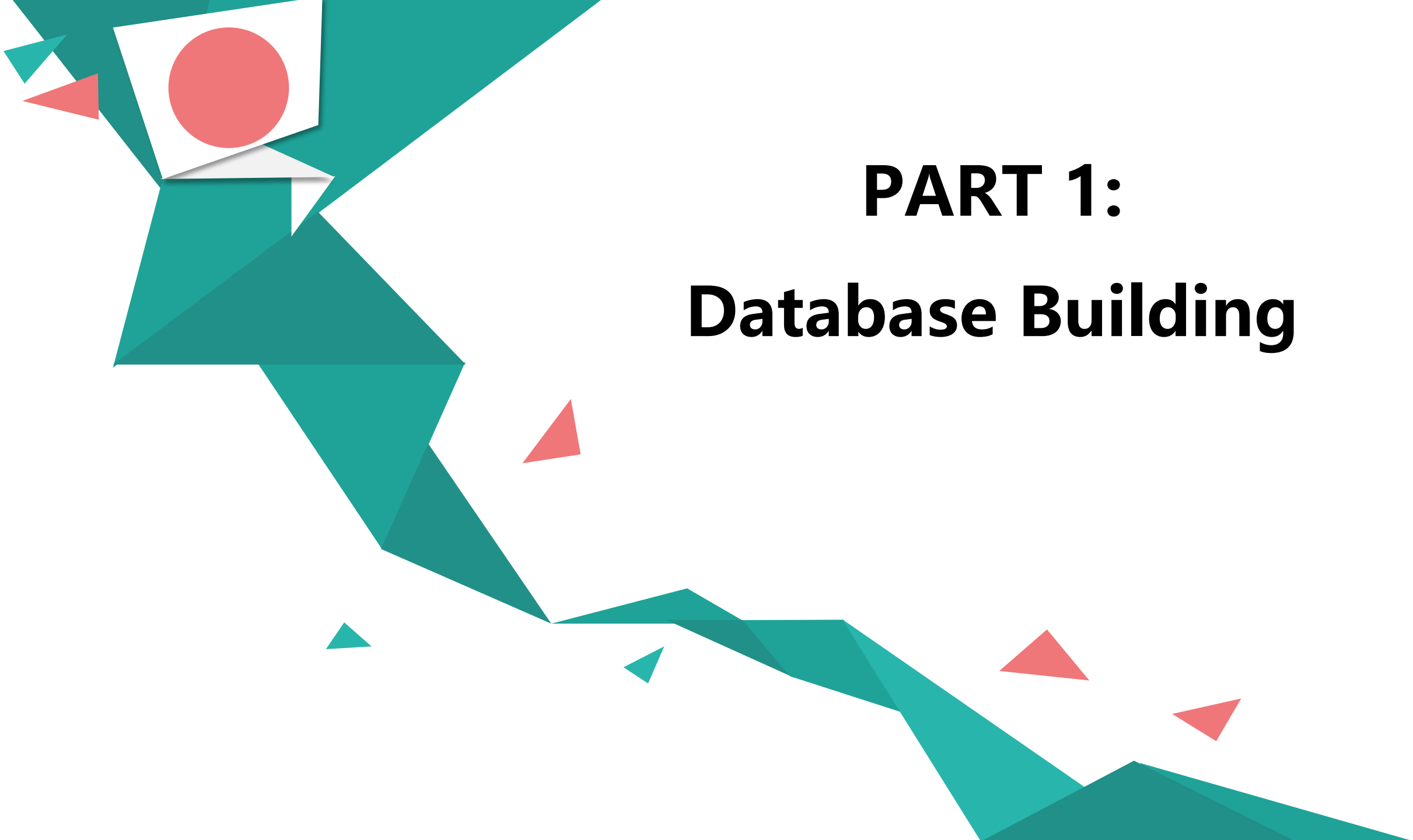
MH6142 Database Systems Hands-On Project

G2103131A WU MINGJING
G2103195K ZHANG BOJUN
G2103160J ZHOU DUO



PART 1:

Database Building

Research Background

The Real-World Problem:

It is crucial in the F&B industry for restaurants to know the number of customers visiting in advance to effectively purchase ingredients and schedule staff members. **Generally, such forecast is not easy as there are many unpredictable factors affecting the actual number of customers. It's even more challenging for newer restaurants with little historical data.**

In this project, we will look into such real - world problem in Japan by using the dataset provided by Recruit Holdings to make some predictions.

Kaggle Competition Link:
https://www.kaggle.com/c/recruit-restaurant-visitor-forecasting

Problem Statement

Reservation and visitation data are provided from two systems:

- **Hot Pepper Gourmet (hpg):** similar to Yelp, users can search restaurants and make a reservation online.
- **AirREGI / Restaurant Board (air):** similar to Square, a reservation control and cash register system.

Both have different list of restaurants and customers can make reservations through two systems.

Problem statement:

Using past reservation and visitation data to predict the total number of visitors of **air restaurants** for future dates and meanwhile draw interesting insights from the analysis.

Problem Statement

For this project, we will only use the AirREGI / Restaurant Board (air) data. Why?

As compared to the air data, insufficient data provided for the Hot Pepper Gourmet (hpg), such as the actual number of customers visiting restaurants. We cannot extract useful information from it and do accurate prediction. The AirREGI / Restaurant Board (air) data has **both reservation data and visitation data**, we can explore more meaningful pattern and make predictions for the future.

Data Set Information

air_store_info

air_store_id	air_genre_name	air_area_name	latitude	longitude
air_df9355c47c5df9d3	Italian/French	Fukuoka-ken Fukuoka-shi Daimyo	33.5892157	130.3928134
air_3440e0ea1b70a99b	Italian/French	Osaka-fu Osaka-shi Ogimachi	34.7053617	135.5100252
air_d186b2cb0b9ce022	Italian/French	Tokyo-to Shibuya-ku Shibuya	35.6617773	139.7040506

air_store_id

air_genre_name: food style

air_area_name: location of the restaurant in Japan

latitude: geolocation information

longitude: geolocation information

air_visit_data

air_store_id	visit_date	visitors
air_ba937bf13d40fb24	13/1/2016	25
air_ba937bf13d40fb24	14/1/2016	32
air_ba937bf13d40fb24	15/1/2016	29

air_store_id

visit_date

visitors: the number of visitors to the restaurant on the date

air_reserve

air_store_id	visit_datetime	reserve_datetime	reserve_visitors
air_877f79706adbfb06	1/1/2016 19:00	1/1/2016 16:00	1
air_789466e488705c93	4/1/2016 17:00	4/1/2016 17:00	2
air_789466e488705c93	4/1/2016 17:00	4/1/2016 17:00	2

air_store_id: restaurant id in air system

visit_datetime: time of reservation

reserve_datetime: time reservation was made

reserve_visitors: number of visitors for that reservation

date_info

calendar_date	day_of_week	holiday_flg
1/1/2016	Friday	1
2/1/2016	Saturday	1
3/1/2016	Sunday	1

calendar_date

day_of_week

holiday_flg: whether the day is a holiday in Japan

Air_reserve Adjustment

air_reserve

air_store_id	visit_datetime	reserve_datetime	reserve_visitors
air_877f79706adbfb06	1/1/2016 19:00	1/1/2016 16:00	1
air_789466e488705c93	4/1/2016 17:00	4/1/2016 17:00	2
air_789466e488705c93	4/1/2016 17:00	4/1/2016 17:00	2

air_reserve

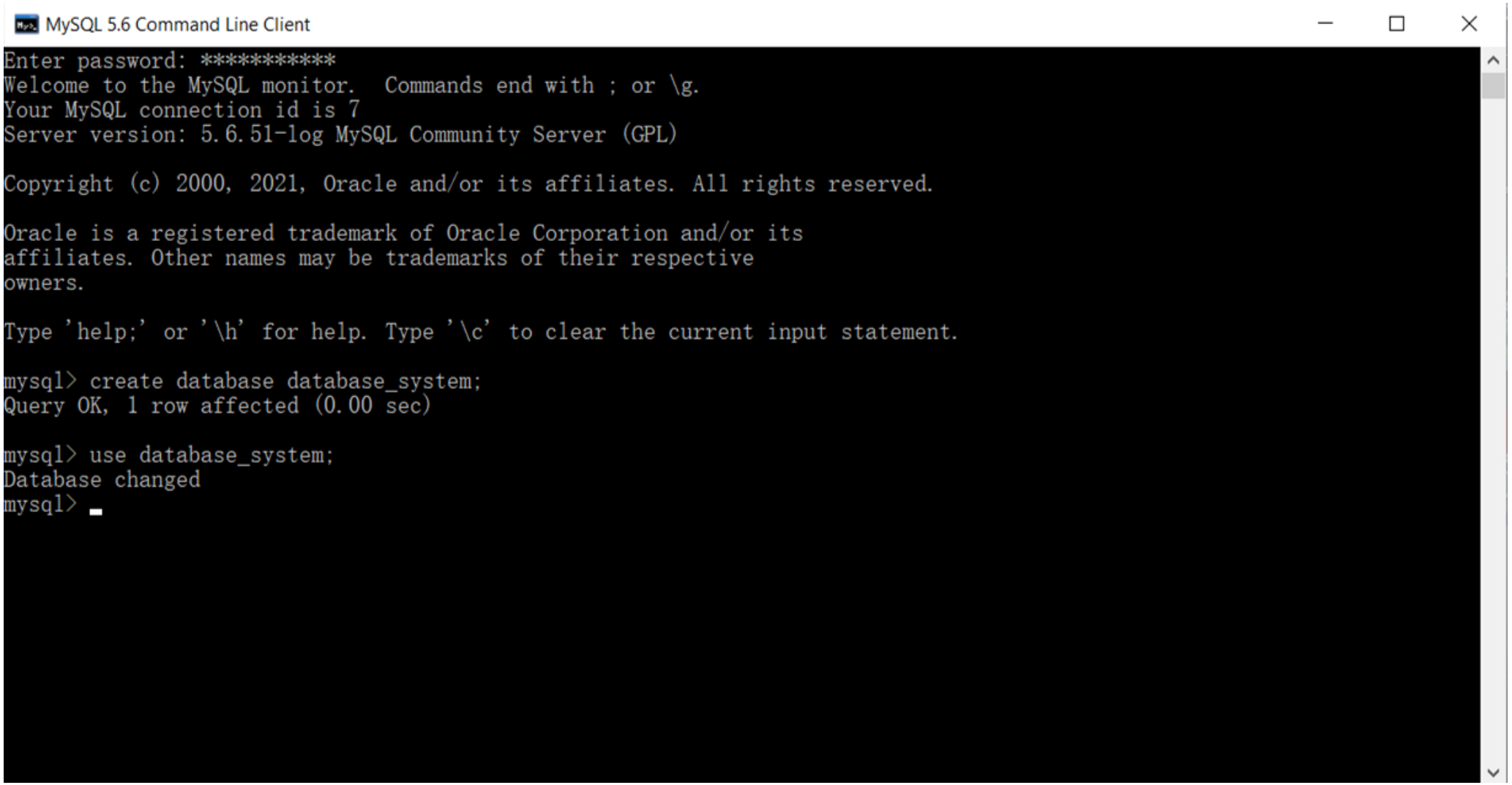
air_store_id	visit_date	visit_time	reserve_date	reserve_time	reserve_visitors
air_877f79706adbfb06	1/1/2016	19:00:00	1/1/2016	16:00:00	1
air_789466e488705c93	4/1/2016	17:00:00	4/1/2016	17:00:00	4

Separate date and time, aggregate number of visitors under the same day and same time

```
air_reserve <- air_reserve %>%
  mutate(reserve_date = date(reserve_datetime),
         reserve_time = format(reserve_datetime, format = "%H:%M:%S"),
         visit_date = date(visit_datetime),
         visit_time = format(visit_datetime, format = "%H:%M:%S"))
air_reserve <- air_reserve[,c(1,7,8,5,6,4)]
air_reserve_aggregate <- air_reserve %>%
  group_by(air_store_id,visit_date,visit_time,reserve_date,reserve_time) %>%
  summarise_all(sum)
write.csv(air_reserve_aggregate,"D:\\NTU\\Database System\\Group Project\\adjusted csv\\air_reserve.csv", row.names = FALSE)
```

Local Database Creation

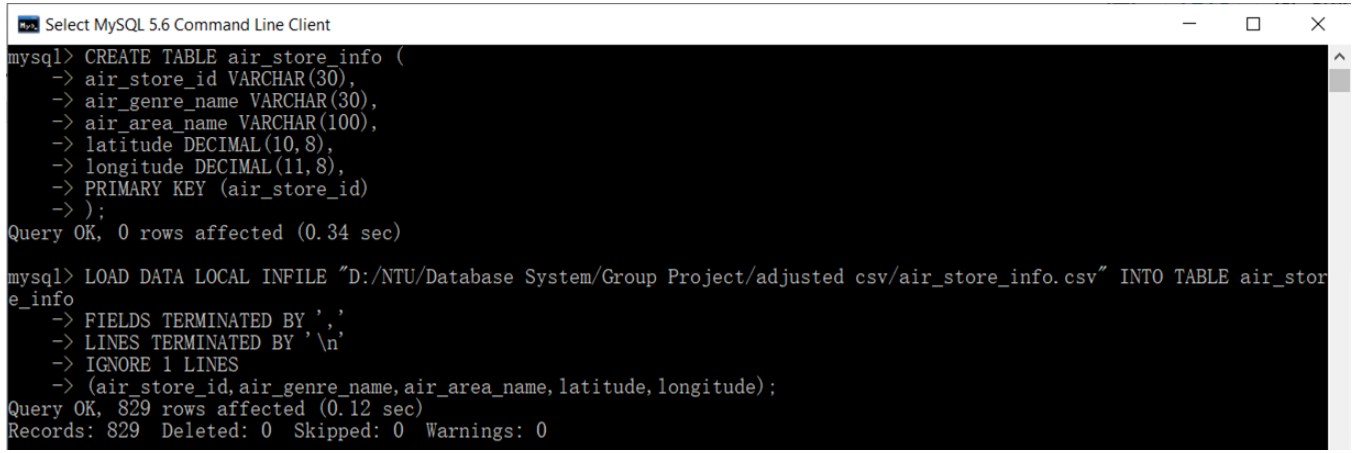
First, we create local database through MySQL5.6:



Local Database Creation

Create table air_store_info

```
CREATE TABLE air_store_info (
air_store_id VARCHAR(30),
air_genre_name VARCHAR(30),
air_area_name VARCHAR(100),
latitude DECIMAL(10,8),
longitude DECIMAL(11,8),
PRIMARY KEY (air_store_id)
);
```




```
LOAD DATA LOCAL INFILE "D:/NTU/Database System/Group Project/adjusted csv/air_store_info.csv" INTO TABLE air_store_info
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(air_store_id,air_genre_name,air_area_name,latitude,longitude);
```

Local Database Creation

Create table date_info

```
CREATE TABLE date_info(
calendar_date DATE,
day_of_week VARCHAR(20),
holiday_flg INT,
PRIMARY KEY (calendar_date)
);
```


```
LOAD DATA LOCAL INFILE "D:/NTU/Database System/Group Project/adjusted csv/date_info.csv" INTO TABLE date_info
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(@datevar,day_of_week,holiday_flg)
set calendar_date=STR_TO_DATE(@datevar,'%e/%c/%Y');
```

Local Database Creation

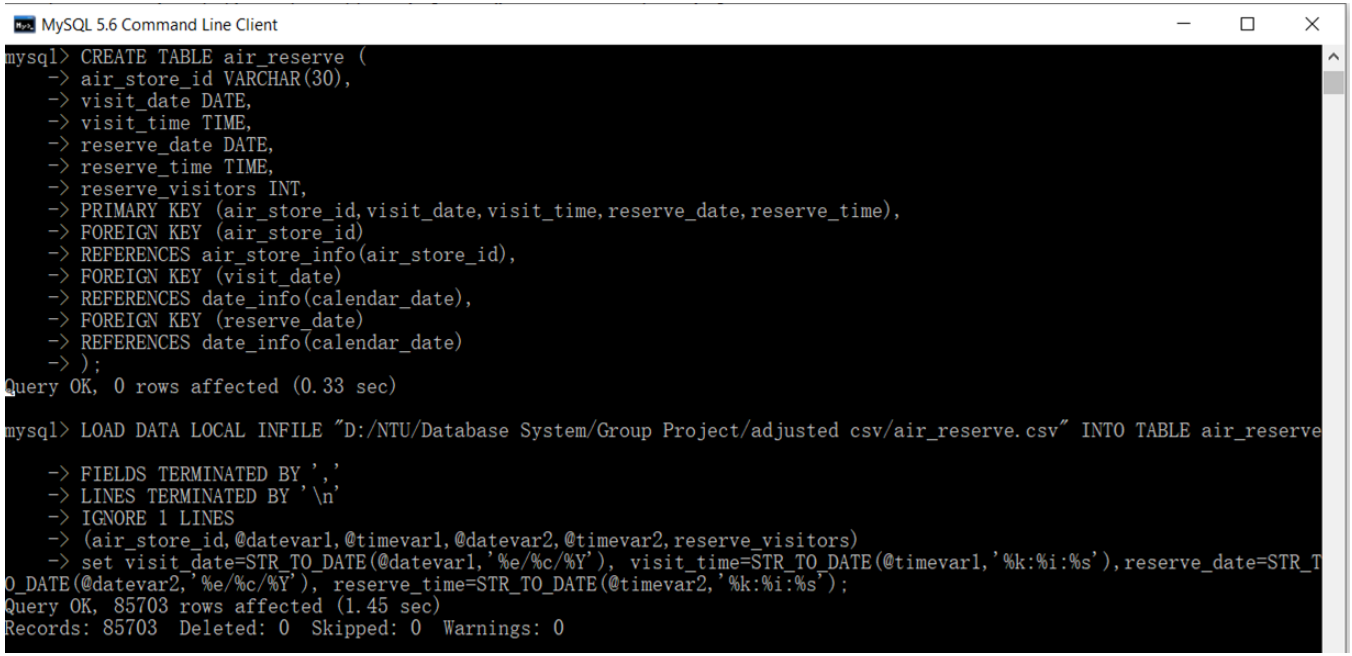
Create table air_reserve

```
CREATE TABLE air_reserve (
air_store_id VARCHAR(30),
visit_date DATE,
visit_time TIME,
reserve_date DATE,
reserve_time TIME,
reserve_visitors INT,
PRIMARY KEY (air_store_id,visit_date,visit_time,reserve_date,reserve_time),
FOREIGN KEY (air_store_id)
REFERENCES air_store_info(air_store_id),
FOREIGN KEY (visit_date)
REFERENCES date_info(calendar_date),
FOREIGN KEY (reserve_date)
REFERENCES date_info(calendar_date)
);
```

Local Database Creation

Create table air_reserve

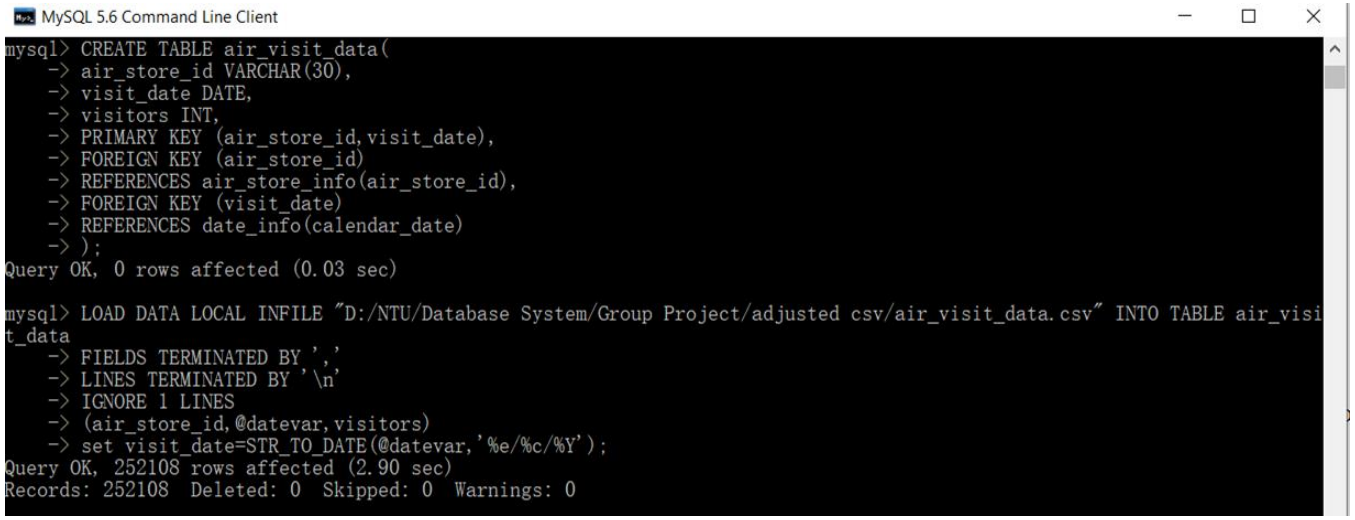
```
LOAD DATA LOCAL INFILE "D:/NTU/Database System/Group Project/adjusted csv/air_reserve.csv" INTO TABLE air_reserve
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(air_store_id,@datevar1,@timevar1,@datevar2,@timevar2,reserve_visitors)
set visit_date=STR_TO_DATE(@datevar1,'%e/%c/%Y'),
visit_time=STR_TO_DATE(@timevar1,'%k:%i:%s'),reserve_date=STR_TO_DATE(@datevar2,'%e/%c/%Y'),
reserve_time=STR_TO_DATE(@timevar2,'%k:%i:%s');
```



