Citi Bike Jersey City

Aline de Almeida Ribeiro

November, 2018

Abstract

The main goal of this project is to identify patterns and trends in the way bikers behave while using Citi Bike Jersey City services. This project also details the creation of an algorithm that uses weather and individual trip data to predict the total number of bike journeys on a given date and hour in Jersey City, through the bike share service.

My Clients

My client is Citibank and their sponsored bikeshare service called Citi Bike, available in two North American cities: New York and Jersey City. I will focus my analysis on the trips made by users in Jersey City exclusively during the three first years of operations. Citi Bike is actually owned and operated by Motivate, the manager of several bike share systems in the world, which makes Motivate a potential client as well.

City Bike share system allows users to pick up a bike at any location and return it to any other station of the network. Anyone 16 or older can either become an annual member or buy a short-term pass (24-hour or 3-day). All they have to do is find an available bike nearby through the City Bike app, unlock it with a ride code or member key, take as many time-limited rides as they want and then return the bike to a docking station.

Project Proposal

Here are some of the questions I will try to answer with my analysis:

- Can we predict the total number of trips starting at a given date and time?
- Which journeys are most popular?
- Which start station is most frequent? Does it change during the day?
- Is it possible to identify any seasonality on this data? For example: what days of the week, hours of the day and weeks of the year are most rides taken on?
- What are the differences between trips made on weekdays and weekends?

Citi Bikes could use this analysis to have insights on how to run a more efficient and profitable service, as well as to identify business opportunities, such as: deciding in each region they should change the quantity of bikes available; when and where to relocate a docking station; what their frequent clientes are like; how users prefer to pay for this service; what is the most adequate moment to schedule maintenance; forecast the need of replacing a bike; measuring the program's success. Being able to predict the number of trips in a certain time and date could be used to help the service provider anticipate the demand for their service.

Original Data

The data consists of 36 csv files, one for each operating month. They have been made available by Citi Bikes and can be downloaded here (https://s3.amazonaws.com/tripdata/index.html).

I am going to analyze data from September 2015 until August 2018, compressing all trips made in Jersey City since the launch of the service in this region.

The overall data set includes over 860,000 observations and 15 variables, as follows:

- Trip Duration in seconds
- Start Time and Date
- Stop Time and Date
- Start Station Name
- End Station Name
- Start and End Station IDs
- Station Lat/Long
- Bike ID
- User Type
- Gender
- Year of Birth

It is important to highlight that Citi Bikes have processed the data to remove trips that had been taken by their staff to test their system and services. They have also deleted any trips under 60 seconds in duration.

Initial Data Wrangling

After downloading and unzipping all 36 csv files, they were read into R Studio individually using read_csv().

```
library(readr)
jc201509 <- read_csv("JC-201509-citibike-tripdata.csv")
jc201510 <- read_csv("JC-201510-citibike-tripdata.csv")
jc201511 <- read_csv("JC-201511-citibike-tripdata.csv")
jc201512 <- read_csv("JC-201512-citibike-tripdata.csv")
#...
```

I tried binding all data sets with bind_rows(), but the names of the columns were different in some of the csv files, which caused some columns to repeat themselves and produced some NA values. Also, not all values in the variable 'birth year' were integers, which was stopping me from successfully binding all files. The solution was to change 'birth year' from character into integer on some of the data sets and then rename all columns in all datasets at once. To rename the columns in all data sets, as a first step, I created a vector with definite column names, using colnames(). Then, I created a list containing all individual data sets.

Finally I used lapply(), to apply colnames() over dfList, and list2env(), to make sure all individual datasets would be available on my Global Environment, in case I needed to check any information from a specific month.

```
list2env(lapply(dfList, setNames, colnames), .GlobalEnv)
```

```
## <environment: R_GlobalEnv>
```

I was then finally able to bind all 36 data sets using bind_rows().

As I knew great part of my workload would be related to time series analysis, I decided to extract some extra information from my variable start_time, a POSIXct object. I used some functions from the lubridate package to extract month, year, day of the week,week of the year and hour of the day. All these items were put in separated columns, so that I could have easy access to them.

```
library(lubridate)
df$month <- month(df$start_time)
df$year <- year(df$start_time)
df$year_month <- format(df$start_time, format = "%Y-%m")
df$day_of_week <- wday(df$start_time, label = TRUE)
df$hour_of_day <- hour(df$start_time)
df$week_of_year <- week(df$start_time)
df$day_of_month <- day(df$start_time)
```

The trip_duration variable was in seconds. As it is not very intuitive to work with this unit of measurement, I used the mutate() function from the dplyr package to transform seconds into minutes, by simply dividing values by 60 and storing them in a new column, duration_minutes.

```
df <- mutate(df, duration_minutes = round(trip_duration / 60))
```