Local Database Creation

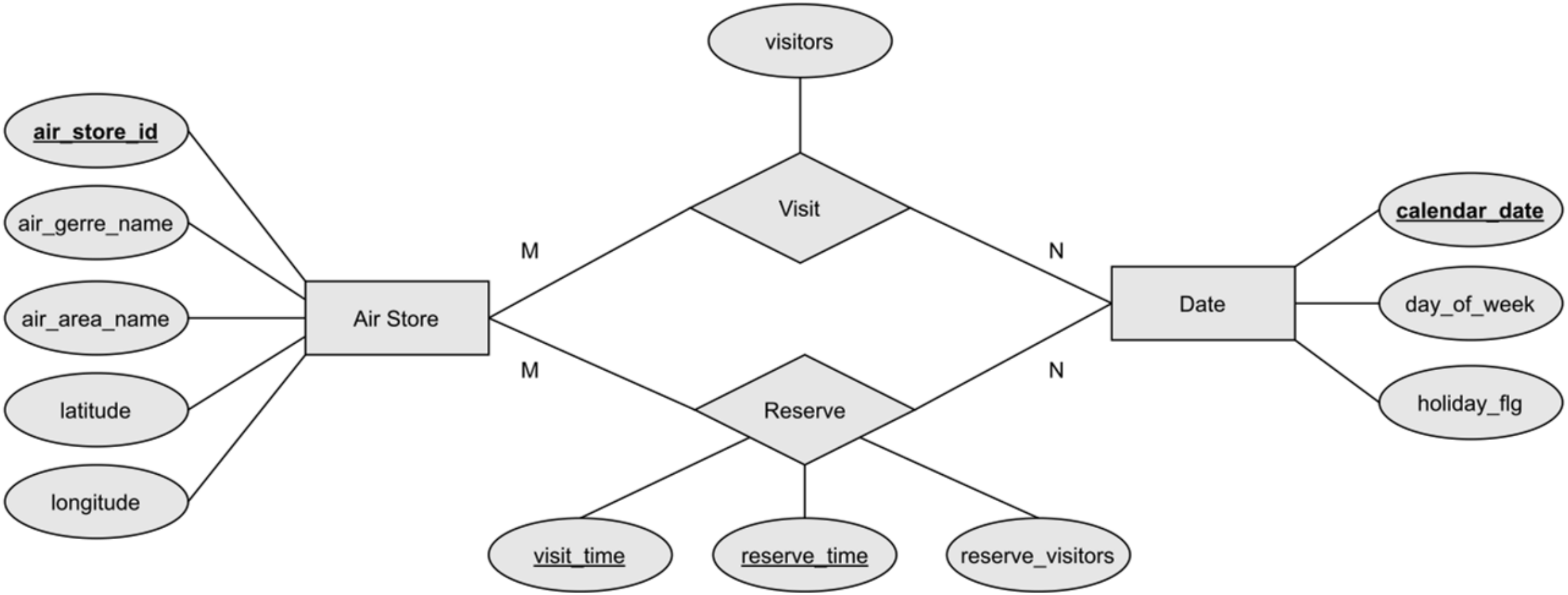
Create table air_reserve

```
CREATE TABLE air_visit_data(
air_store_id VARCHAR(30),
visit_date DATE,
visitors INT,
PRIMARY KEY (air_store_id,visit_date),
FOREIGN KEY (air_store_id)
REFERENCES air_store_info(air_store_id),
FOREIGN KEY (visit_date)
REFERENCES date_info(calendar_date)
);
```




```
LOAD DATA LOCAL INFILE "D:/NTU/Database System/Group Project/adjusted csv/air_visit_data.csv" INTO TABLE air_visit_data
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(air_store_id,@datevar,visitors)
set visit_date=STR_TO_DATE(@datevar,'%e/%c/%Y');
```

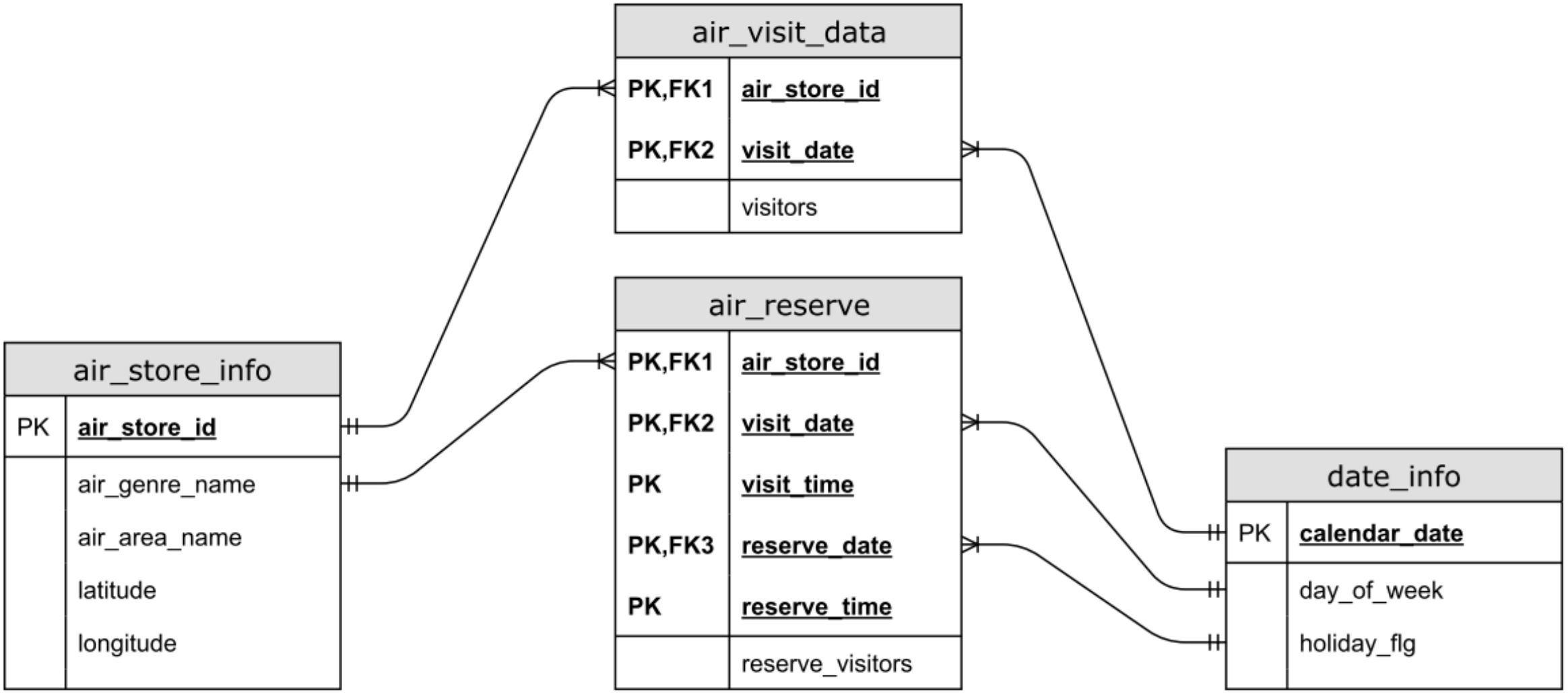
ER Diagram



Each store is visited / reserved on multiple days

Each day people visit / reserve multiple stores

Table Relationship

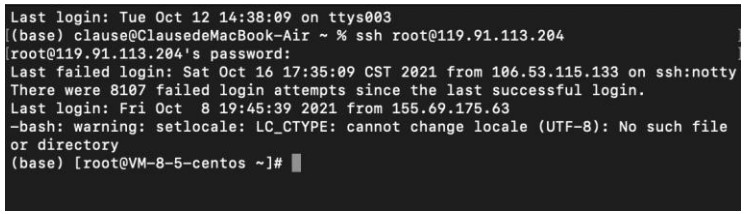


Database Server Building

Introduction

MySQL is an open-source database management system, commonly installed as part of the popular LEMP (Linux, Nginx, MySQL/MariaDB, PHP/Python/Perl) stack. It uses a relational database and SQL (Structured Query Language) to manage its data.

I bought a CentOS 7 server from Tencent Cloud with a root user with sudo privileges. You can learn more about how to set up a user with these privileges in the Initial Server Setup with CentOS 7 guide.

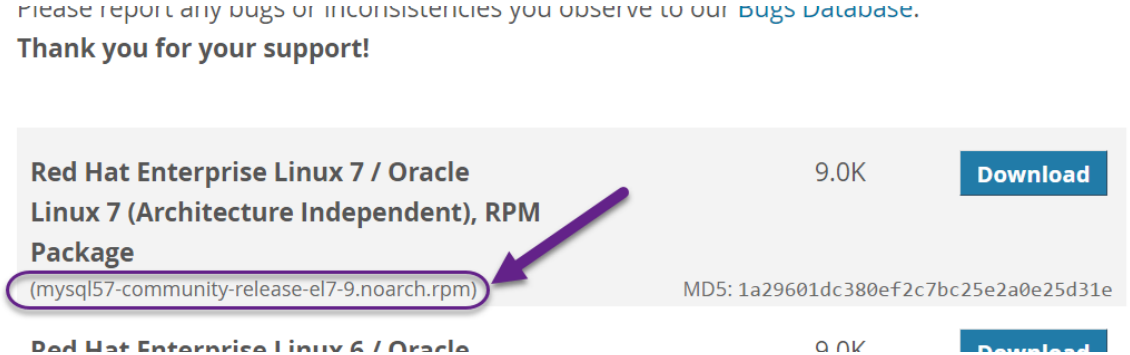


Database Server Building

Step 1 — Installing MySQL

As mentioned, the Yum command to install MySQL in fact installs MariaDB. To install MySQL, we'll need to visit the MySQL community Yum Repository which provides packages for MySQL.

wget https://dev.mysql.com/get/mysql57-community-release-el7-9.noarch.rpm



Now that we'll install the package:

sudo rpm -ivh mysql57-community-release-el7-9.noarch.rpm

This adds two new MySQL yum repositories, and we can now use them to install MySQL server:

sudo yum install mysql-server

Press y to confirm that you want to proceed. Since we've just added the package, we'll also be prompted to accept its GPG key. Press y to download it and complete the install.

Database Server Building

Step 2 — Starting MySQL

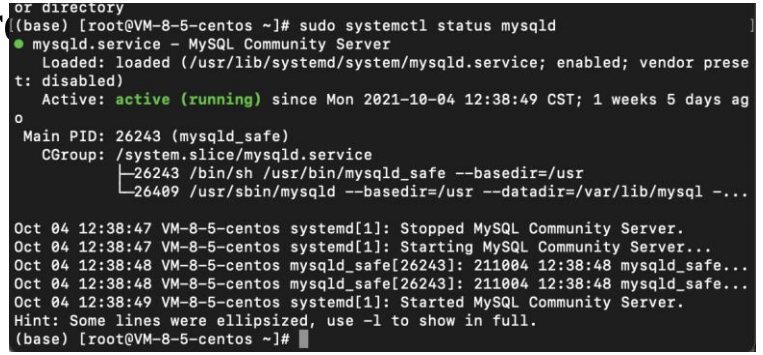
We'll start the daemon with the following command:

```
sudo systemctl start mysqld
```

systemctl doesn't display the outcome of all service management commands, so to be sure we succeeded, we'll use the following command:

```
sudo systemctl status mysqld
```

As the image, MySQL has successfully started, the output should contain Active: active (running). During the installation process, a temporary password is generated for the MySQL root user. Locate it in the mysqld.log with this command:




```
sudo grep 'our temporary password' /var/log/mysqld.log
```

The default password policy requires 12 characters, with at least one uppercase letter, one lowercase letter, one number and one special character.

Database Server Building

Step 3 — Testing MySQL

We can verify our installation and get information about it by connecting with the mysqladmin tool, a client that lets you run administrative commands. Use the following command to connect to MySQL as root (-u root), prompt for a password (-p), and return the version.

mysqladmin -u root -p version

```
error: 'Access denied for user 'root'@'localhost' (using password: YES)'
(base) [root@VM-8-5-centos ~]# mysqladmin -u root -p version
Enter password:
mysqladmin  Ver 8.42 Distrib 5.6.51, for Linux on x86_64
Copyright (c) 2000, 2021, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Server version          5.6.51
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /var/lib/mysql/mysql.sock
Uptime:                 12 days 8 hours 30 min 47 sec

Threads: 1  Questions: 5693  Slow queries: 54  Opens: 93  Flush tables: 1  Open
tables: 86  Queries per second avg: 0.005
(base) [root@VM-8-5-centos ~]#
```

This indicates your installation has been successful.

Database Server Building

Step 4. MySQL user settings

We need to add a MySQL user, just add a new user in the user table in the mysql database.

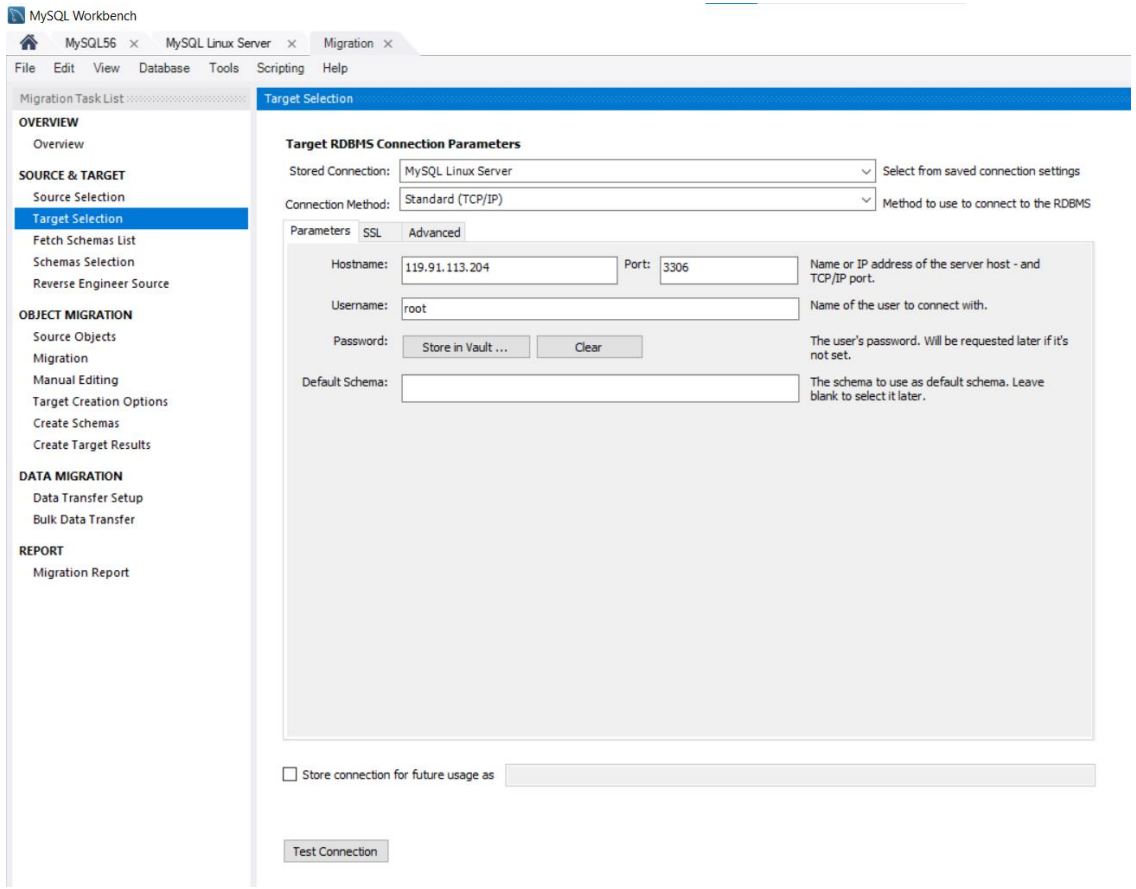
The following is an example of adding a user, the user name is myuser, the password is guest123, and the authorized user can perform SELECT, INSERT and UPDATE operation permissions:

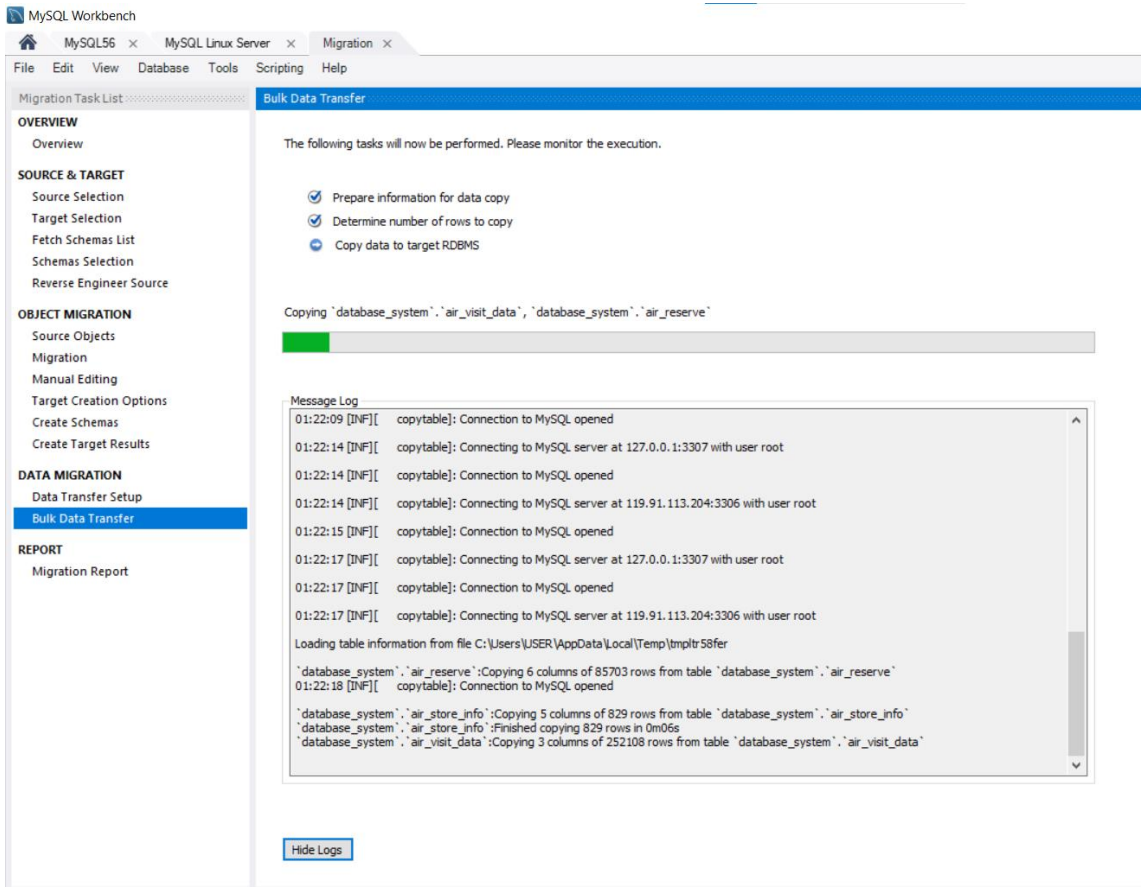
```
root@host# mysql -u root -p
Enter password:*******
mysql> use mysql;
Database changed

mysql> INSERT INTO user
          (host, user, password,
           select_priv, insert_priv, update_priv)
           VALUES ('%', 'myuser',
           PASSWORD('123456'), 'Y', 'Y', 'Y');
Query OK, 1 row affected (0.20 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT host, user, password FROM user WHERE user = 'myuser';
                                  +------+--------+-------------------------------------------+
                                  | host | user   | password                                  |
                                  +------+--------+-------------------------------------------+
                                  | %    | myuser | *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9 |
                                  +------+--------+-------------------------------------------+


  1 row in set (0.00 sec)
```

```
[mysql> use mysql;                                                    ]
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
[mysql> SELECT host, user, password FROM user WHERE user = 'myuser';  ]
+------+--------+-------------------------------------------+
| host | user   | password                                  |
+------+--------+-------------------------------------------+
| %    | myuser | *6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9 |
+------+--------+-------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

Data Migration

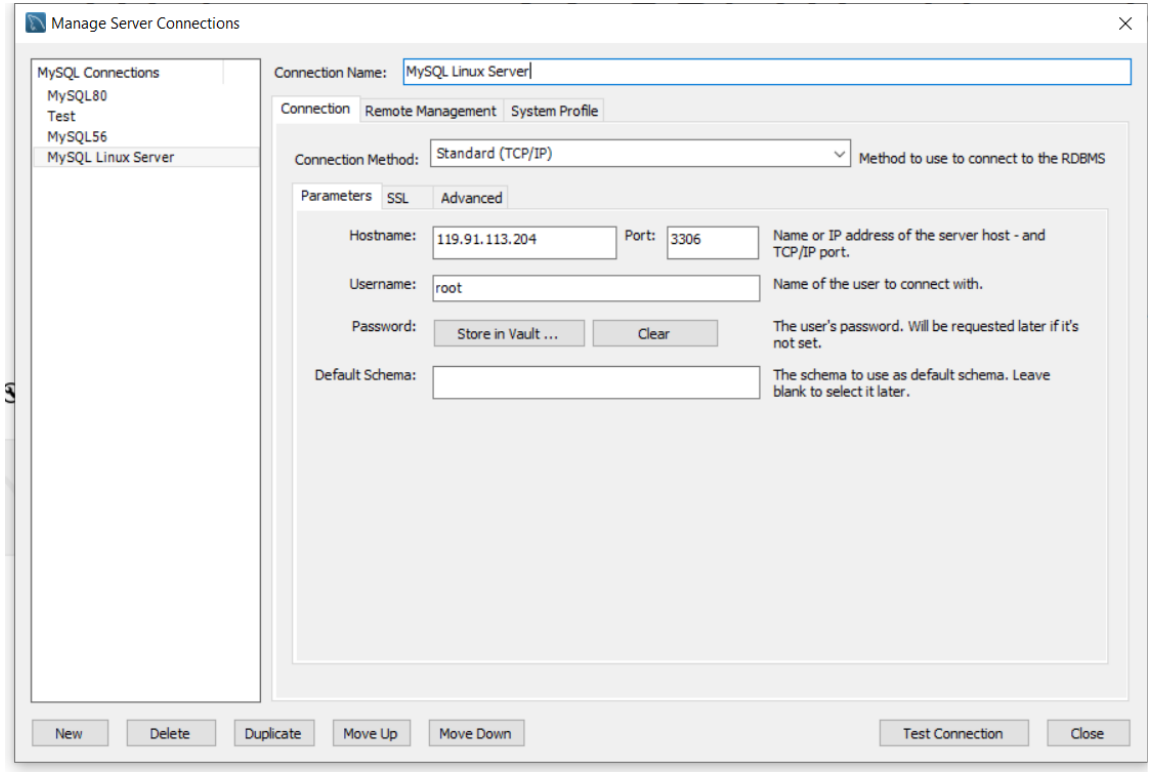
Then, we migrate our databases (including tables) to the MySQL Linux Server.

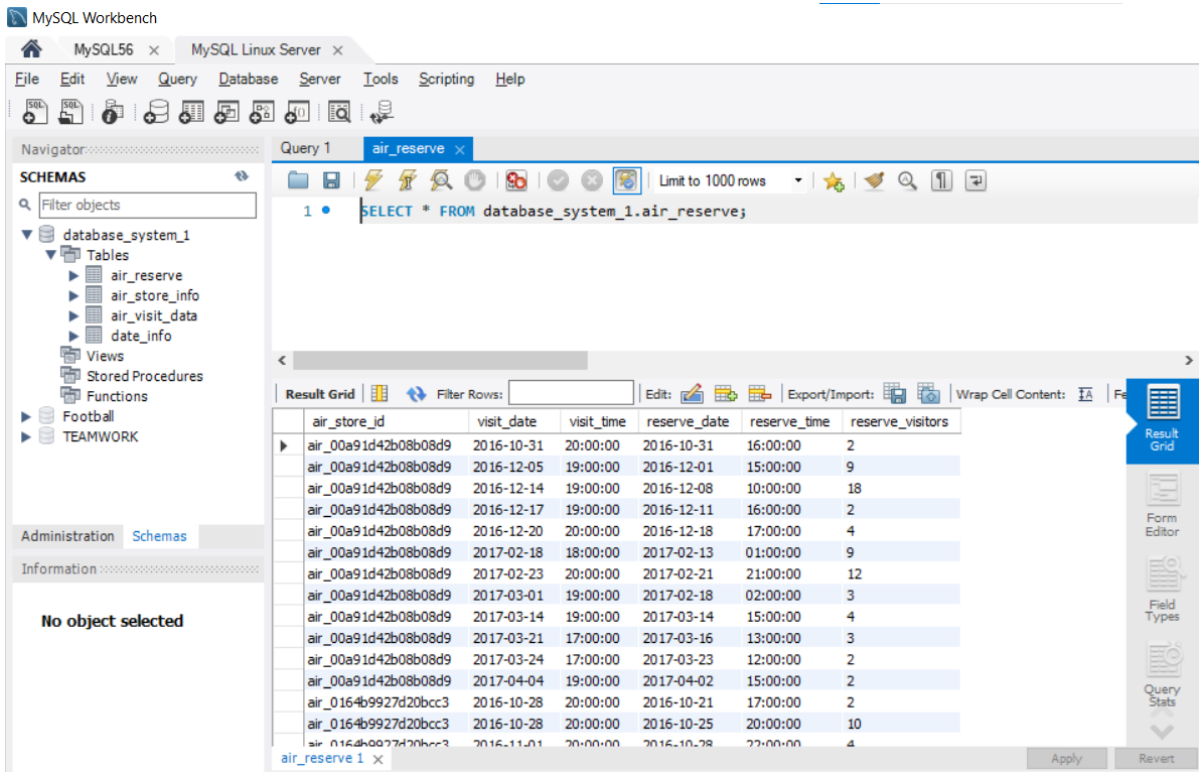


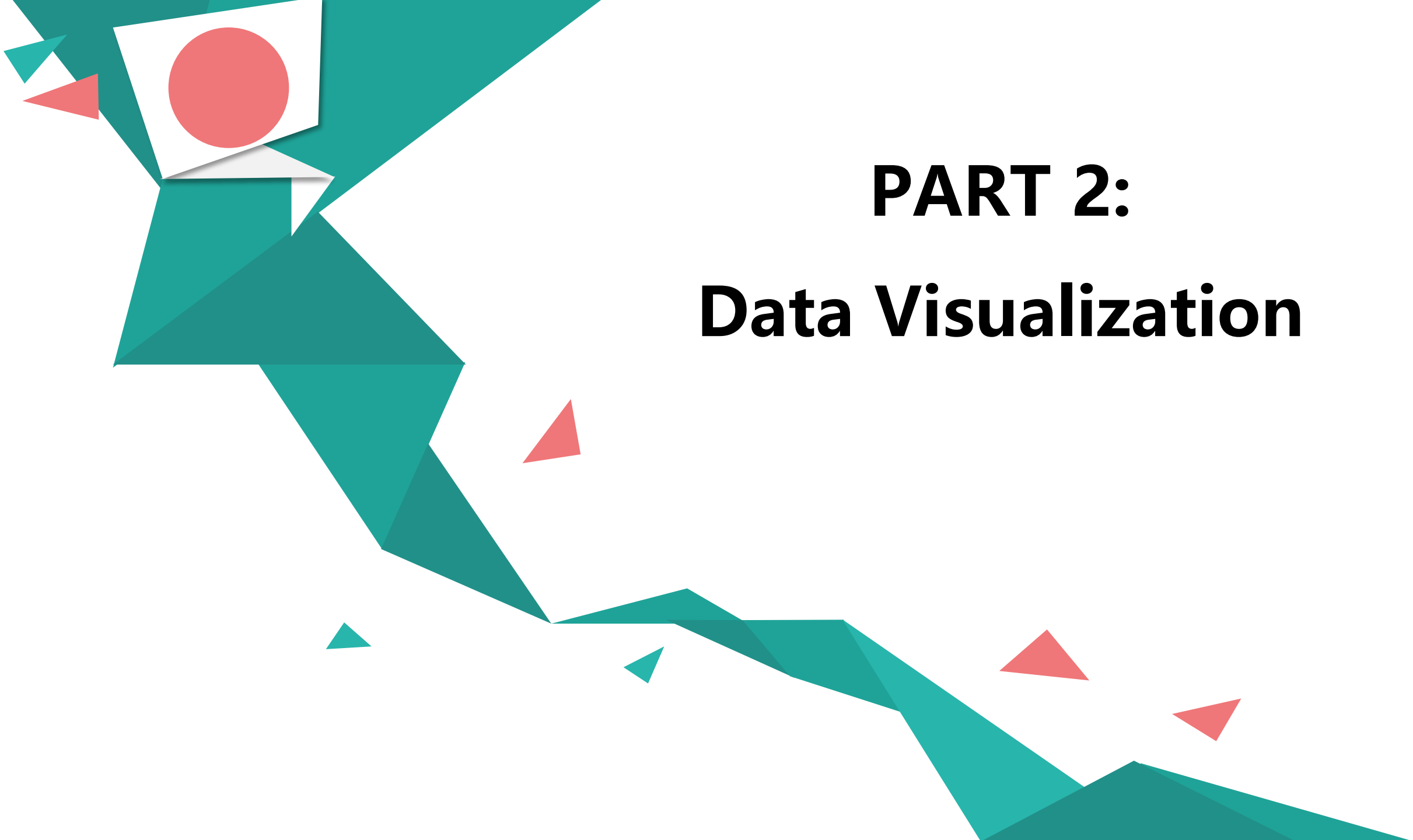


Retrieve Data From Server

Now, we can access the data from the server with any computer, it frees the constraint that the data can only be obtained from the local server.





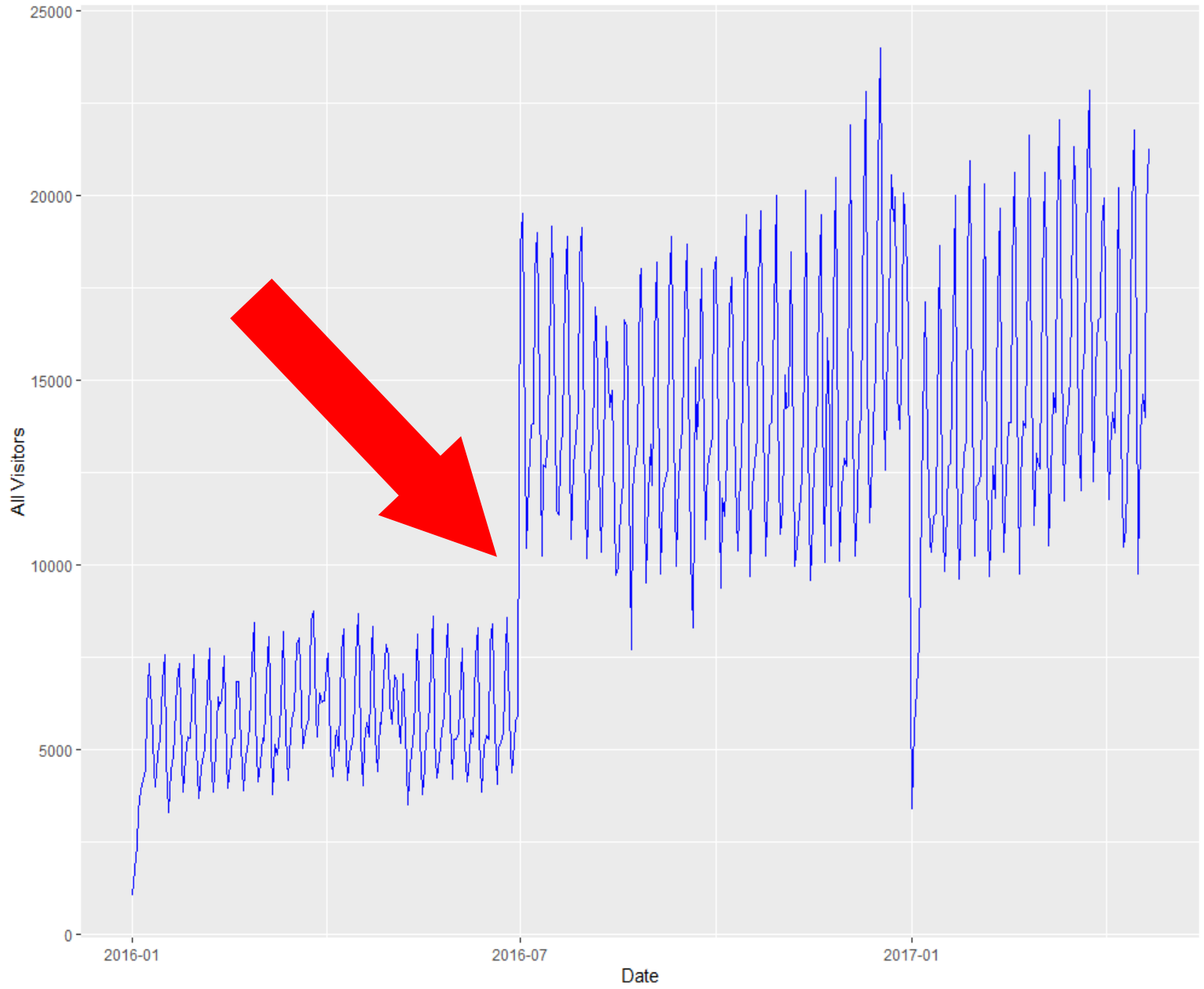


PART 2:

Data Visualization

Data Visualization

The number of visitors in each day



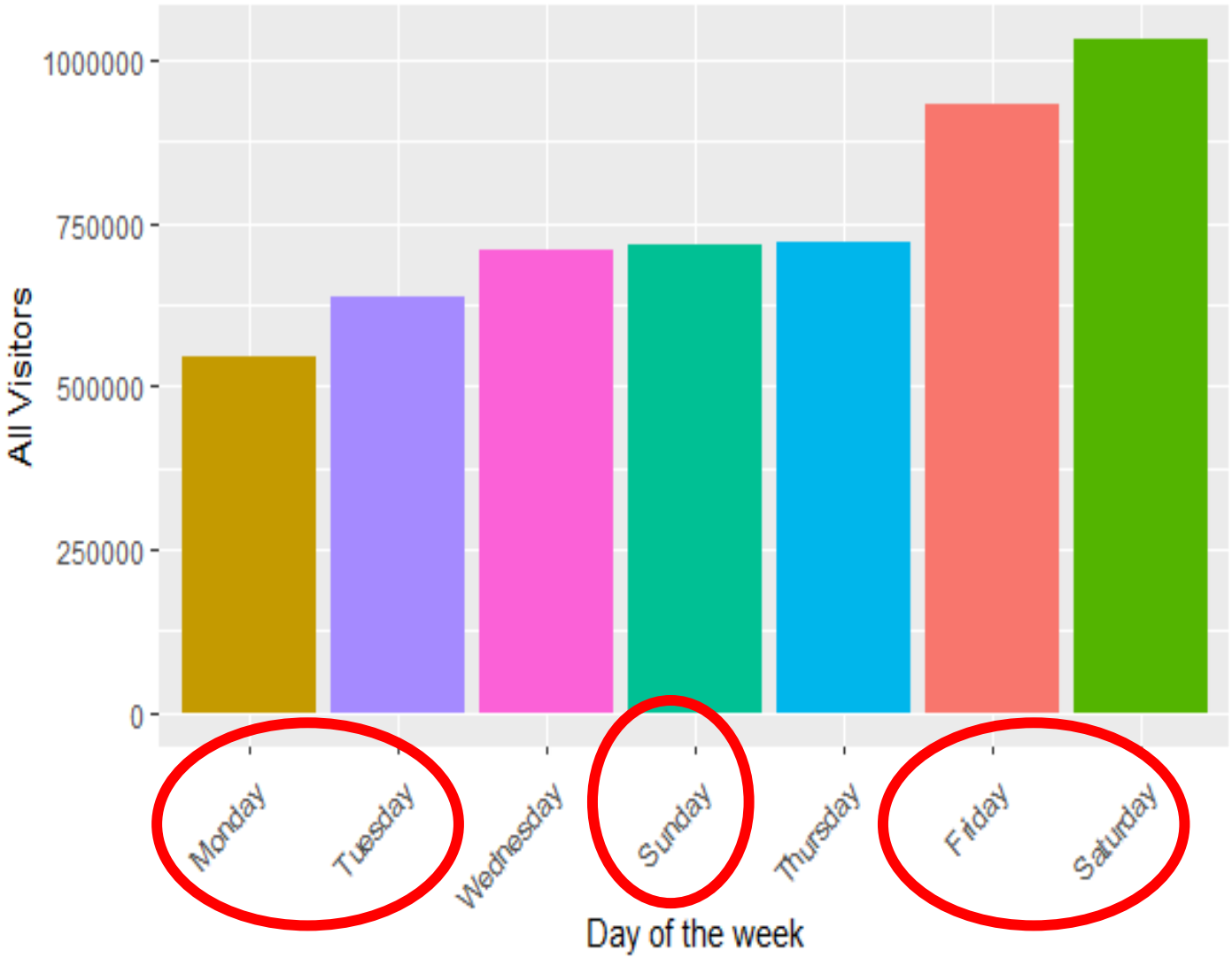
We plot the total number of visitors **per day** over the full time range.

We find:

- There is an interesting **long-term step structure** in the overall time series. This might be **related to new restaurants being added** to the data base.
- In addition, we already see a **periodic pattern** that most likely **corresponds to a weekly cycle**.

Data Visualization