Now I am finally ready to check into the data set's structure and investigated for NA and values that don't make sense.

```
glimpse(df)
```

```
## Observations: 826,012
## Variables: 23
## $ trip_duration           <int> 61, 290, 786, 477, 451, 401, 1215, 114...
## $ start_time              <dttm> 2015-09-21 14:53:16, 2015-09-21 14:55...
## $ stop_time               <dttm> 2015-09-21 14:54:17, 2015-09-21 15:00...
## $ start_station_id        <int> 3185, 3183, 3183, 3203, 3203, 3195, 31...
## $ start_station_name      <chr> "City Hall", "Exchange Place", "Exchan...
## $ start_station_latitude  <dbl> 40.71773, 40.71625, 40.71625, 40.72760...
## $ start_station_longitude <dbl> -74.04385, -74.03346, -74.03346, -74.0...
## $ end_station_id          <int> 3185, 3187, 3183, 3203, 3186, 3195, 32...
## $ end_station_name        <chr> "City Hall", "Warren St", "Exchange Pl...
## $ end_station_latitude    <dbl> 40.71773, 40.72112, 40.71625, 40.72760...
## $ end_station_longitude   <dbl> -74.04385, -74.03805, -74.03346, -74.0...
## $ bike_id                 <int> 24722, 24388, 24442, 24678, 24574, 245...
## $ user_type               <chr> "Subscriber", "Customer", "Subscriber"...
## $ birth_year              <int> 1975, NA, 1962, 1977, 1977, 1987, 1964...
## $ gender                  <int> 1, 0, 1, 2, 2, 2, 1, 1, 1, 2, 1, 1, 1,...
## $ month                   <dbl> 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,...
## $ year                    <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 20...
## $ year_month              <chr> "2015-09", "2015-09", "2015-09", "2015...
## $ day_of_week             <ord> Mon, Mon, Mon, Mon, Mon, Mon, Mon, Mon...
## $ hour_of_day             <int> 14, 14, 14, 14, 14, 15, 15, 15, 15, 15...
## $ week_of_year            <dbl> 38, 38, 38, 38, 38, 38, 38, 38, 38, 38...
## $ day_of_month            <int> 21, 21, 21, 21, 21, 21, 21, 21, 21, 21...
## $ duration_minutes        <dbl> 1, 5, 13, 8, 8, 7, 20, 19, 33, 3, 11, ...
```

Generally speaking, there is not much strange data and NA values, except for two variables: birth_year and duration_minutes (trip_duration).

Most of the data was generated automatically as users rode their bikes. Birth year is the only information in this data set users were requested to inform Citi Bikes. And that's why there are some NA values (5.35% of the data) and some others don't make much sense. As much as I would like to believe elderly people have been exercising a lot in New Jersey, it seems a bit unrealistic that too many people over 80 years of age would be riding a Citi bike. So I decided to replace all birth years prior to 1938 with NA values.

```
df$birth_year <- replace(df$birth_year, df$birth_year < 1938, NA)
summary(df$birth_year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##    1939    1974    1982    1980    1987    2002   44843
```

Regarding the duration_minutes variable, there are a few trips which lasted many hours, even days. That does not make sense since Citi Bikes charges their bikers per amount of time spent riding, which means that very long rides would be way too expensive. As we go ahead with the analysis, I will present a plot with the duration_minutes distribution and these erroneous data will be further analyzed. For now, I will keep this data in the data set.

```
summary(df$duration_minutes)
```

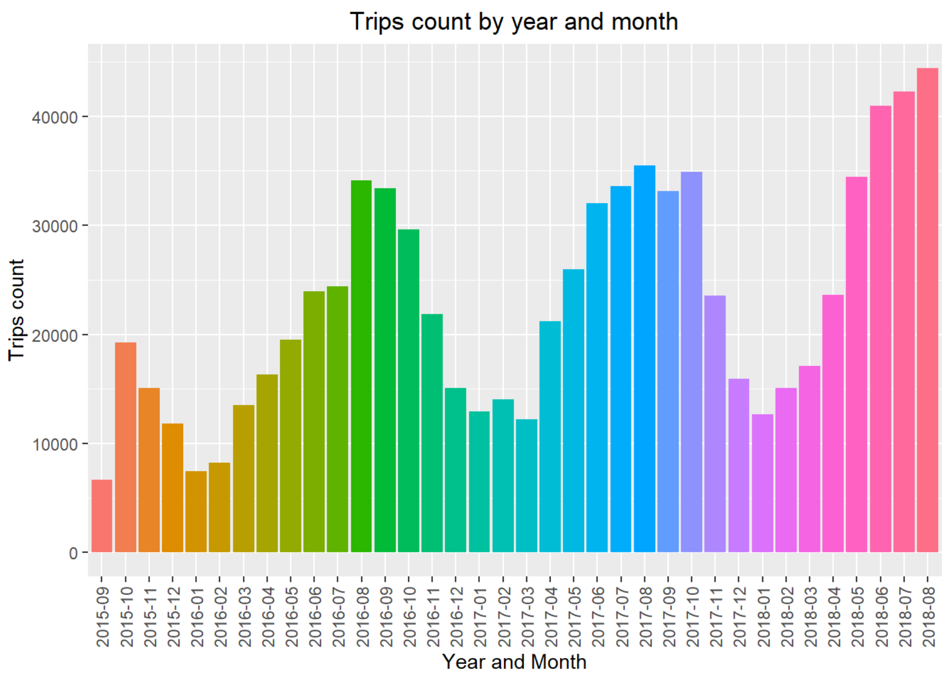
```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
##      1.0      4.0      6.0     13.7     10.0 337670.0
```