The number of visitors from Monday to Sunday



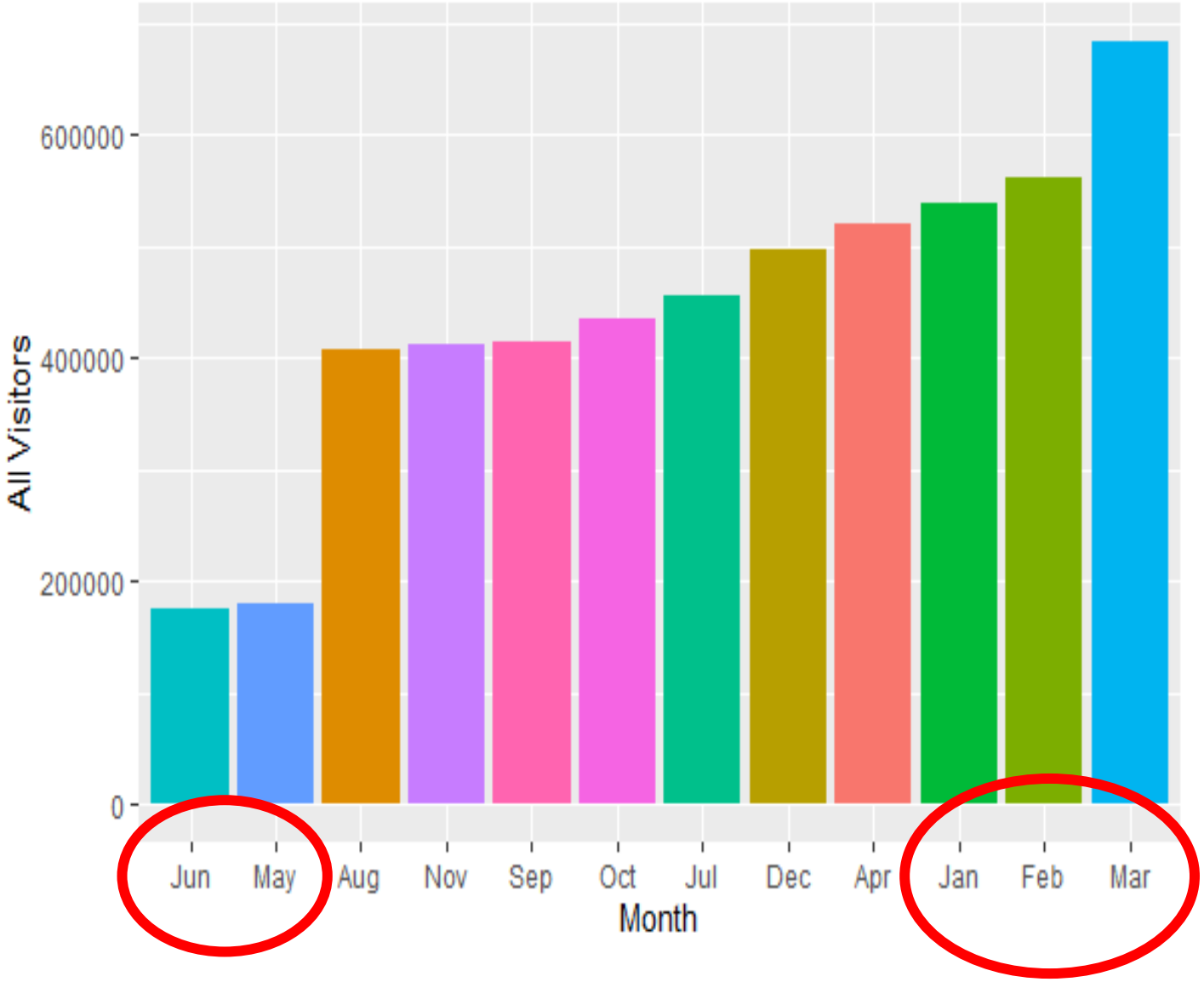
We plot the total number of visitors **per weekday** and **per weekend** over the full time range.

We find:

- **Friday** and **Saturday** appear to be the **most popular** days; which is to be expected. Sunday is not one of the popular days; which probably because the restaurant take a rest on Sunday.
- **Monday** and **Tuesday** have the **lowest numbers** of average visitors.

Data Visualization

The number of visitors on each month



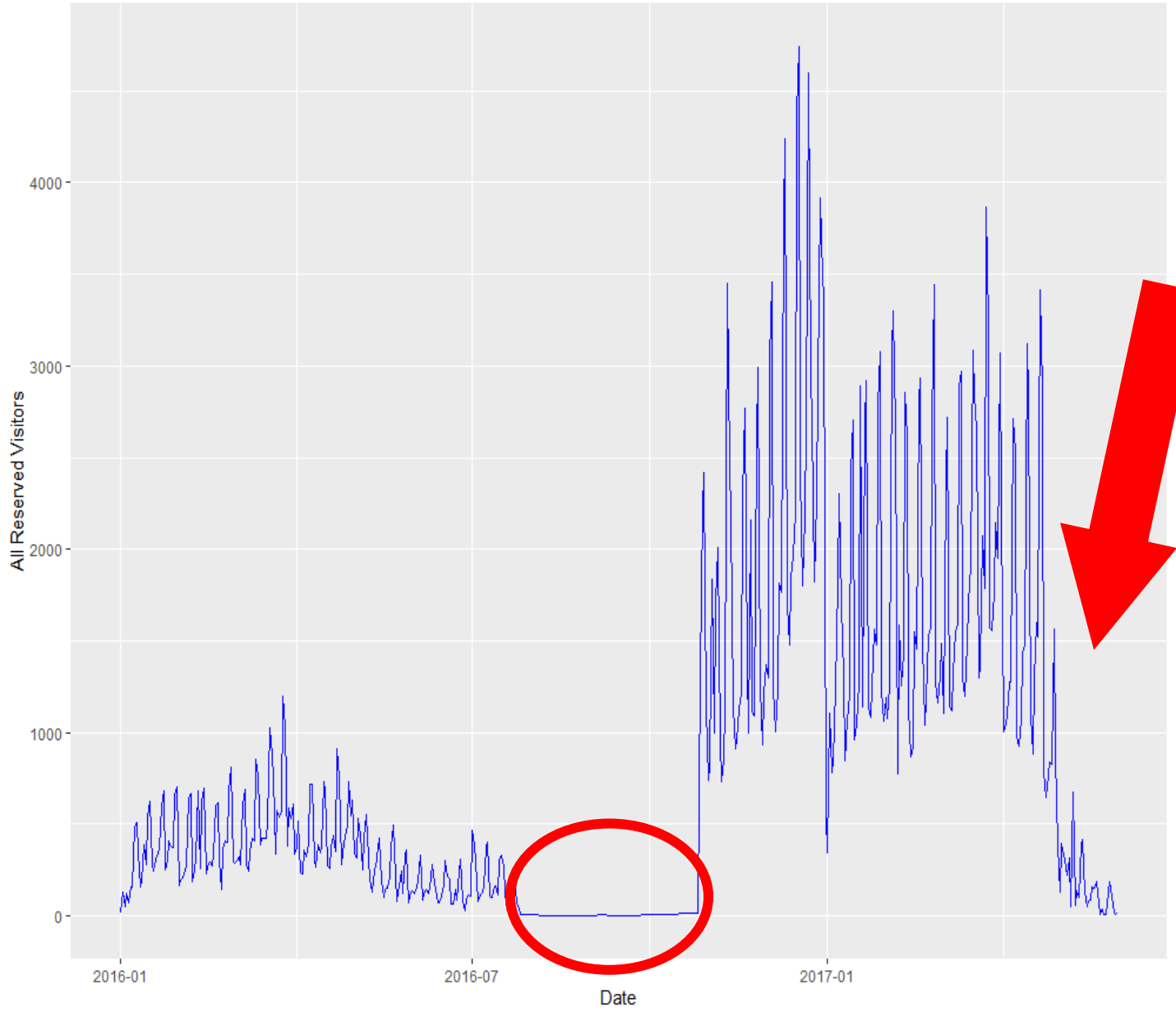
We plot the total number of visitors per **month** over the full time range.

We find:

- Also during the year there is a certain amount of variation.
- The period of **January - March** is consistently **busy**.
- **June** and **July** are the **easiest** months.
- Personally, I think the popularity of the month is related to the holiday.

Data Visualization

The number of reserved visitors on each day

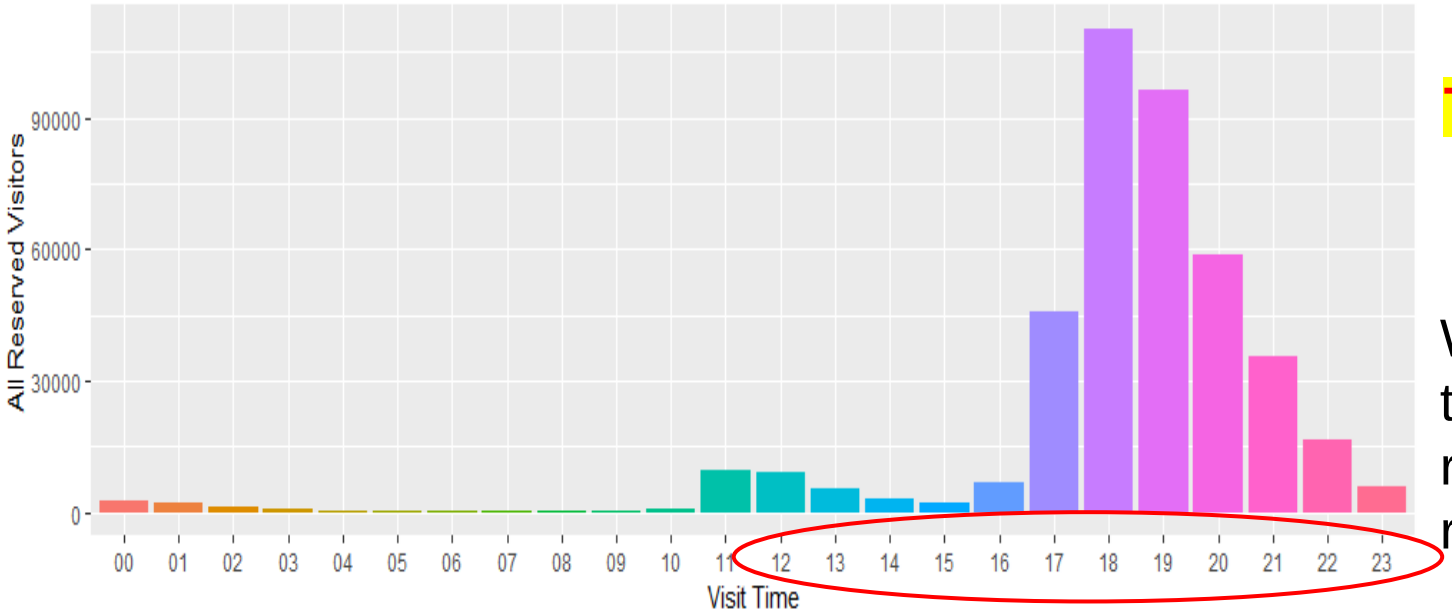


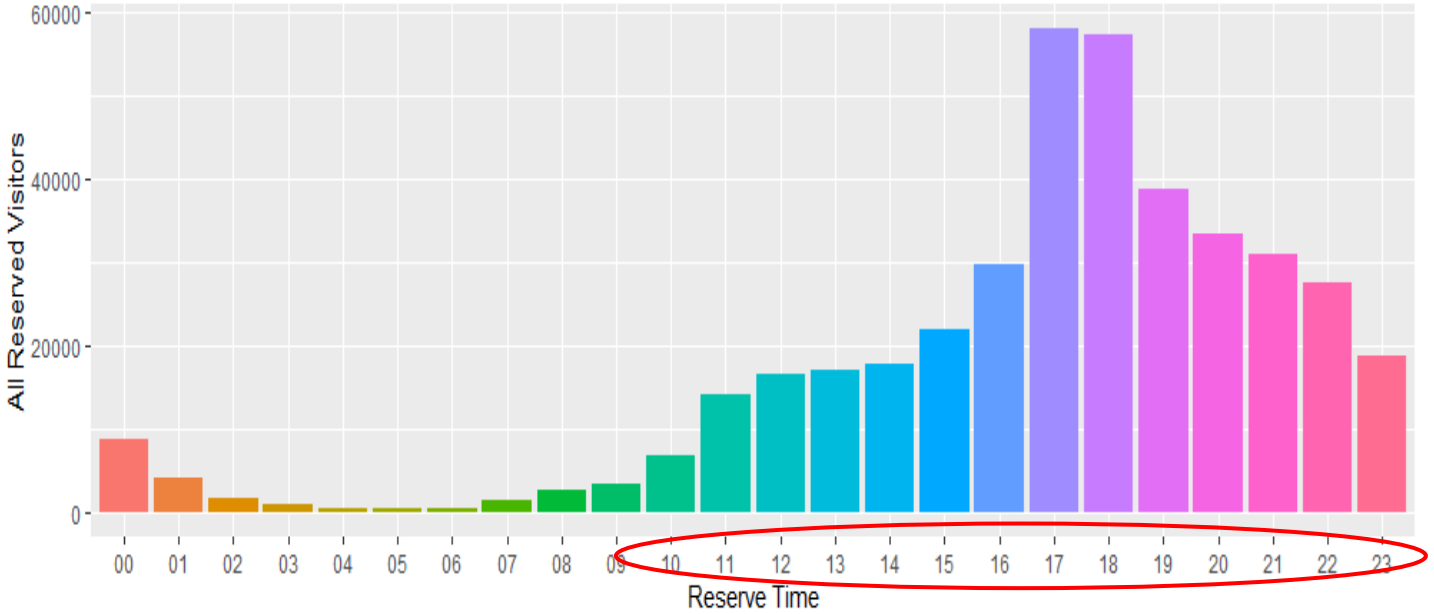
We plot the total number of reserved visitors per **day** over the full time range.

We find:

- There were much **fewer** reservations made in **2016,** even **none** for a long stretch of time (Personally, I think it's data loss).
- In **2017** the visitor numbers **stayed strong**.
- The **great decline** we see **after the first quarter** is most likely related to these reservations being **at the end of the time frame**, which means that **long-term reservations would not be part of this data set**.

Data Visualization





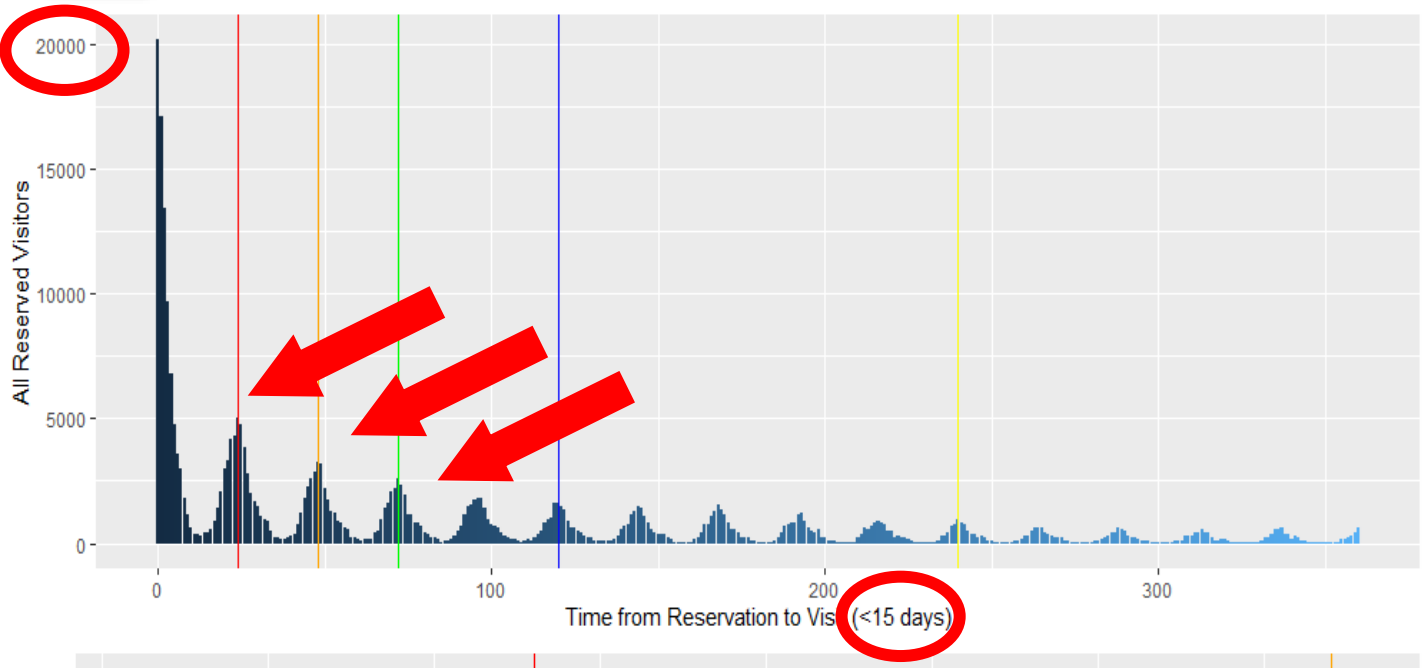
The number of reserved visitors to visit restaurants on each hour
and
The number of reserved visitors to reserve on each hour

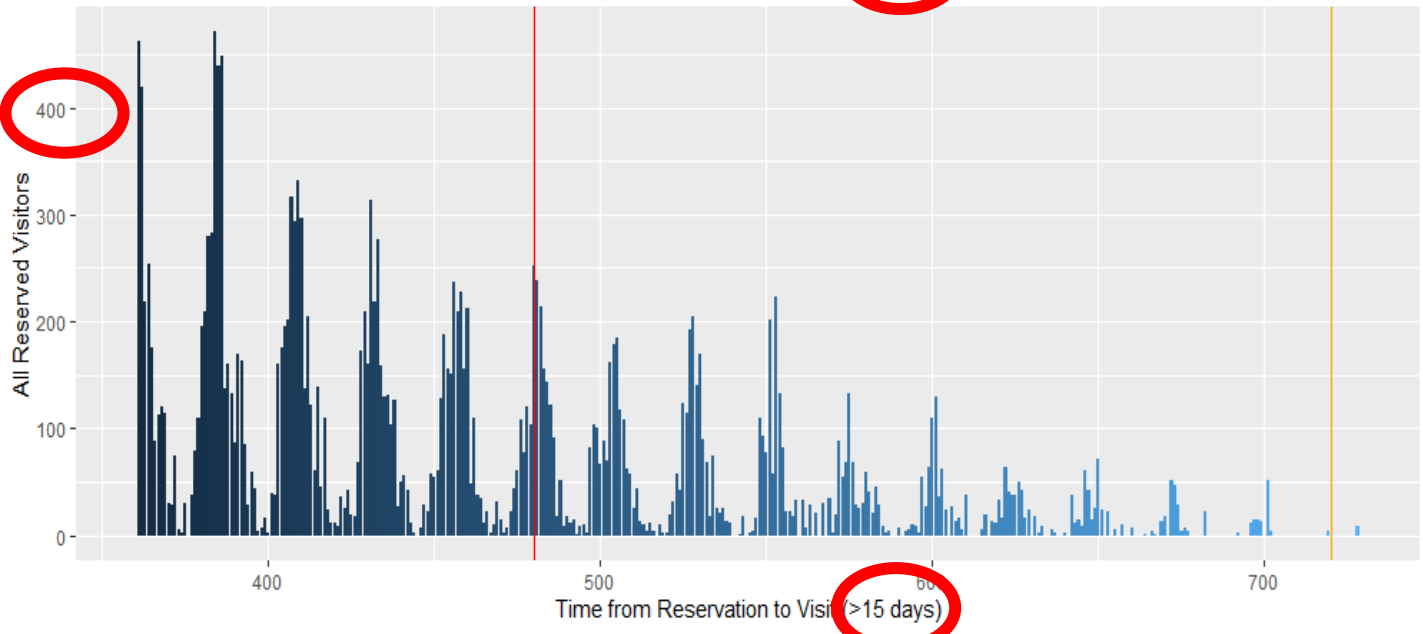
We plot the total number of reserved visitors to do the reservations and visit the restaurants the per **hour** over the full time range.

We find:

- **Reservations** are made typically for the **dinner hours in the evening**.
- The **mealtime** is about the **same** as the **reserved time**, which means that visitors usually **make their next meal reservation at the mealtime**

Data Visualization





The difference(interval) between reserve time and actual visit time

We plot the **difference**(interval) between **reserve time** and **actual visit time** within 15 days and over 15 days throughout the whole time range.

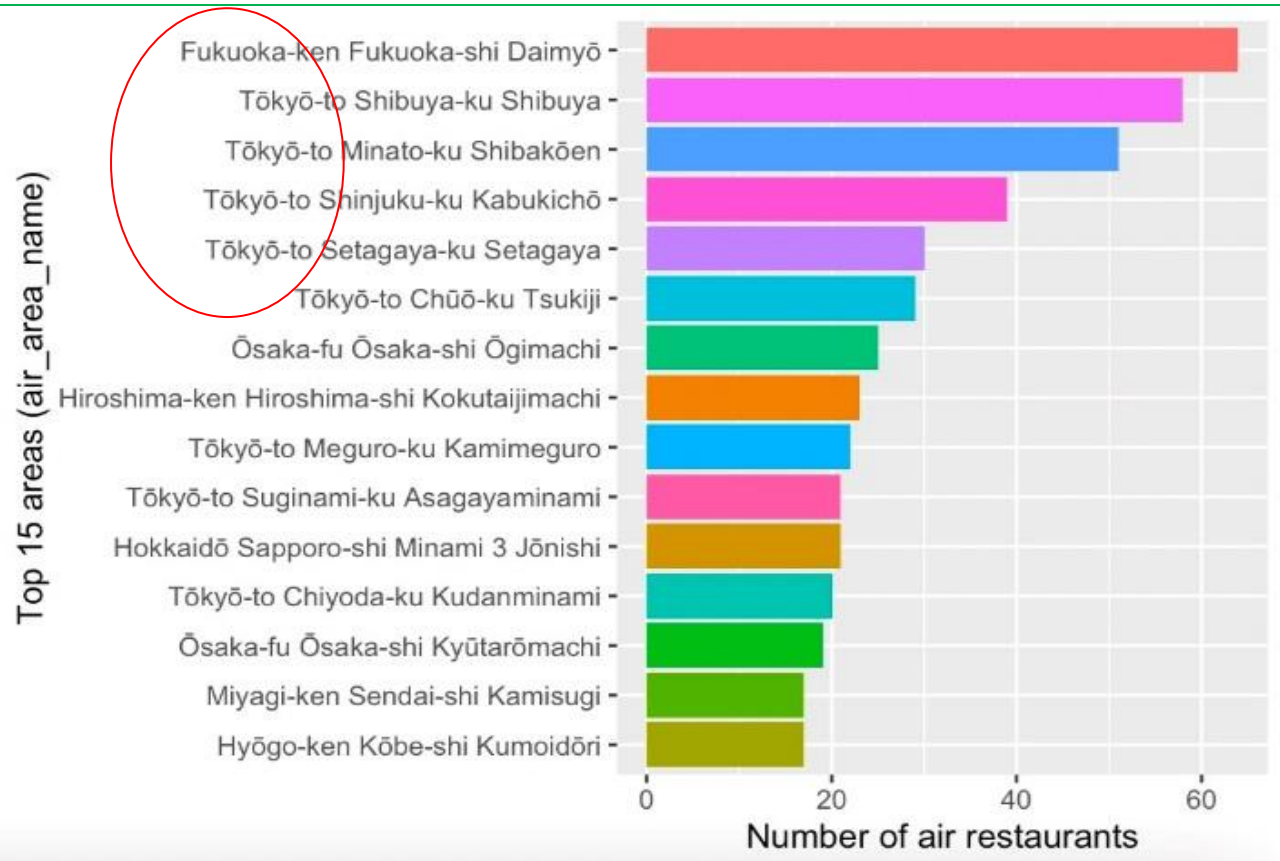
We find:

- The time, here shown in hours, between making a reservation and visiting the restaurant follow a **nice 24-hour pattern**. The most popular strategy is to **reserve a couple of hours before the visit**, but if the reservation is made more in advance, then it seems to be common to **book a table in the evening for one of the next evenings**.

- There are still **many** reservations for **more than 15 days**, but very **few** for **more than two months**.

Data Visualization

The popular list of different types of Restaurant and The popular list of different areas

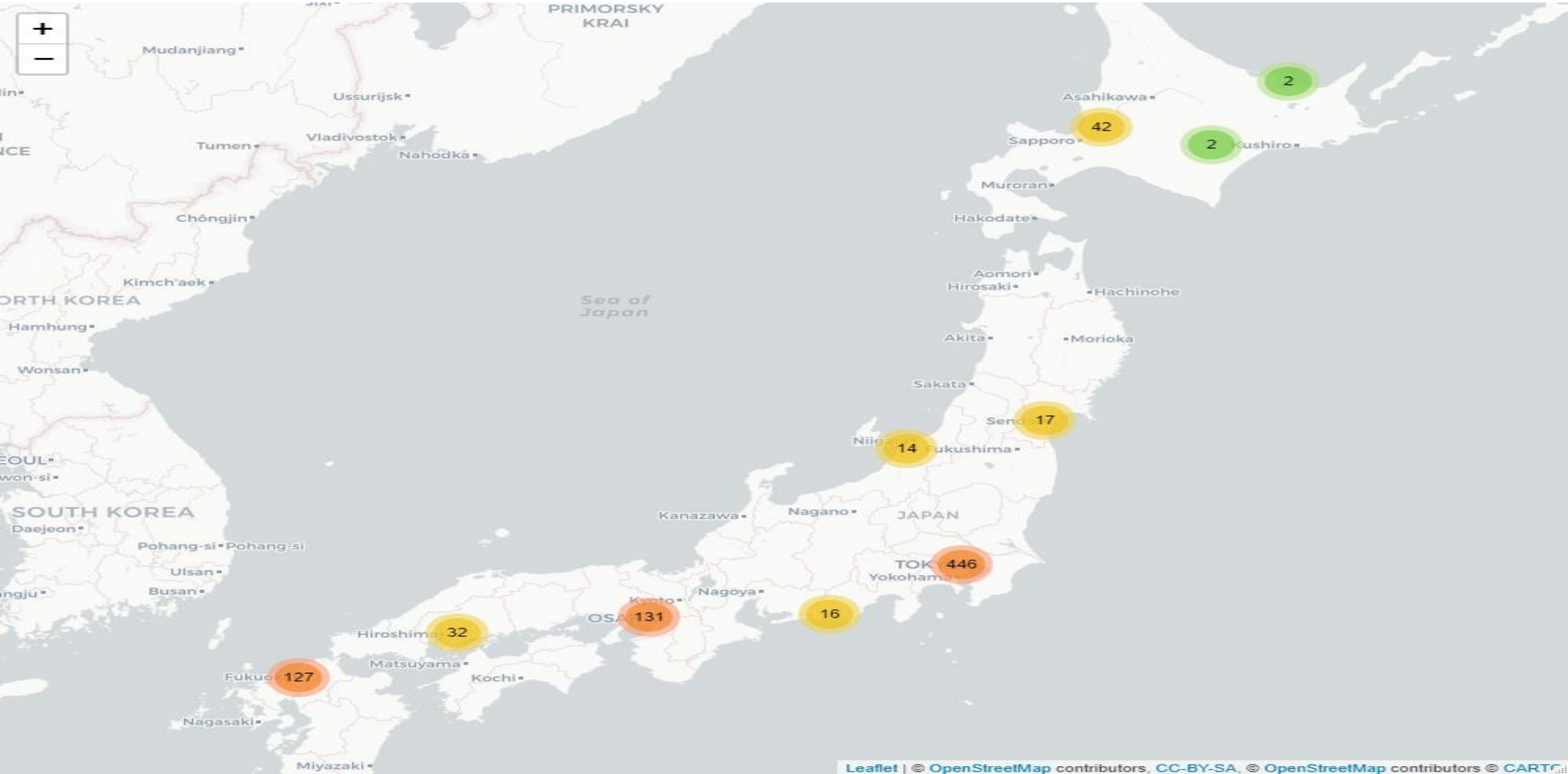


We find:

- There are **lots of Izakaya and Cafe**.
- **Fukuoka** has the **largest number** of air restaurants per area, followed by **many many Tokyo areas**!

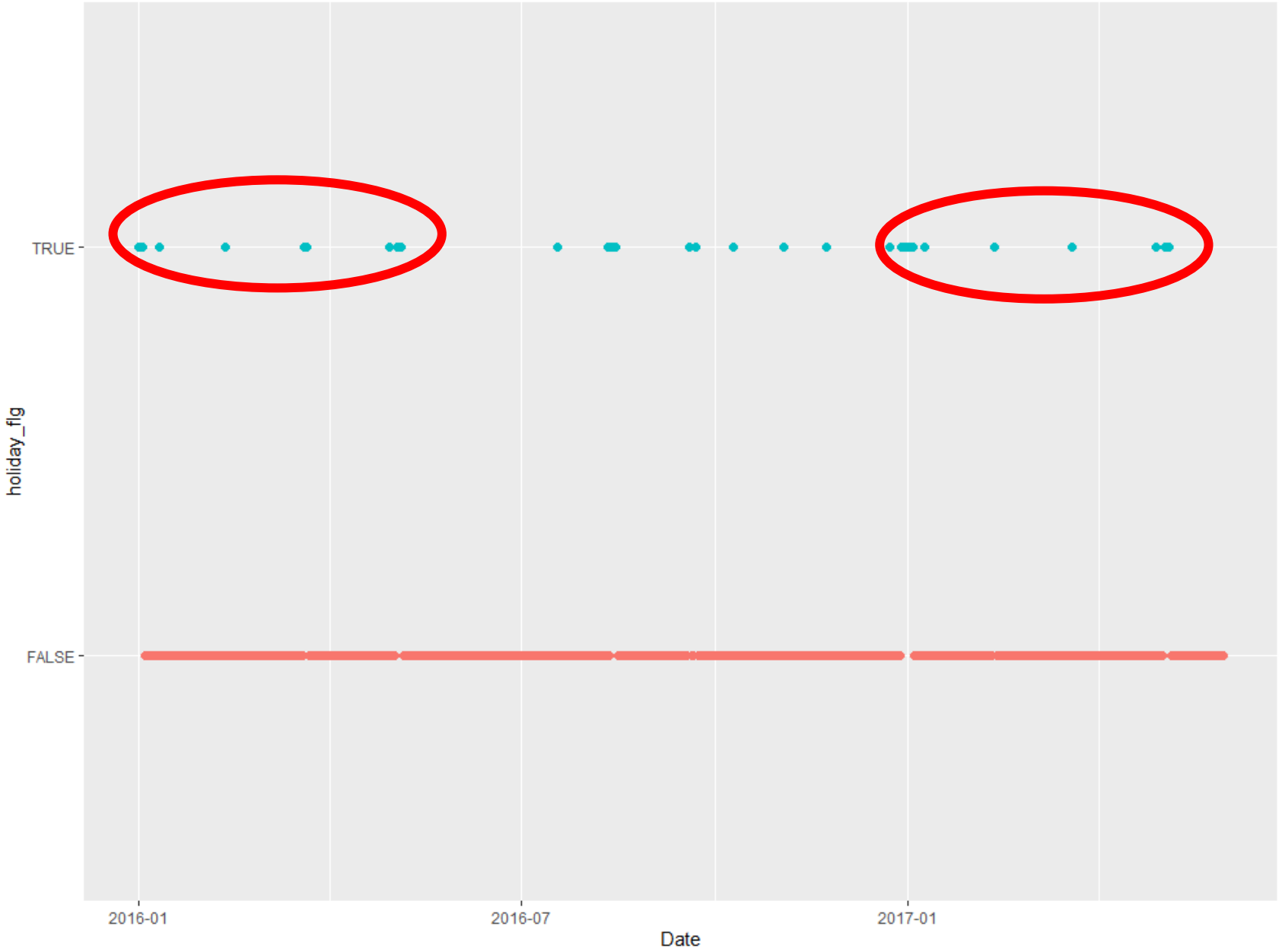
Data Visualization

The Map of all restaurants' location (Dynamic)



Data Visualization

Holiday Distribution (All Time)



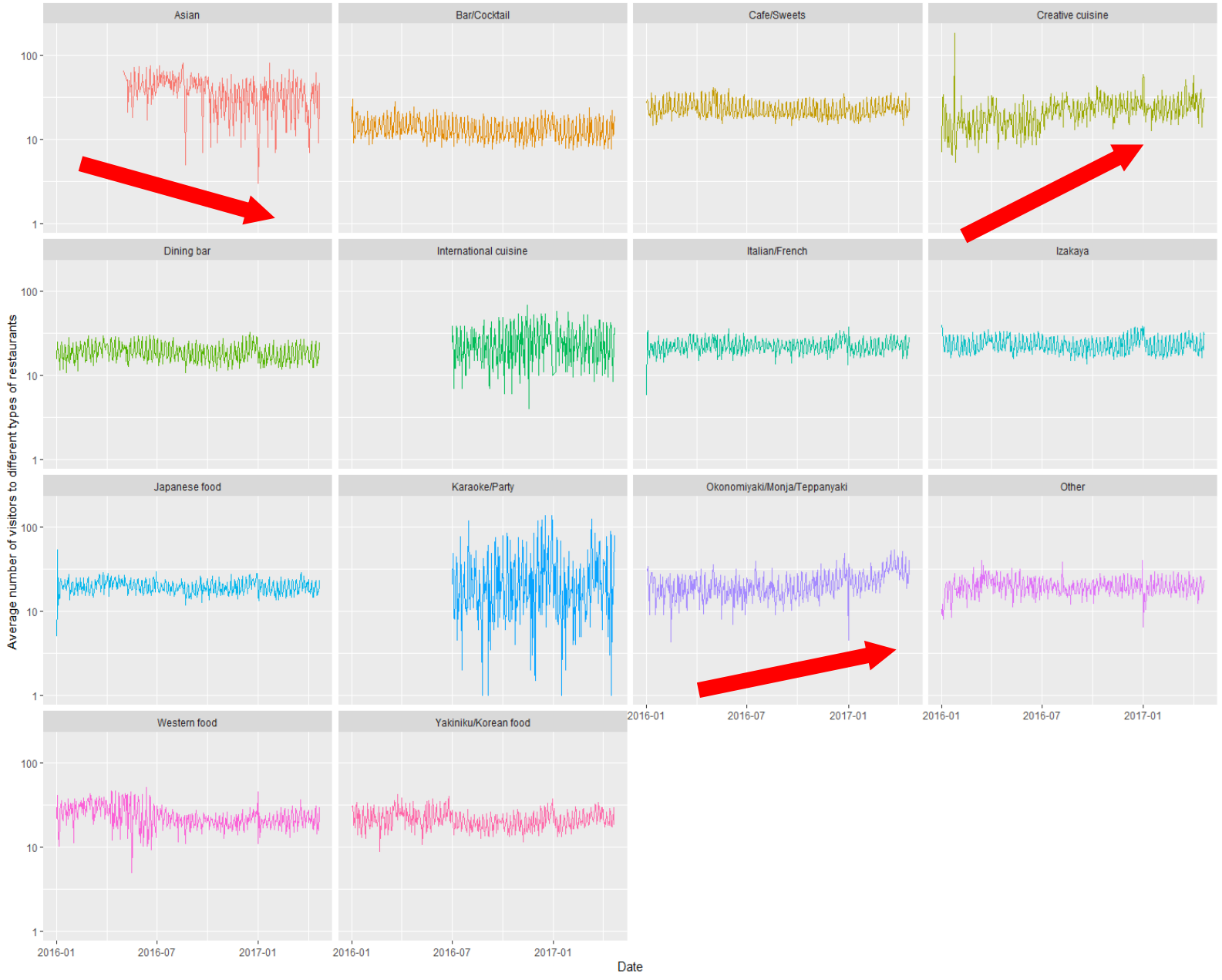
We plot how many there are in total and how they are distributed throughout the whole time range.

We find:

- The **same days** were holidays **from January to May in 2016 as in 2017**.
- There are about **7% holidays in our data.**

Data Visualization

The Average number of visitors to different types of restaurants on each day



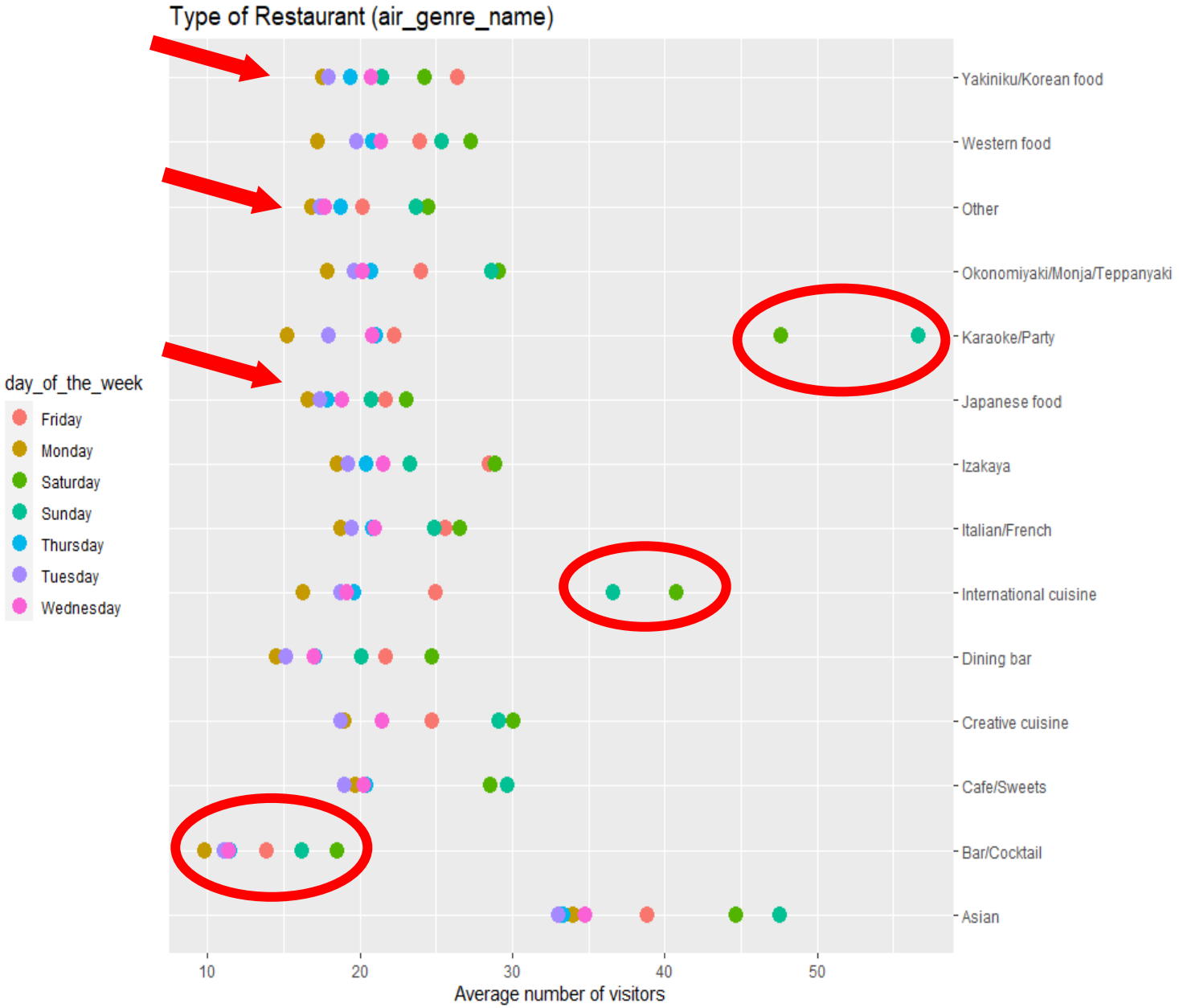
Our first plot of the multi-feature space deals with the average number of air restaurant visitors with different types of restaurant. We use a facet plot to distinguish the types of restaurant for the 14 categories.

We find:

- The **mean values** range **between 10 and 100** visitors per genre per day. Within each category, the **long-term** trend looks reasonably **stable**.
- There is an **upward trend for "Creative Cuisine" and "Okonomiyaki"**. while the popularity of "**Asian" food has been declining** since late 2016.

Data Visualization

For different types of restaurants, WHEN do the visitors like to go



Greenish is the weekend.

We find:

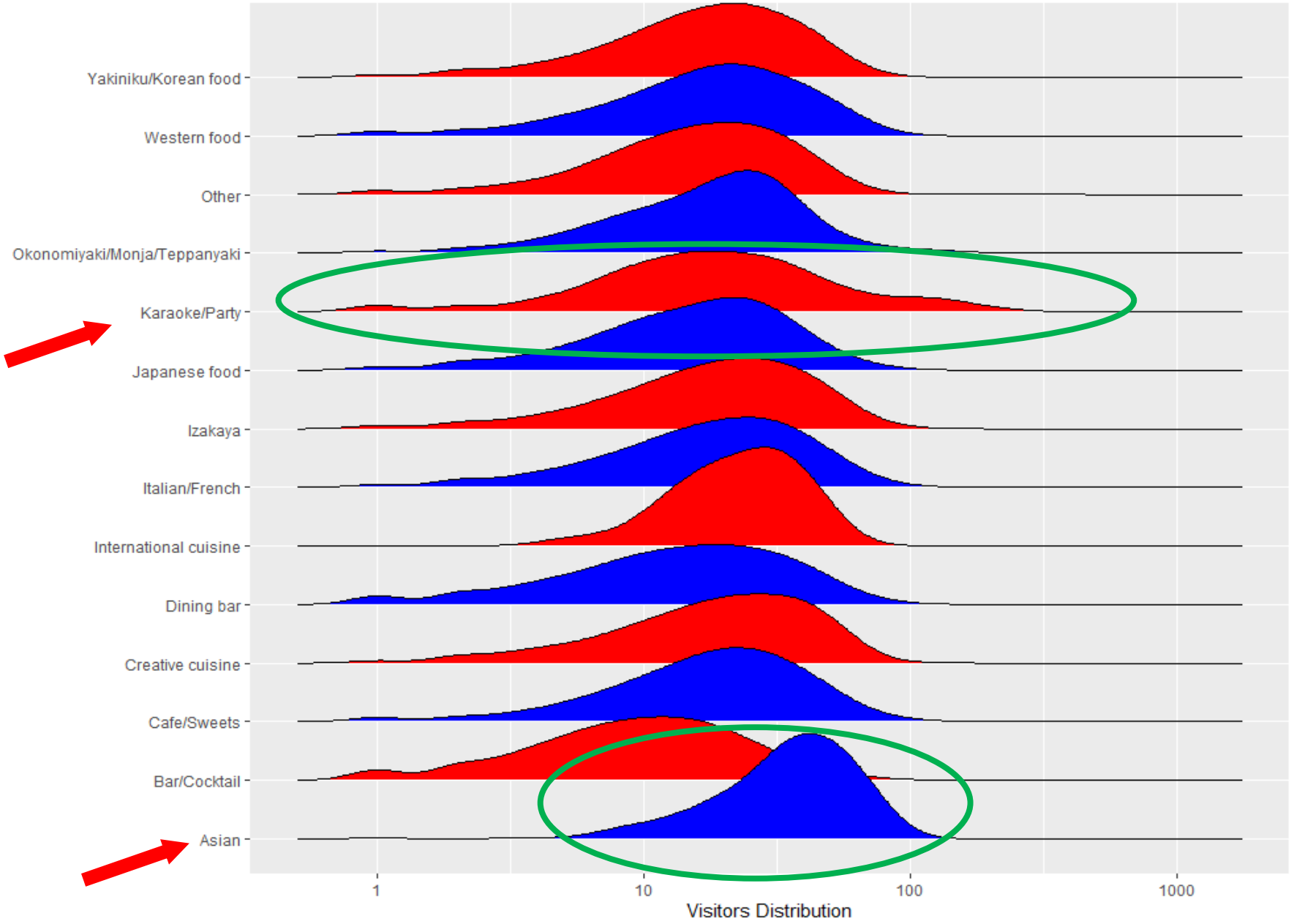
- The **biggest difference between weekend and weekdays** exists for the **“Karaoke”** bars, which **rule the weekend**. Similarity**,** the **“International” cuisine also rule the weekend**.

- No genre really goes against the trend of busier weekends. The **smallest variations** are in the generic “**Other**” category, the “**Japanese**” food, and the “**Korean**” which is the only category where Fridays are the busiest days.