Initial Exploratory Data Analysis

Seasonality

Considering the first three years of operations, there were 826,012 rides, since day one, with a monthly distribution as follows:

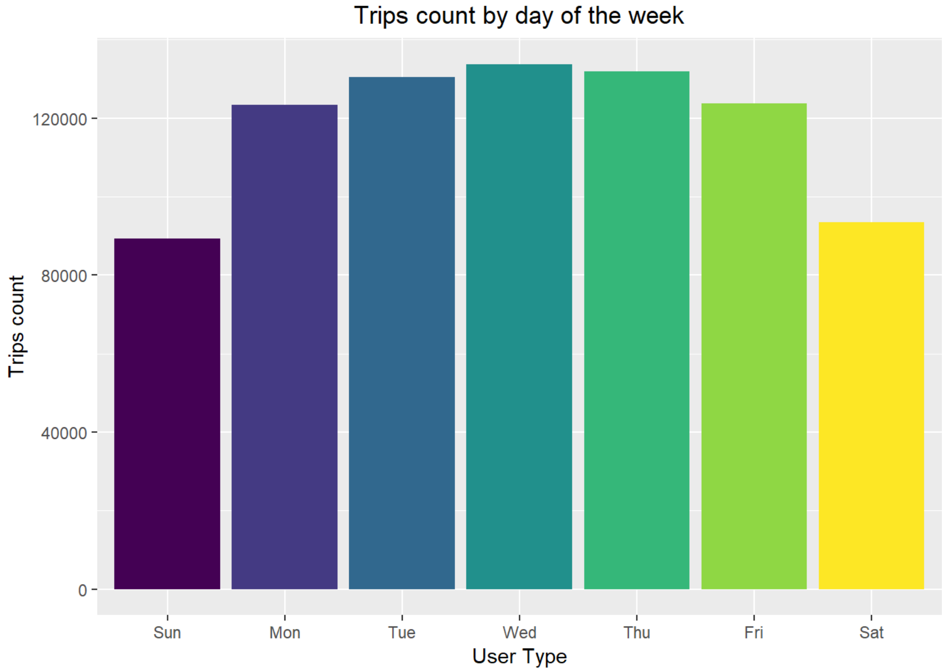
```
library(ggplot2)
trips_monthly_p <- ggplot(df, aes(x = year_month)) +
  geom_bar(aes(fill=year_month))+
  theme(axis.text.x=element_text(angle=90,hjust=1,vjust=0.5),
        legend.position="none", plot.title = element_text(hjust = 0.5))+
  scale_y_continuous("Trips count")+
  scale_x_discrete("Year and Month")+
  ggtitle("Trips count by year and month")


trips_monthly_p
```



As expected, the month of the year has a big impact on the amount of trips: more trips are taken on warmer months than during winter and fall. This piece of information is the first cue that month might be a strong independent variable for building our algorithm.

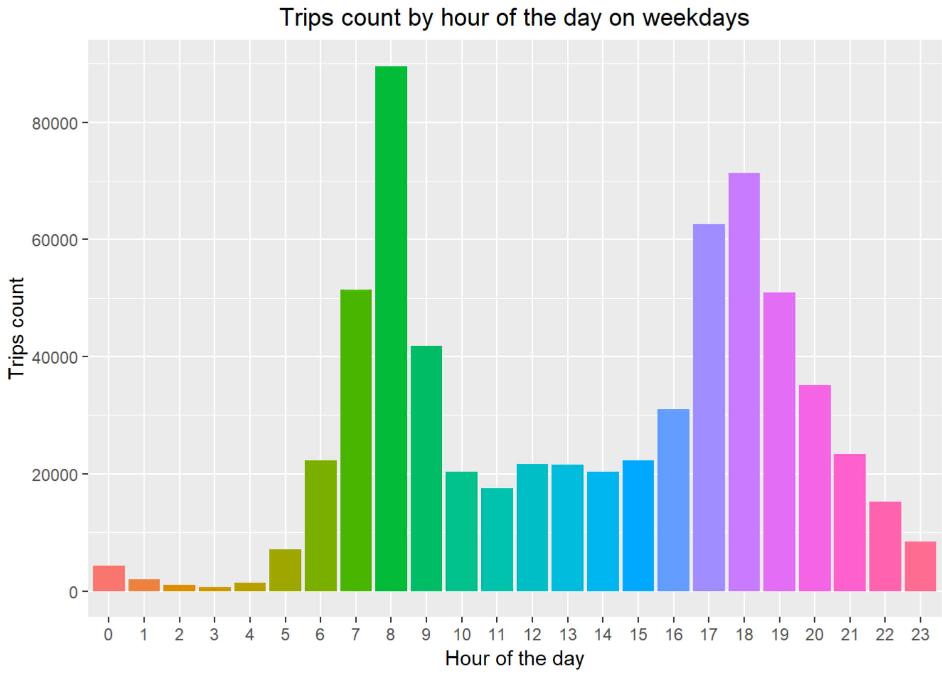
Another expected fact is that weekdays are busier than weekends, with the busiest day being Wednesday, and the smallest quantity of trips being on Sunday.

```
trips_daily_p <- ggplot(df, aes(x = day_of_week)) +
  geom_bar(aes(fill = day_of_week)) +
  theme(legend.position="none", plot.title = element_text(hjust = 0.5))+
  scale_y_continuous("Trips count", breaks = seq(0, 200000, by = 40000)) +
  scale_x_discrete("User Type")+
  ggtitle("Trips count by day of the week")

trips_daily_p
```