- “**Bars/Cocktail**” are notably **unpopular** overall.

Data Visualization

For different types of restaurants, HOW MANY people usually go together



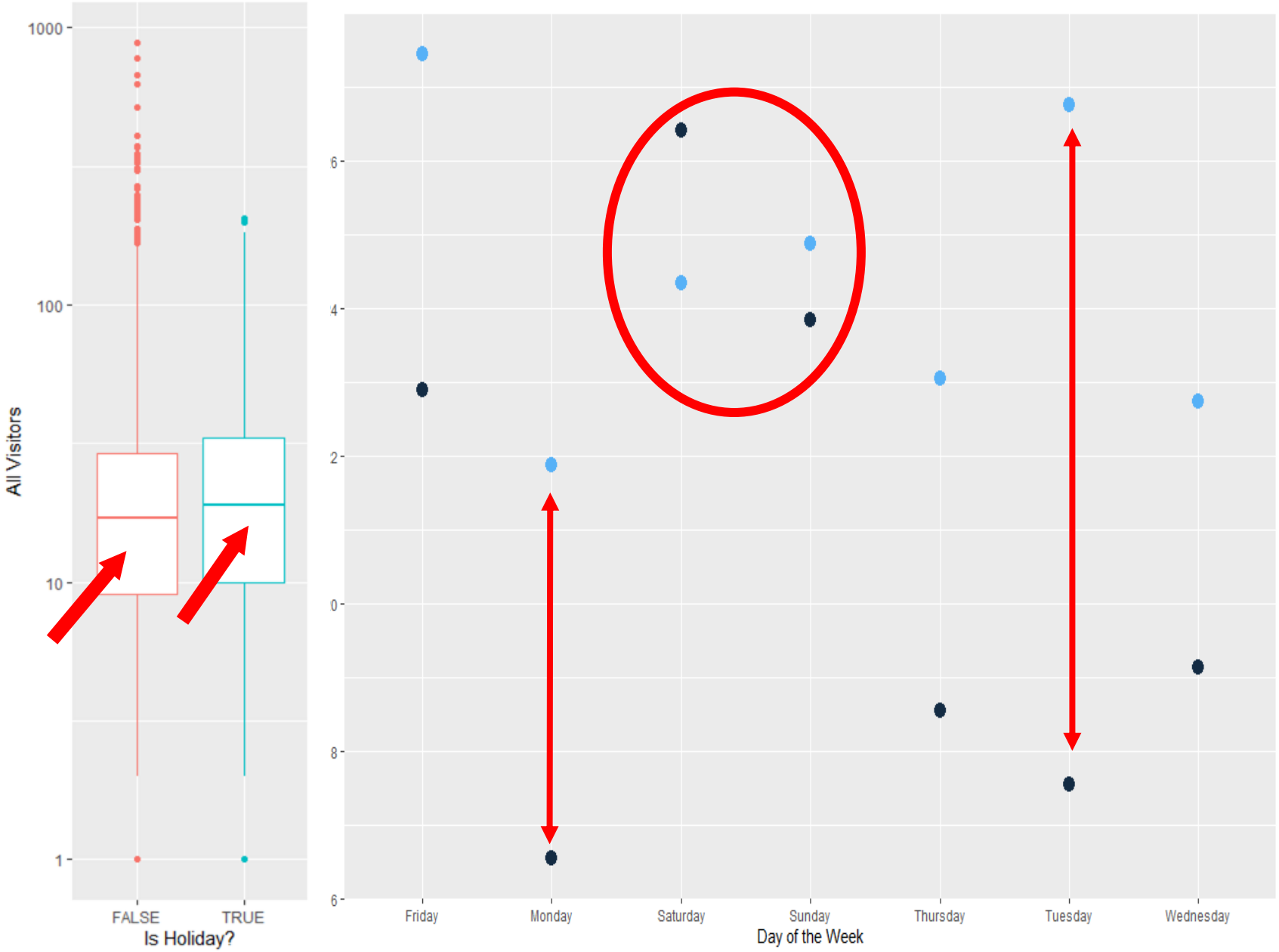
We find:

The density curves confirm the impression we got from the week-day distribution:

- the "**Asian**" restaurants have **rarely less than 10 visitors per date**
- the "**Karaoke**" places show a very **broad distribution due to the strong impact of the weekends**.

Data Visualization

The impact of holidays



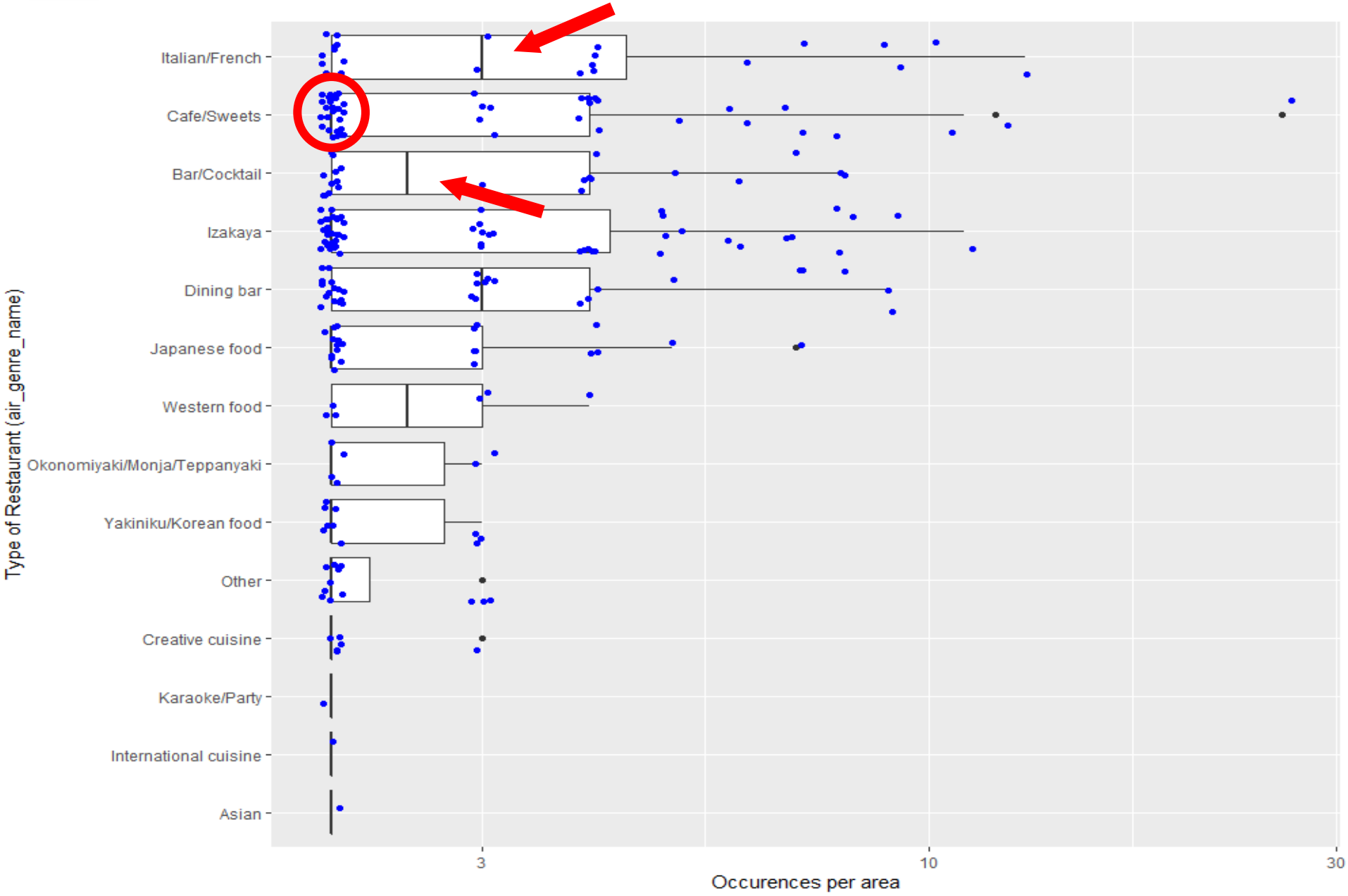
We will study the influence of holidays on our visitor numbers by comparing the statistics for days with holidays vs those without holiday flag:

We find:

- Overall, **holidays don't have any impact on the average visitor numbers (left panel).**
- For **weekend,** holiday has **little impact** on the visitor numbers.
- **holiday** has much **more effect for the weekdays**; especially Monday and Tuesday.

Data Visualization

For different types of restaurant and different types of areas, the distribution of visitors



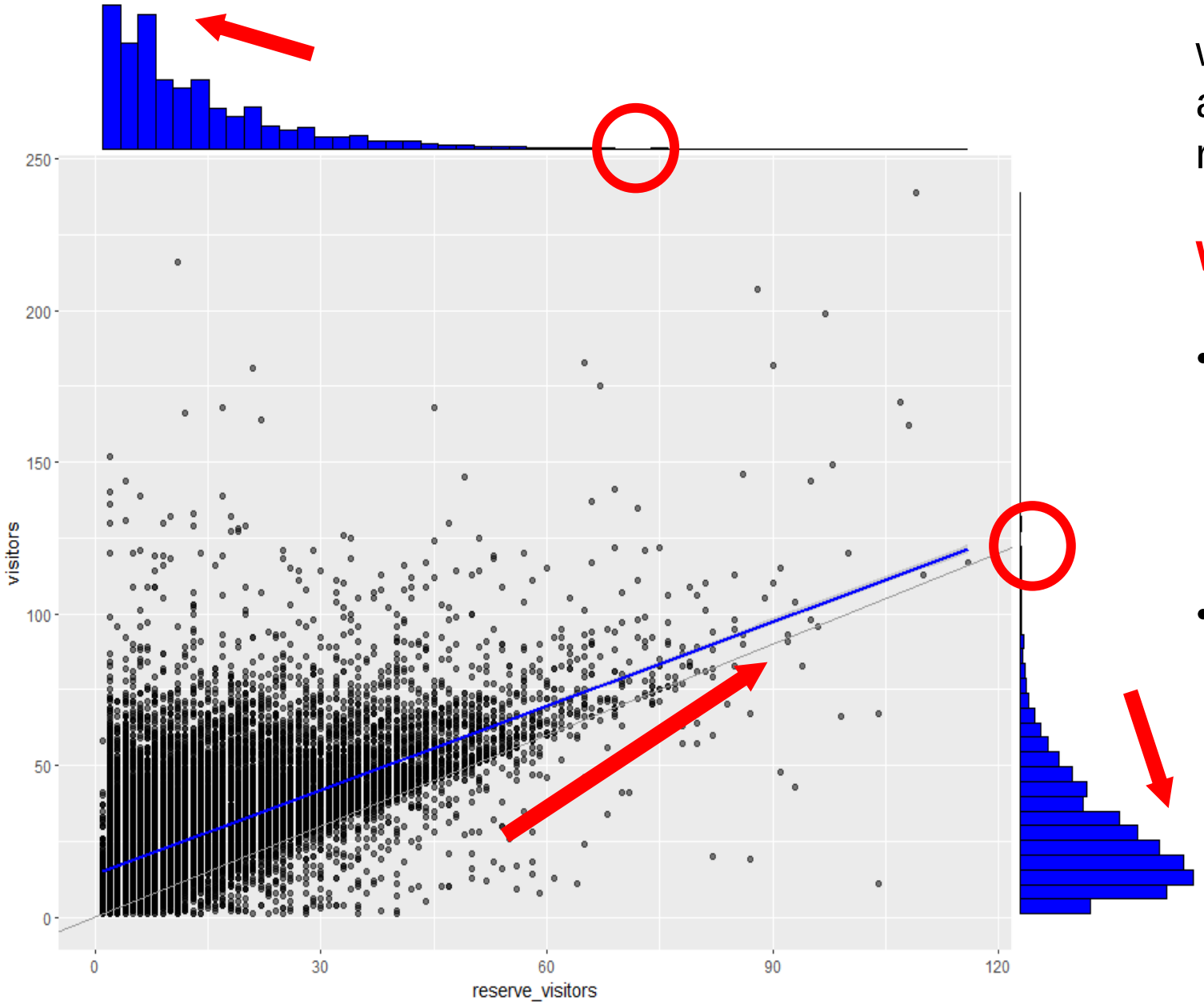
We find:

- Only **few types of restaurants have medians of more than 2 restaurants per area**. Like: **“Italian/French”** restaurants or “**Bar/Cocktail**” places

- **“Cafes” have the highest number with 26 occurences in a single area** (Fukuoka-ken Fukuoka-shi Daimyō).

Data Visualization

The relationship between reserved visitors and non-reserved visitors (visitors) on each day



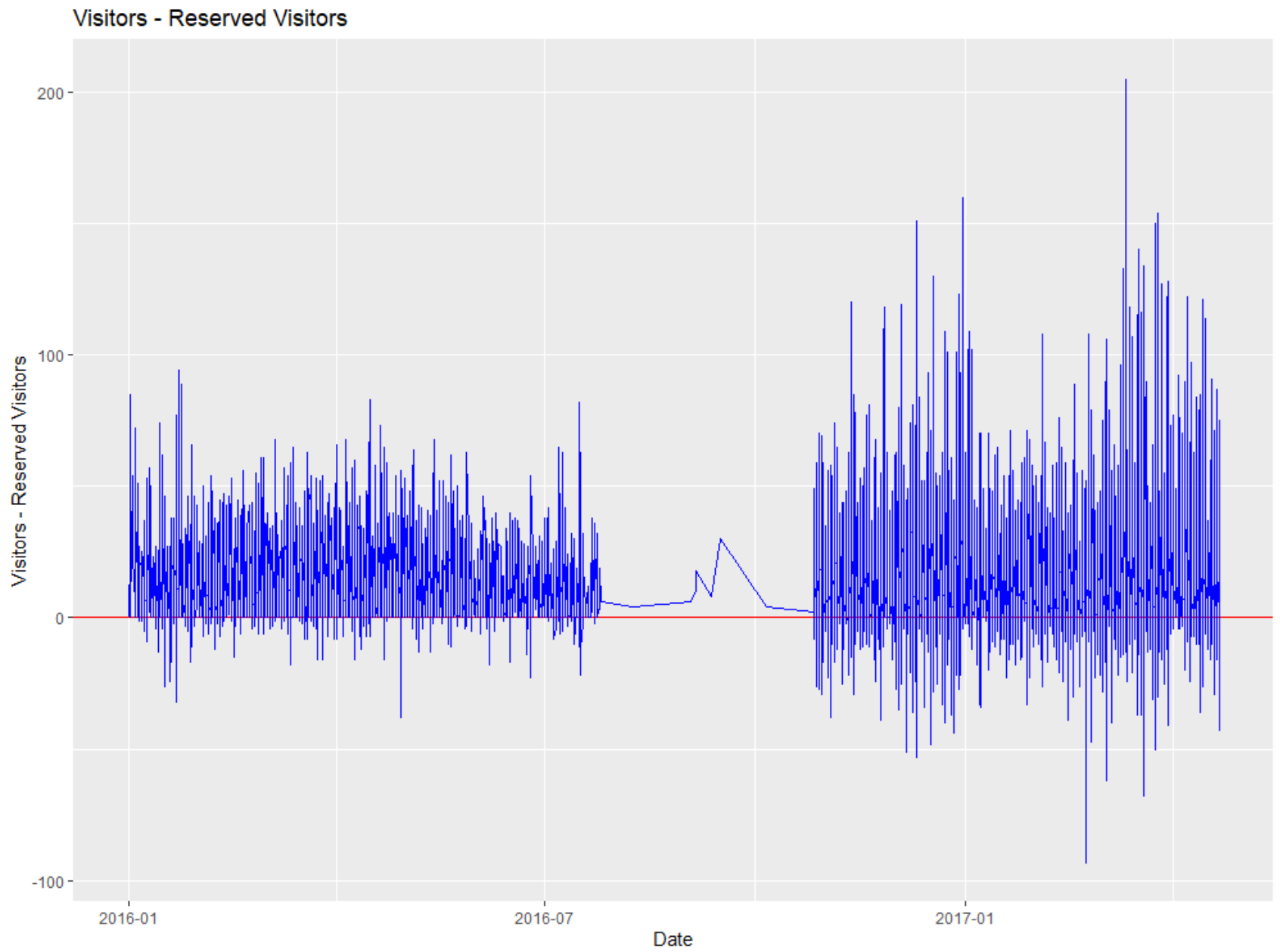
we plot the total reserve_visitor numbers against the actual visitor numbers for the air restaurants.

We find:

- The histograms show that the **reserve_visitors and visitors numbers peak below ~20** and are **largely** confined to the range **below 100**.
- The scatter **points fall largely above the line of identity**, indicating that there were **more visitors than reserved visitors on one day**. This is not surprising, from my perspective, since a certain number of people will always be walk-in customers.

Data Visualization

The difference between reserved visitors and non-reserved visitors (visitors) on each day

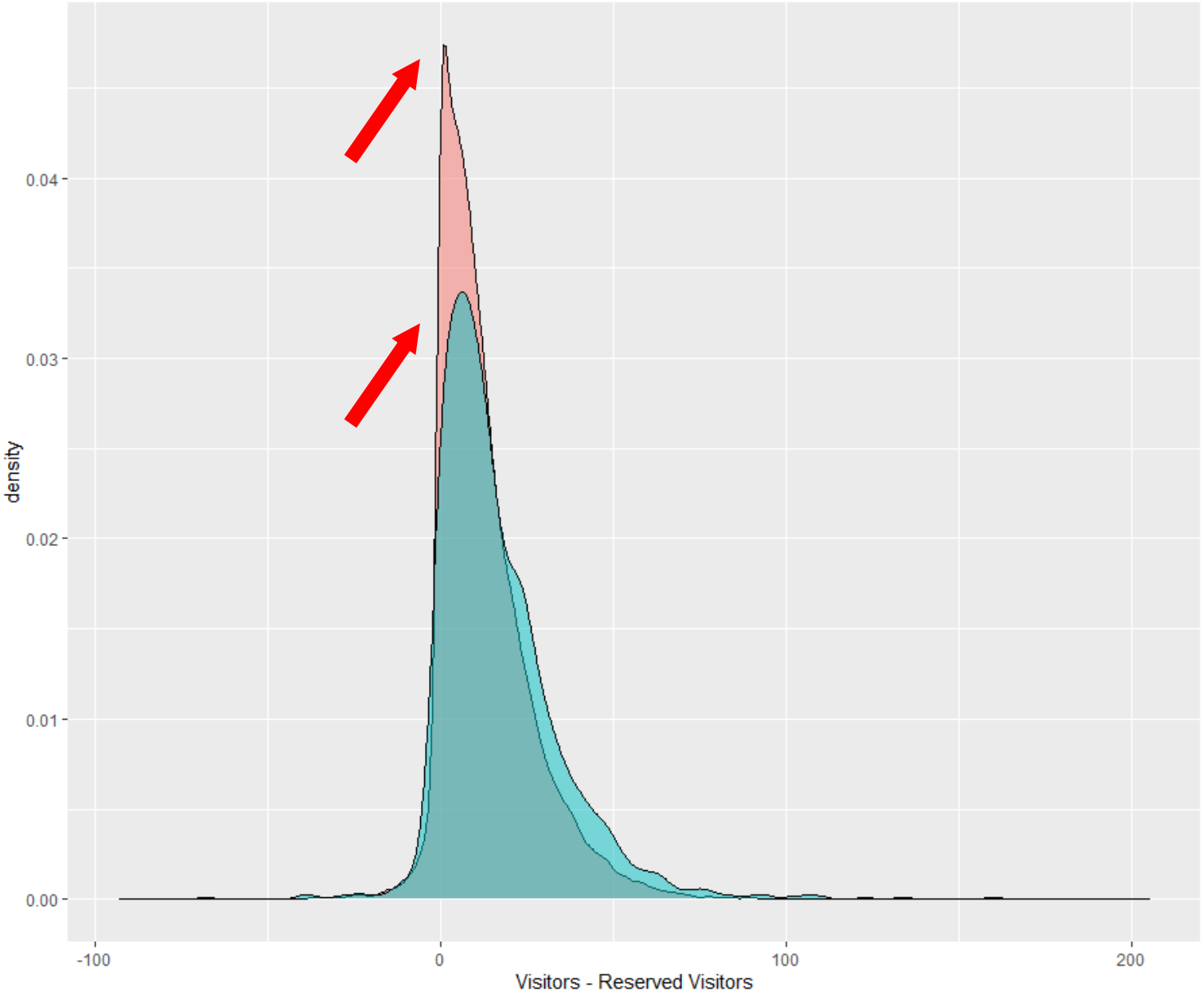


We find:

- While **the curves are predominantly above the baseline (i.e. more visitors than reservations)**,

Data Visualization

The density difference between reserved visitors and non-reserved visitors (visitors) on each day whether it is holiday or not