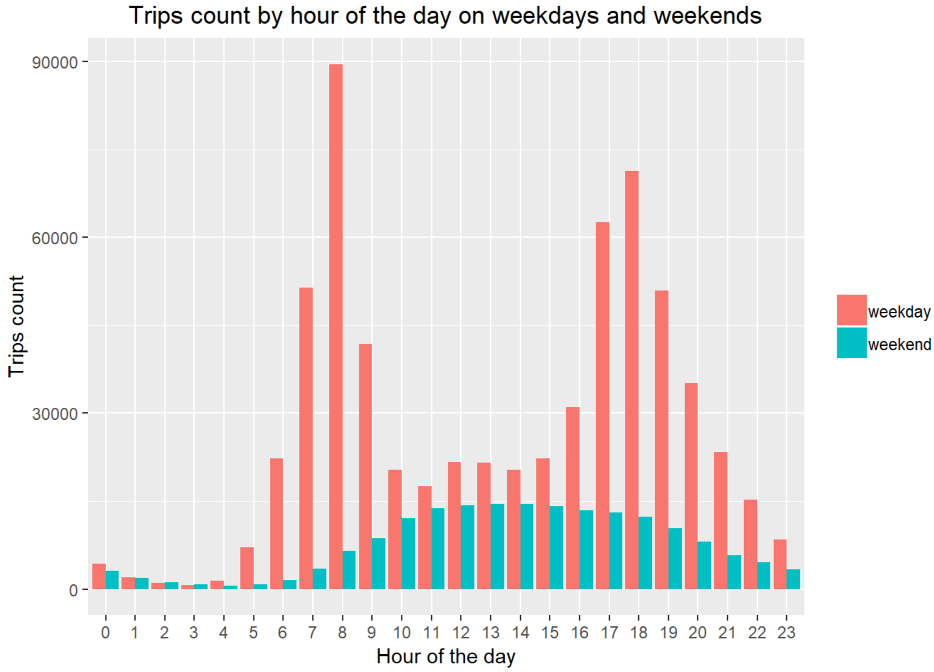
On weekdays, most of the trips are taken in two different moments of the day: between 7 and 9 am, and between 5 and 7 pm, which is also very predictable, as it is the most usual hours people leave/arrive home from either work or school.

```
trips_weekdays_plot <- ggplot(trips_weekdays, aes(x = factor(hour_of_day))) +
  geom_bar(aes(fill = factor(hour_of_day))) +
  theme(legend.position="none", plot.title = element_text(hjust = 0.5))+
  scale_y_continuous("Trips count", breaks = seq(0, 130000, by = 20000)) +
  scale_x_discrete("Hour of the day")+
  ggtitle("Trips count by hour of the day on weekdays")


trips_weekdays_plot
```



On the other hand, on weekends, trips are more evenly spread throughout the day, with higher concentration between 10 am and 7 pm. The difference between these two distributions and the importance of time of the day for the variation of trips quantity are very clear on the following plot:

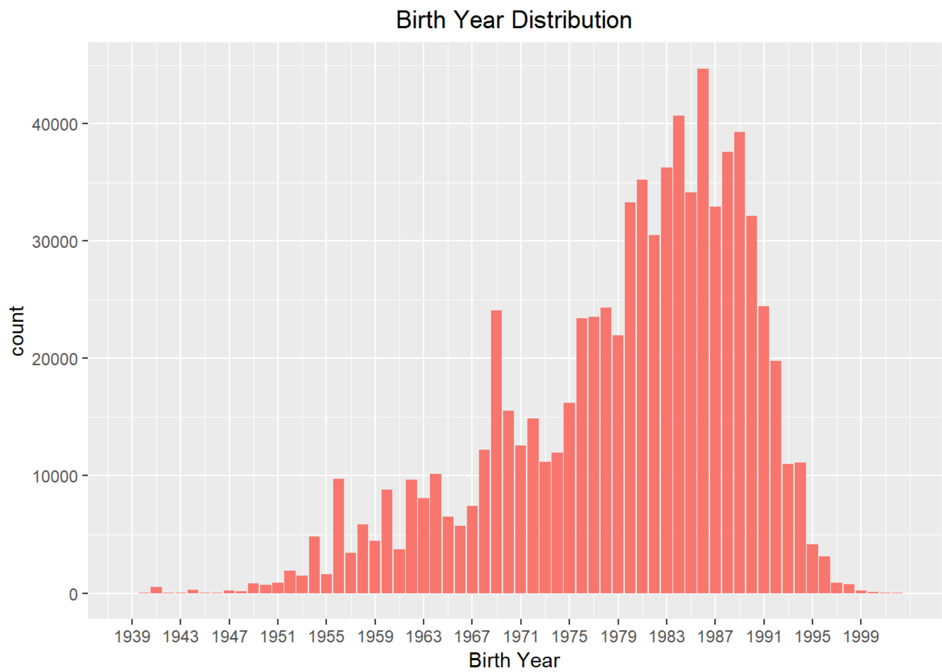
```
wkday_wkend_p <- ggplot(df, aes(x = factor(hour_of_day), fill = week)) +
  geom_bar(position = "dodge" ) +
  theme(legend.title=element_blank(), plot.title = element_text(hjust = 0.5))+
  scale_y_continuous("Trips count", breaks = seq(0, 1200000, by = 30000)) +
  scale_x_discrete("Hour of the day")+
  ggtitle("Trips count by hour of the day on weekdays and weekends")

wkday_wkend_p
```