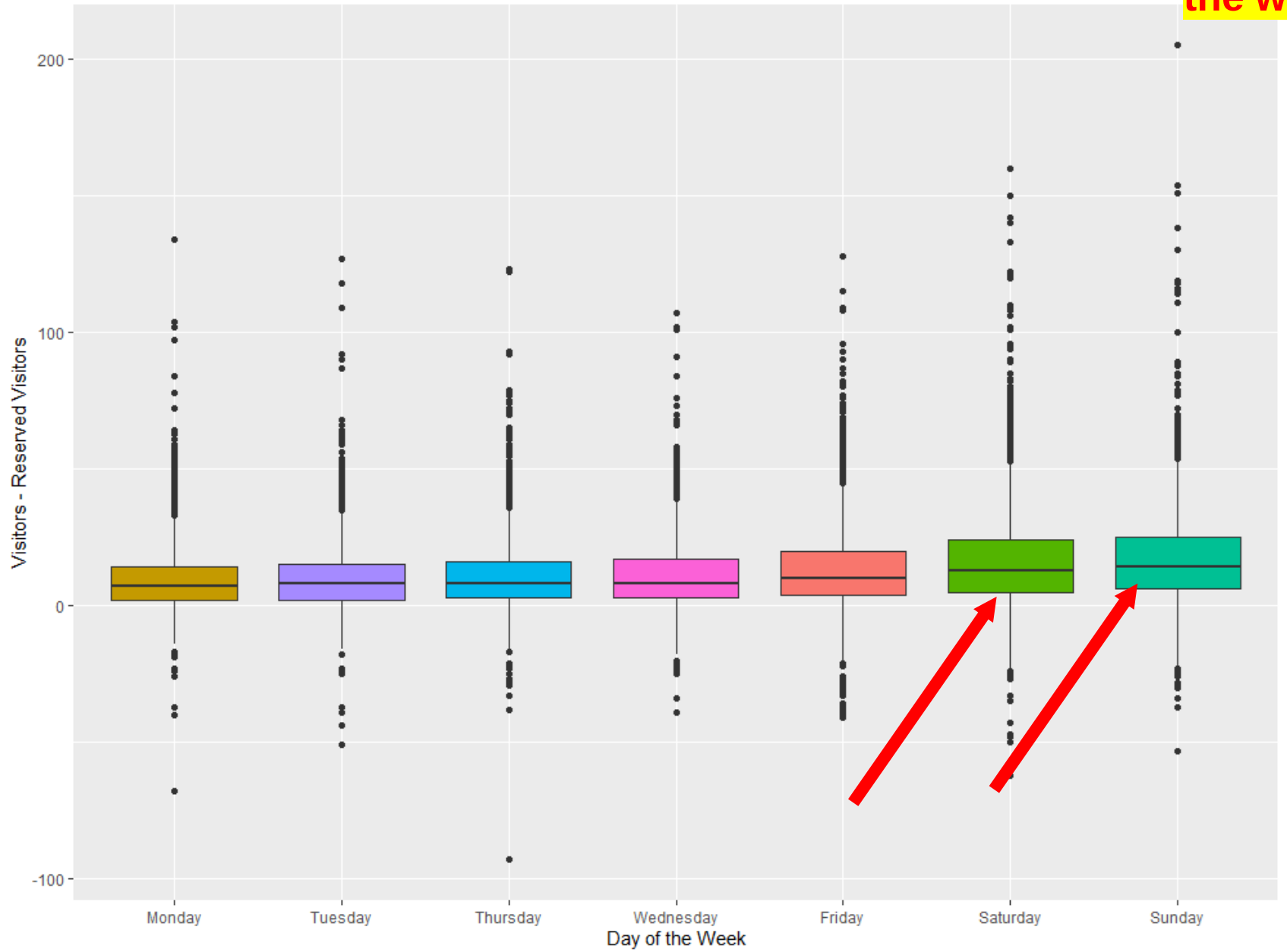
Finally, let's have a quick look at **the impact of holidays** on the discrepancy between **reservations** and **visitors**.

We find:

- On **holiday**, there are somewhat **higher** numbers of **visitors compared to reservations**.

Data Visualization

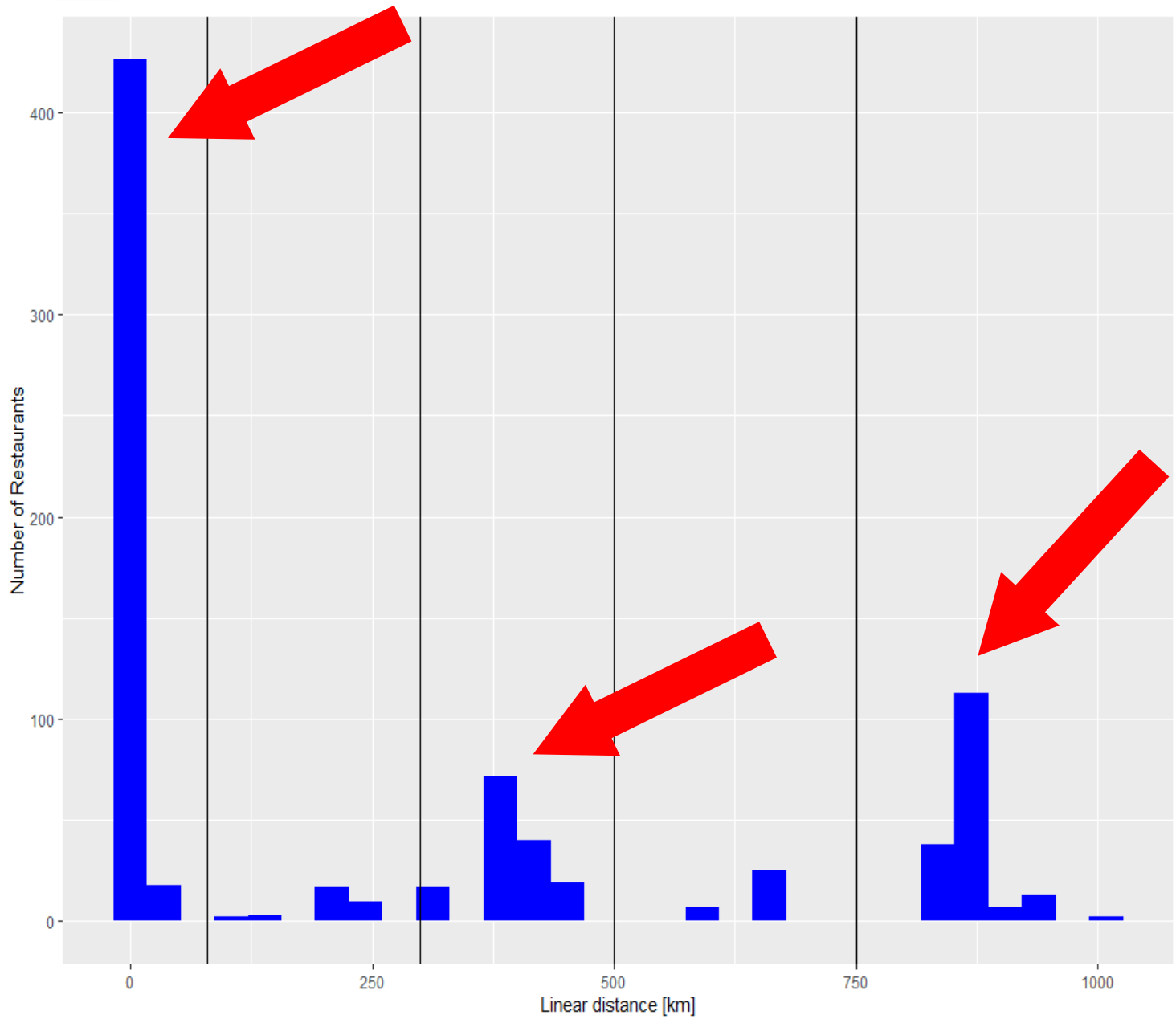
The difference between reserved visitors and non-reserved visitors (visitors) on each day of the week (shown by boxplot)



We find:

- We find that **there is no significant difference**
- although we see a slight trend for **more visitors compared to reservations on the weekend**, especially Sunday.

Data Visualization



The distance from the busiest area

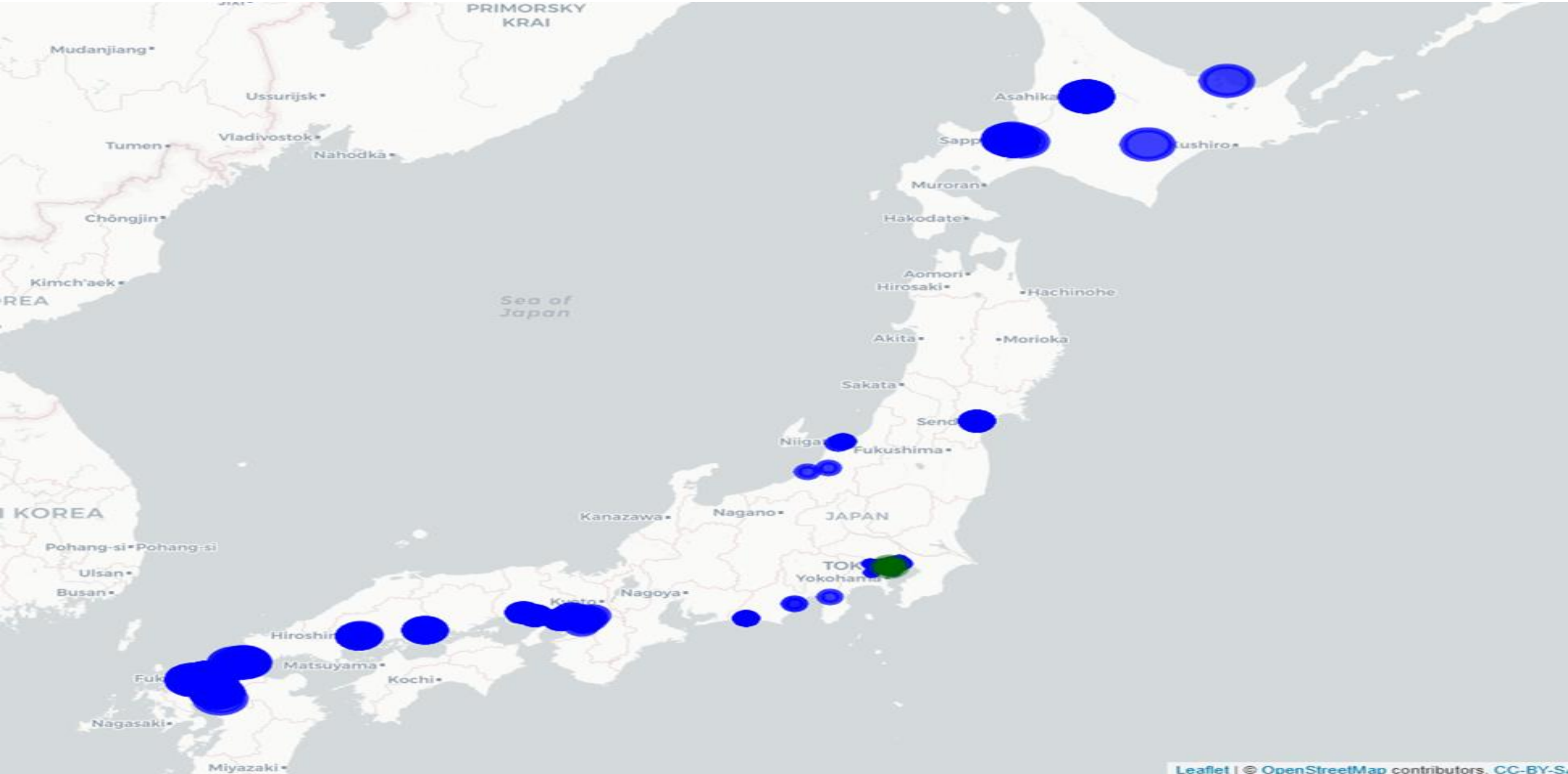
As the single **reference** point for all distances, we choose **the median latitude and median longitude**.

We find:

- The distance distribution has **a clear peak at very short values**; which is to be expected since our reference is the median.
- We also see **further peaks**, most prominently around ~450 km and ~800 km. These will **correspond to (groups of) cities**.

Data Visualization

The distribution of Restaurants (After Clustering)



PART 3:

Time Series Analysis

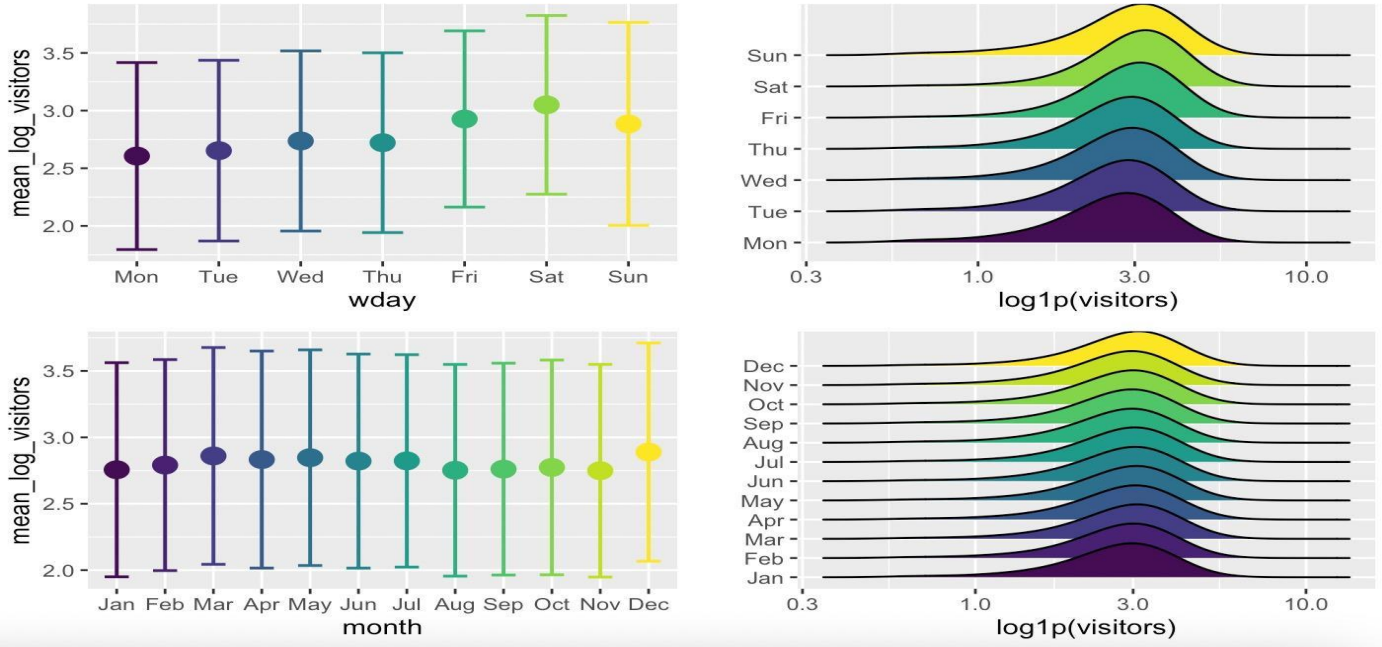
Time Series Model

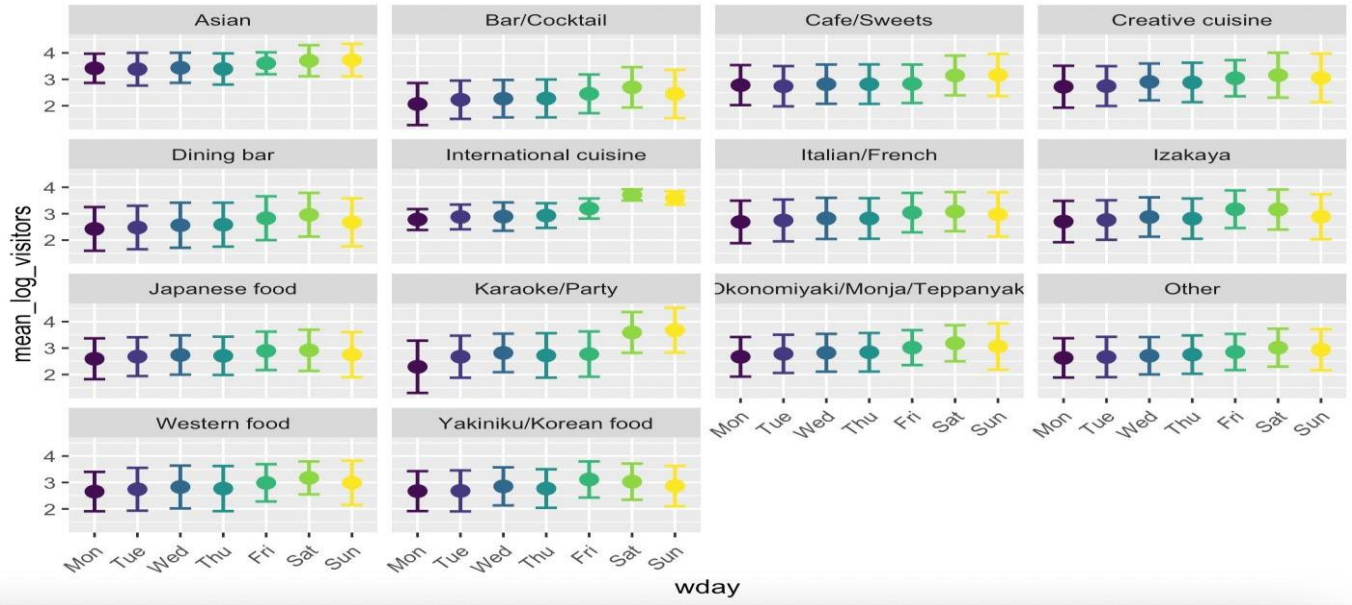
ARIMA is actually a class of models that explains a given time series based on its own past values, so that equation can be used to forecast future values. Any ‘non-seasonal’ time series that exhibits patterns and is not a random white noise can be modeled with ARIMA models.

To alleviate this supply gap and to make scalable forecasting dramatically easier, the Core Data Science team at Facebook created Prophet, a forecasting library for Python and R, which they open-sourced in 2017.

This method uses exponential smoothing to combine current information and past information. The weight of the old and new information is controlled by an adjustable parameter.

Feature Engineering





Deriving new features from the existing ones can provide additional predictive power for our goal to forecast visitor numbers.

Days of the week & months of the year

Restaurant per area - air_count

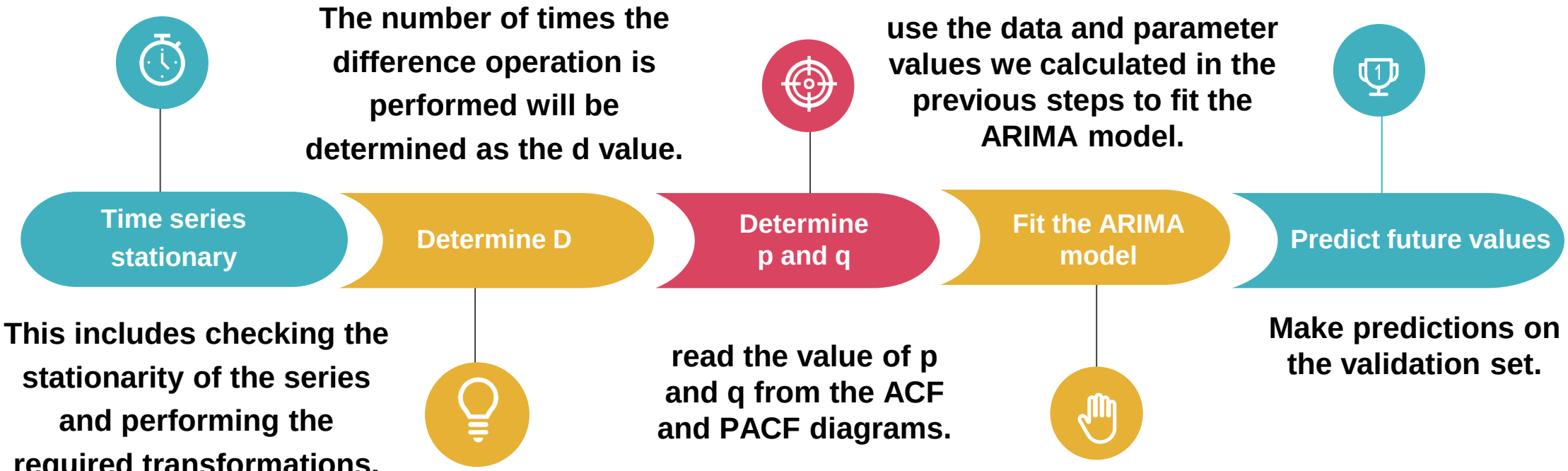
Distance from the busiest area

Prefectures

Time series parameter: mean, standard deviation, slope, and slope error of the time series

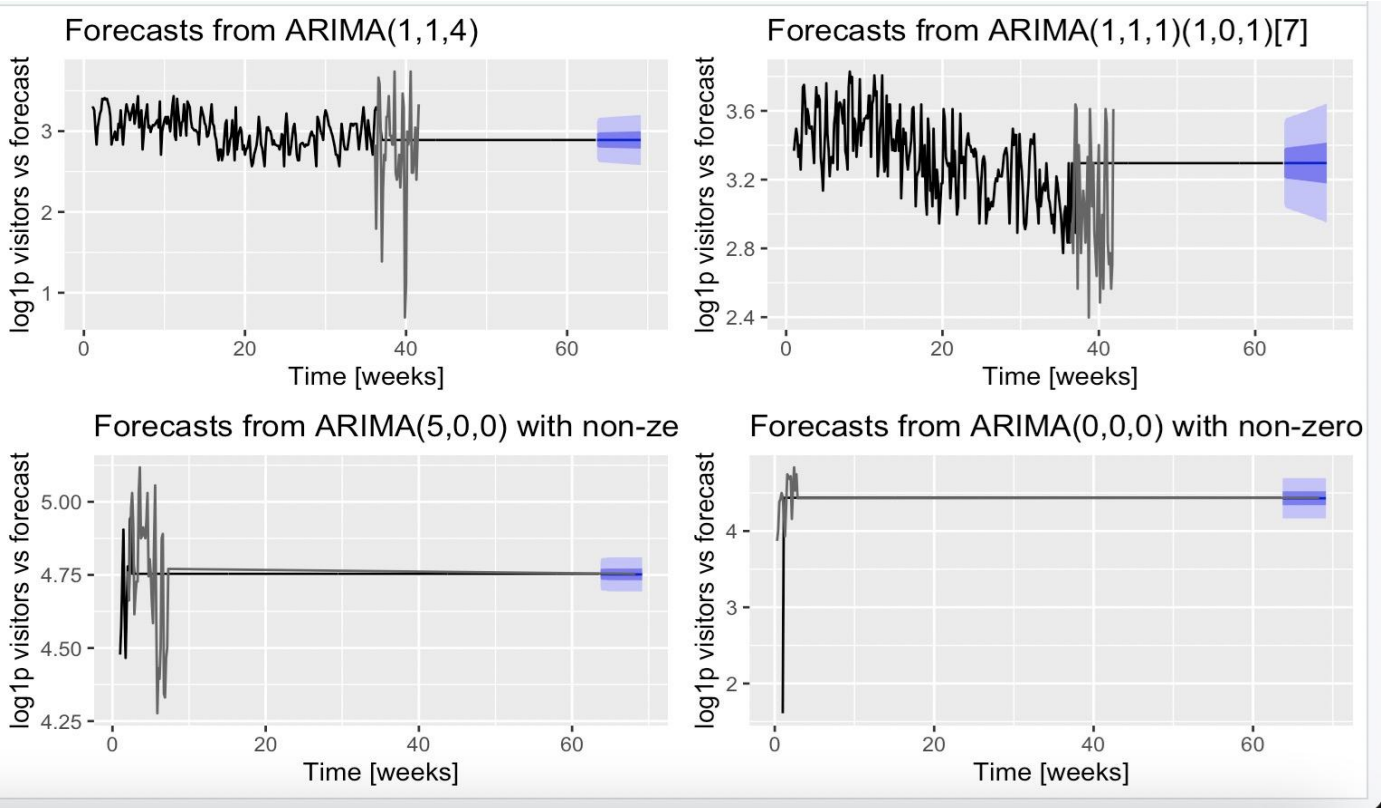
Model Demonstration

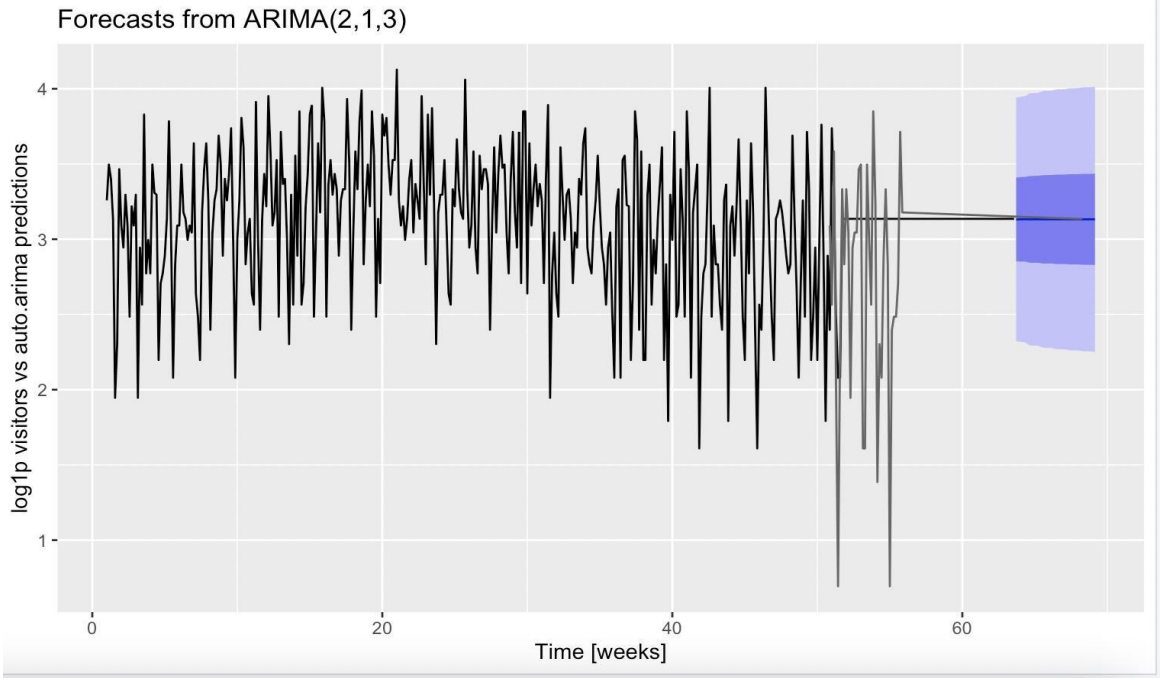
ARIMA (p, q ,d)