Demographics

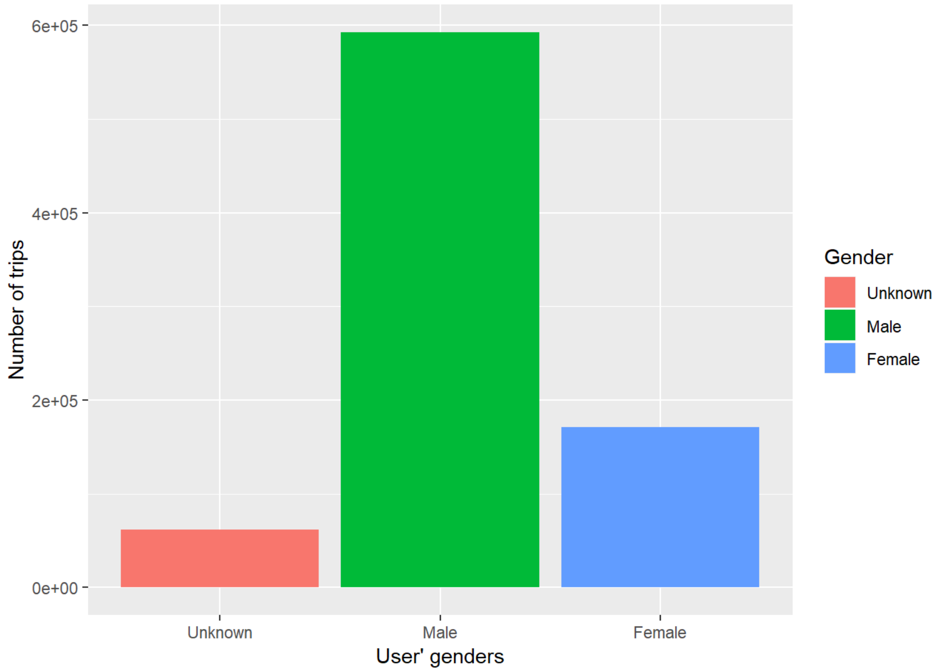
Most users were born in the 80s, following by those born in the lates 70s and early 90s. See full distribution below. It is important to highlight that one has to be 16 years or older to ride a Citi bike.

```
birth_year_p <- ggplot(df, aes(x = birth_year))+
  geom_bar(aes(fill = "birth_year"))+
  scale_x_continuous("Birth Year", breaks = seq(1939, 2002, by = 4) )+
  ggtitle("Birth Year Distribution")+
  theme(legend.position = "none", plot.title = element_text(hjust = 0.5))

birth_year_p
```



The bar plot below tells us that most of the trips were made by men (592.683), while only 171.404 trips were taken by women.

```
gender_bar <- ggplot(df, aes(x = factor(gender), fill = factor(gender))) +
  geom_bar() +
  scale_x_discrete("User' genders", breaks=c("0", "1", "2"),labels=c("Unknown", "Male", "Female")) +
  scale_fill_discrete("Gender", breaks=c("0", "1", "2"),labels=c("Unknown", "Male", "Female")) +
  scale_y_continuous("Number of trips")

gender_bar
```