Model Demonstration

Result of ARIMA (p, q ,d)





Overall, the results are not great, but given that it's a fully automatic forecast (assuming only weekly periodicities) the auto. Arima tool gives us a first baseline to compare other methods to.

Model Demonstration

Prophet

Building on work based on ARIMA section. Creating a training and validation set for the same periods as above but not replace NA value.

Prophet expects a data frame with two columns: ds for the dates and y for the time series variable.

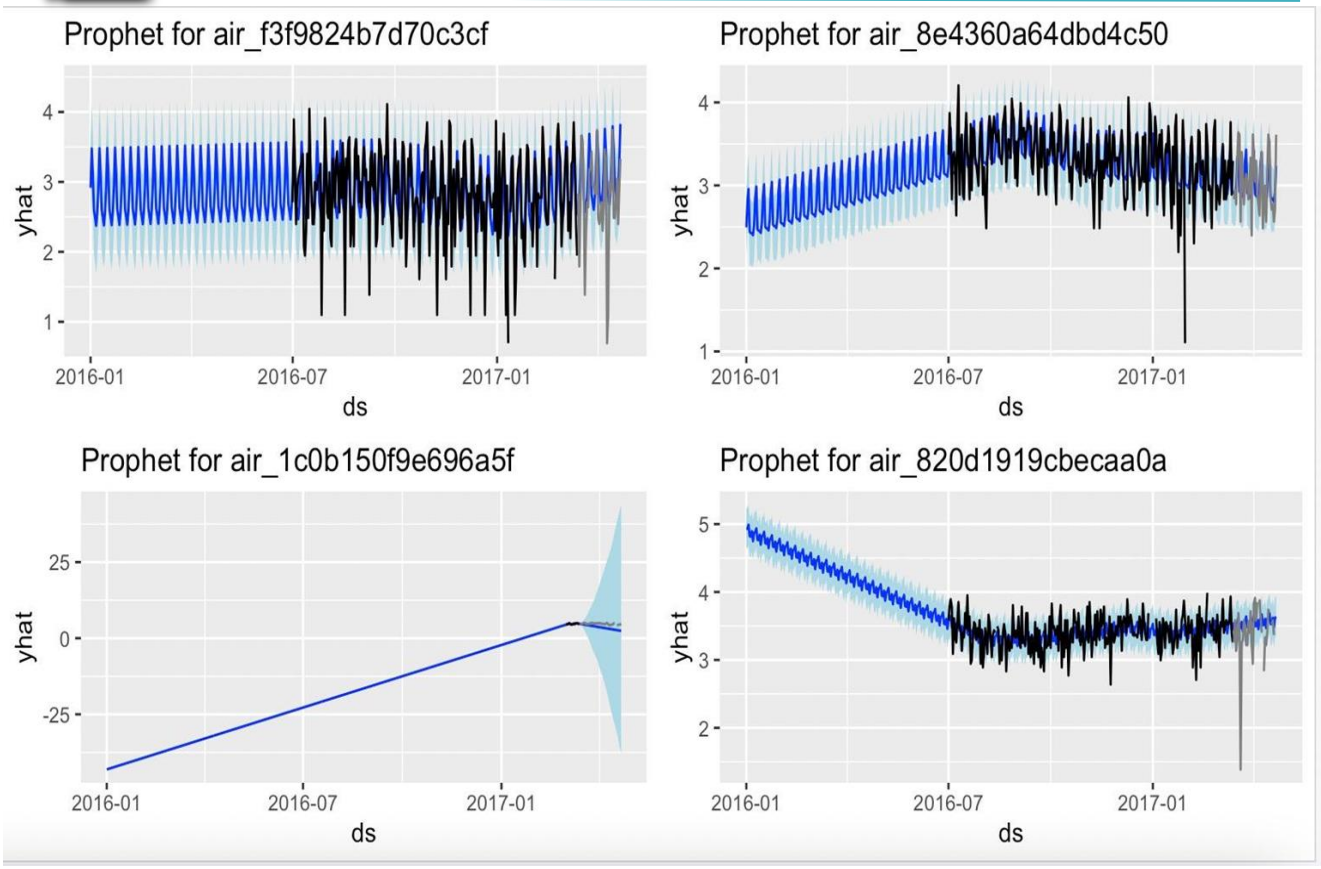
The parameter changepoint-prior-scale adjusts the trend flexibility. The parameter yearly. Seasonality has to be enabled/disabled explicitly and allows prophet to notice large-scale cycles. We have barely a year of data here.

Prophet utilizes an additive regression model which decomposes a time series into:

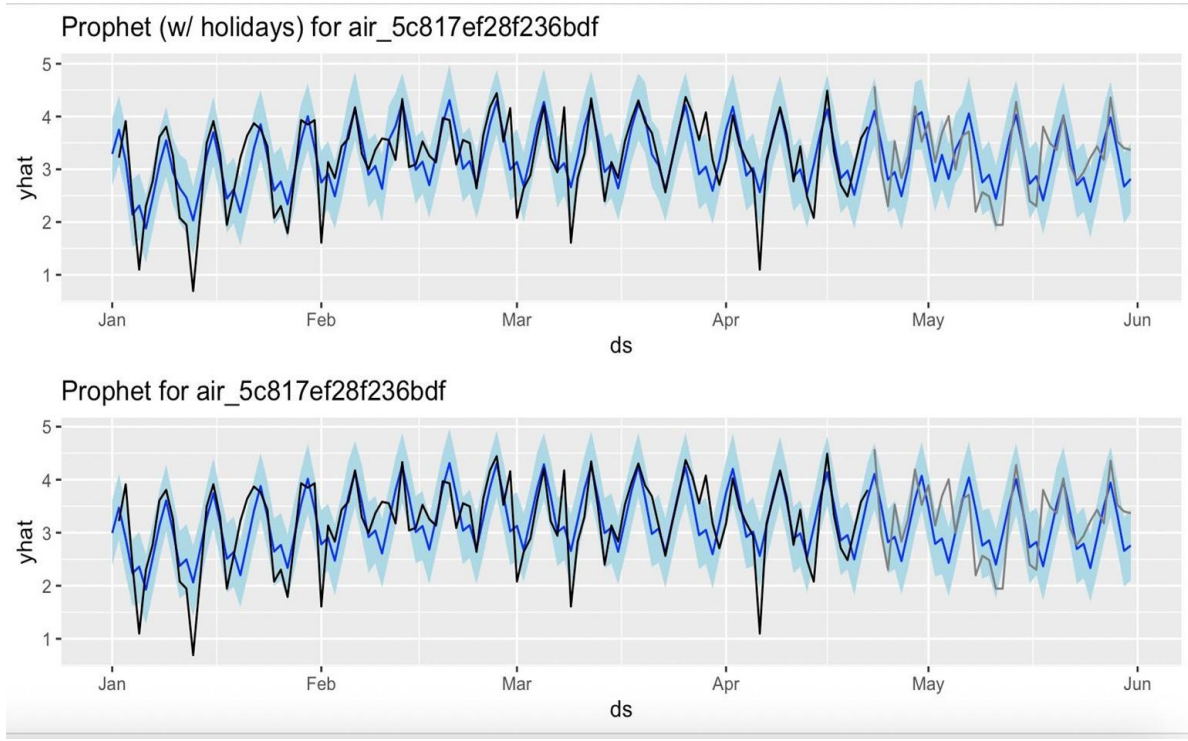
(i) a (piecewise) linear/logistic trend,
(ii) a yearly seasonal component,
(iii) a weekly seasonal component
(iv) an optional list of important days (such as holidays)

It claims to be “robust to missing data, shifts in the trend, and large outliers”. Especially the missing data functionality could be useful in this competition.

Model Demonstration



Result of Prophet



Plotting training visitor counts in black, the validation set in grey, and the forecast plus uncertainties in blue and light blue.

Finding that overestimating the trend component could probably result in less flexibility. The forth time series has too little data for prophet to do much. Here we could discard the trend component completely or simply take the median.

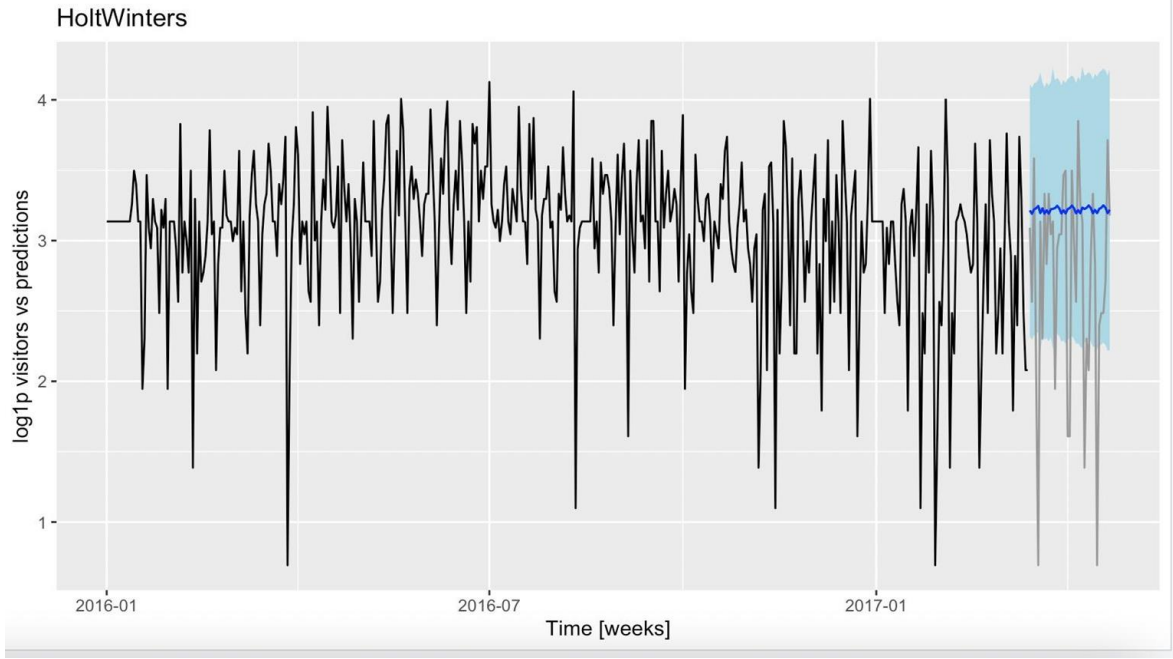
Prophet also gives us the possibility to include holidays and other special events in our forecast, which is very useful in taking the Golden Week into account.

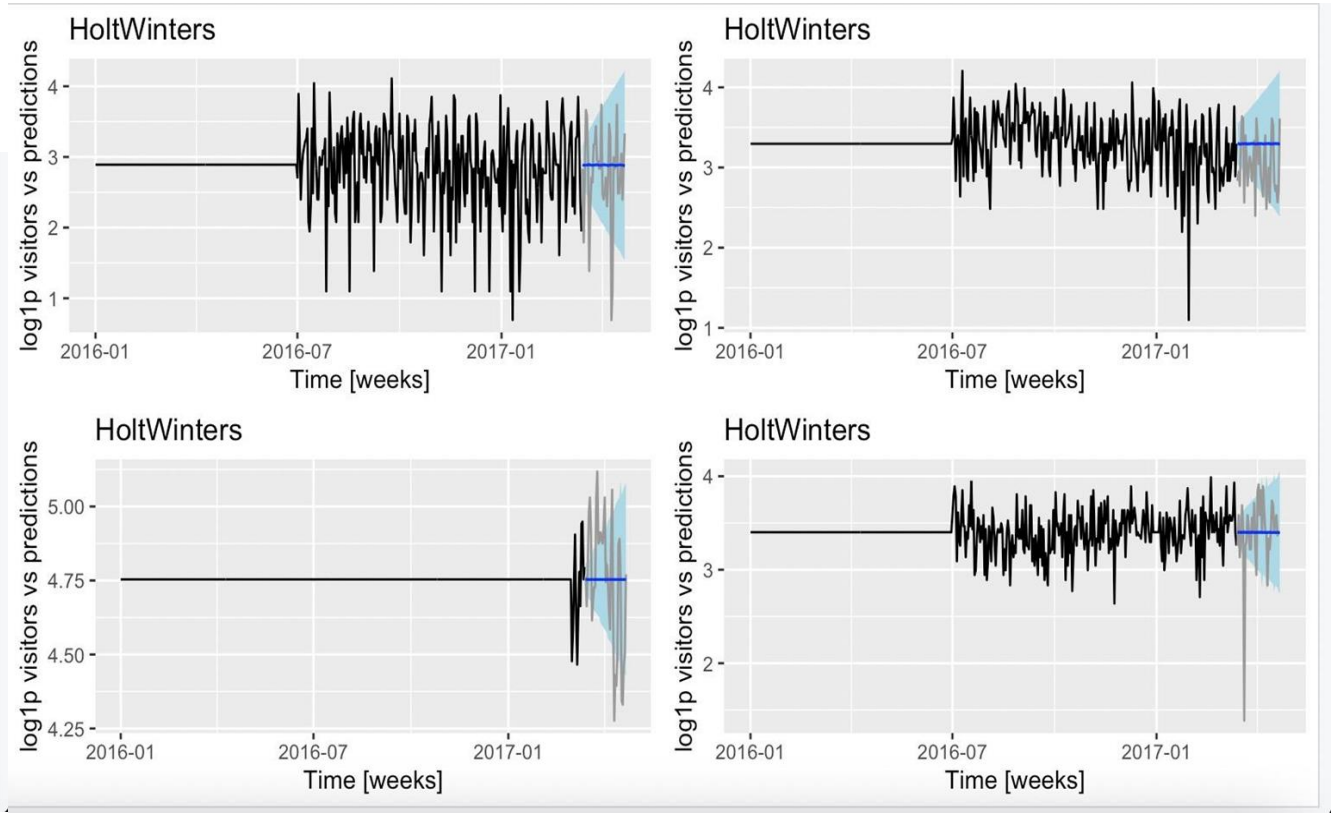
We may see that there is a subtle improvement in fitting the Golden Week visitors when including holidays. The performance of this component might improve if there are more holidays included in the training set.

Model Demonstration

Holt Winters

This is an exponential smoothing method which uses moving averages to take into account the presence of a trend in the data.





We might get a better performance by discarding the missing values instead of replacing them with the median. the first impression looks less promising than the prophet approach.

THANK YOU FOR WATCHING!