User type

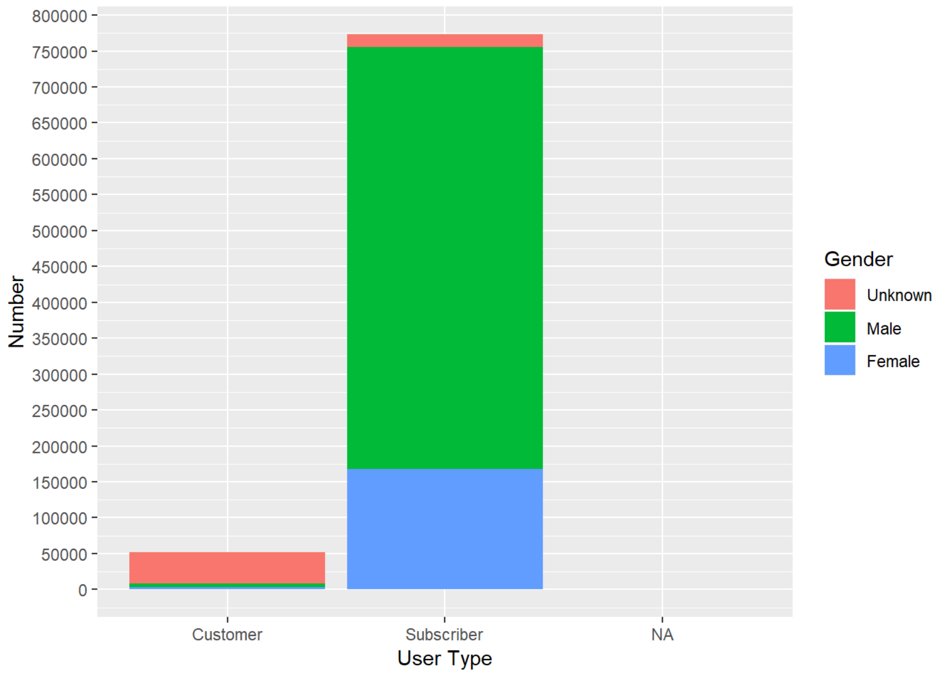
There are two user types:

Customer, that is, a person who bought either a 24-hour pass, for 12 dollars, or 3-day pass, for 24 dollars. These passes include unlimited 30-minute rides. If a customer keeps a bike for more than 30 minutes, he/she will have to pay an extra fee of 4 dollars per additional 15 minutes.

Subscriber, a client who has signed up for an annual membership, for 169 dollars. Annual members can go on unlimited 45-minute rides. For longer rides, they are charged 2.50 dollars per additional 15 minutes.

As Jersey City is not a tourist spot, like NY City, it is not surprising that most of the rides are made by subscribers, and not eventual users.

```
user_gender_stack <- ggplot(df, aes(x = user_type, fill = factor(gender))) +
  geom_bar(position = "stack" ) +
  scale_fill_discrete("Gender", breaks=c("0", "1", "2"),labels=c("Unknown", "Male", "Female")) +
  scale_y_continuous("Number", breaks = seq(0, 900000, by = 50000)) +
  scale_x_discrete("User Type")

user_gender_stack
```



Trips

There are 50 stations serving Jersey Bikes across Jersey city.

These are the top 10 most frequent start stations and the related number of trips:

```
origin <- df %>%  group_by(start_station_name) %>% count() %>%  arrange(desc(n))
origin
```

```
## # A tibble: 61 x 2
## # Groups:   start_station_name [61]
##    start_station_name     n
##    <chr>              <int>
##  1 Grove St PATH      96643
##  2 Exchange Place     54957
##  3 Hamilton Park      52890
##  4 Sip Ave            48134
##  5 Newport PATH       39888
##  6 Newark Ave         27059
##  7 Newport Pkwy       26945
##  8 Van Vorst Park     26235
##  9 Warren St          26099
## 10 Morris Canal       25575
## # ... with 51 more rows
```

These are the top 10 most frequent end stations and the related number of trips:

```
destination <- df %>%  group_by(end_station_name) %>%  count %>%  arrange(desc(n))
destination
```

```
## # A tibble: 203 x 2
## # Groups:   end_station_name [203]
##    end_station_name      n
##    <chr>             <int>
##  1 Grove St PATH    123218
##  2 Exchange Place    66770
##  3 Hamilton Park     50201
##  4 Sip Ave           43890
##  5 Newport PATH      41805
##  6 Newport Pkwy      26645
##  7 Warren St         25813
##  8 Essex Light Rail  25699
##  9 Newark Ave        25051
## 10 City Hall         24948
## # ... with 193 more rows
```

These are the most popular routes:

```
combination<- df %>% group_by(start_station_name, end_station_name) %>% tally() %>%  arrange(desc(n))
combination
```

```
## # A tibble: 3,464 x 3
## # Groups:   start_station_name [61]
##    start_station_name end_station_name     n
##    <chr>              <chr>            <int>
##  1 Hamilton Park      Grove St PATH    20087
##  2 Grove St PATH      Hamilton Park    14273
##  3 Brunswick St       Grove St PATH    13510
##  4 Morris Canal       Exchange Place   11525
##  5 McGinley Square    Sip Ave          10898
##  6 Van Vorst Park     Grove St PATH    10401
##  7 Sip Ave            McGinley Square   9366
##  8 Grove St PATH      Brunswick St      9202
##  9 Exchange Place     Morris Canal      8653
## 10 Jersey & 6th St    Grove St PATH     8195
## # ... with 3,454 more rows
```

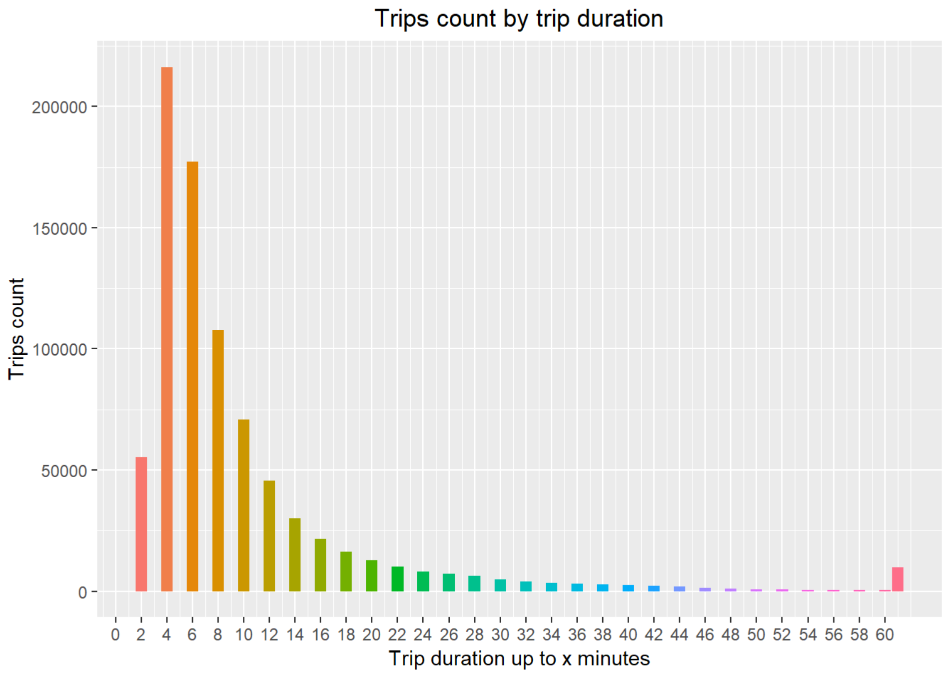
And these are the most popular routes by time of the day:

```
combination_hour<- df %>% group_by(start_station_name, end_station_name, hour_of_day) %>% tally() %>%  arrange(desc(n))
combination_hour
```

```
## # A tibble: 36,014 x 4
## # Groups:   start_station_name, end_station_name [3,464]
##    start_station_name end_station_name hour_of_day     n
##    <chr>              <chr>                  <int> <int>
##  1 Hamilton Park      Grove St PATH              8  7081
##  2 Brunswick St       Grove St PATH              8  4433
##  3 Hamilton Park      Grove St PATH              7  4137
##  4 Grove St PATH      Hamilton Park             18  3578
##  5 Morris Canal       Exchange Place             8  3249
##  6 McGinley Square    Sip Ave                    8  3097
##  7 Grove St PATH      Hamilton Park             17  2552
##  8 Hamilton Park      Grove St PATH              9  2360
##  9 Van Vorst Park     Grove St PATH              8  2222
## 10 Essex Light Rail   Exchange Place             7  2116
## # ... with 36,004 more rows
```

The plot below shows us the distribution of trips according to their duration.

```
duration_p <- ggplot(df, aes(x = intervals)) +
  geom_bar(aes(fill = factor(intervals)))+
  theme(legend.position = "none",  plot.title = element_text(hjust = 0.5))+
  scale_y_continuous("Trips count", breaks = seq(0, 250000, 50000 )) +
  scale_x_continuous("Trip duration up to x minutes", breaks = seq(0, 60, by = 2))+
  ggtitle("Trips count by trip duration")


duration_p
```



Bike Ids

There have been 1564 different bike ids.

Bike ids that have made more trips are:

```
journeys_ind <- df %>% group_by(bike_id) %>% count() %>% arrange(desc(n))
journeys_ind
```

```
## # A tibble: 1,564 x 2
## # Groups:   bike_id [1,564]
##    bike_id     n
##      <int> <int>
##  1   26223  1483
##  2   26306  1473
##  3   26182  1464
##  4   26217  1461
##  5   26295  1459
##  6   26276  1458
##  7   26215  1446
##  8   26219  1438
##  9   26151  1436
## 10   26192  1433
## # ... with 1,554 more rows
```

The average number of trips made by each bike id is:

```
avg_trips <- mean(journeys_ind$n)
avg_trips
```

```
## [1] 528.1407
```

And this is the average life of a bike id:

```
bikeid_life <- df %>%  group_by(bike_id) %>%
  summarise(min_date = min(start_time), max_date = max(stop_time),
            life = min_date - max_date)

bikeid_life
```

```
## # A tibble: 1,564 x 4
##    bike_id min_date            max_date            life
##      <int> <dttm>              <dttm>              <time>
##  1   14529 2017-07-17 08:13:29 2017-07-18 07:47:16 -23.563056 hours
##  2   14552 2016-05-26 12:10:20 2016-05-27 16:01:10 " -1.160301 hours"
##  3   14556 2017-05-18 19:16:22 2017-05-30 17:54:50 -11.943380 hours
##  4   14632 2016-02-10 07:27:34 2016-02-10 07:30:37 " -3.050000 hours"
##  5   14705 2016-06-08 21:33:32 2016-06-09 12:18:20 -14.746667 hours
##  6   14716 2017-06-08 16:02:34 2017-06-09 08:19:48 -16.287222 hours
##  7   14717 2016-04-30 10:39:56 2016-05-01 07:33:53 -20.899167 hours
##  8   14786 2016-06-18 12:51:59 2016-06-18 17:03:53 " -4.198333 hours"
##  9   14863 2017-04-10 02:27:45 2017-05-02 18:10:26 -22.654641 hours
## 10   14872 2016-06-17 16:47:33 2016-06-17 18:04:50 " -1.288056 hours"
## # ... with 1,554 more rows
```

```
mean(bikeid_life$life)
```

```
## Time difference of -320.1042 hours
```

Continuing Data Wrangling

As my main goal is to create an algorithm that predicts the number of rides in a given hour, I have added Newark's airport weather records to this project to help me investigate how weather conditions might impact Citi Bike usage.

After importing the weather dataset and selecting the variables I would work with, I have noticed that the only common aspects between both datasets were date and time. So I used them to create a common identifier to merge the datasets, which was composed of "year month day hour".

```
weather_clean$id <- paste(weather_clean$year, weather_clean$month,
                          weather_clean$day_of_month, weather_clean$hour_of_day)
```

```
df$id <- paste(df$year, df$month, df$day_of_month, df$hour_of_day)
```

year	month	day_of_month	hour_of_day	id
2015	9	1	0	2015 9 1 0
2015	9	1	1	2015 9 1 1
2015	9	1	1	2015 9 1 1
2015	9	1	2	2015 9 1 2
2015	9	1	3	2015 9 1 3

```
df_merged <- left_join(df, weather_clean, by = "id")
```

Now the dataset contains 5 new variables: weather_type, temperature, humidity, wind_speed and precipitation.

Precipitation amounts are given in inches. Trace amounts of precipitation were indicated with a "T" on the original dataset, but they have been changed by into 0, as well as NA values. There were also some rare NA values in temperature and humidity. For such cases, the average value of the previous and following observations was computed.

The weather_type variable contained 128 different combinations of upper and low case letters, ponctuation and numbers. After consulting the data set guide to indetify each component's meaning - the guide can be consulted here (https://www1.ncdc.noaa.gov/pub/data/cdo/documentation/LCD_documentation.pdf) - I created a column for each weather type (drizzle, rain, snow, ice pellets and thunder) and I filled them in with booleans accordingly.

The last point to note is the creation of the column season and filled with 1 (winter), 2 (fall), 3 (spring), 4 (summer). Subsequently I created a column named holiday and I identified all federal and state holidays, as they would presumably have an effect on bike trips.

As I am trying to predict the number of trips per hour, variables that describe individual characteristics of specific trips or users will not be taken into consideration to build our model, such as start/stop station id, name, latitude, longitude, trip duration, gender, user_type and birth_year. So I have subseted the data frame and renamed it into df_merged2.

```
df_merged2 <- df_merged[c(1, 17, 18, 20:23, 25, 27:30, 32:38)]
```

To conclude, using df_merged2, I counted trips per hour and then added the output to column n in df_ml, the dataset will be used to build our model. As rows were duplicated, I removed repetitions with !duplicated(). This is the final dataset.

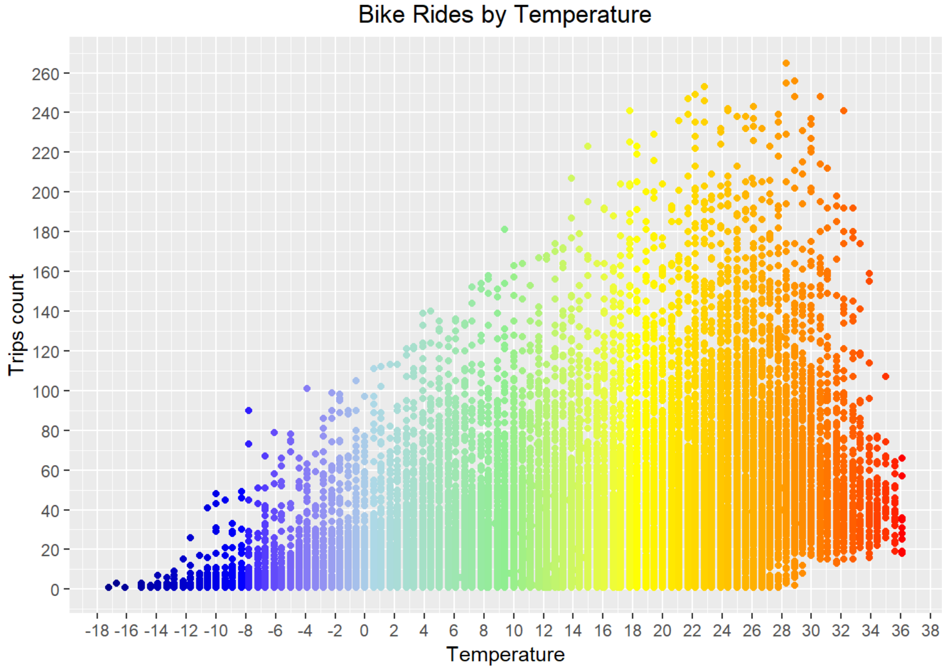
```
trips_hourly <- (df_merged2 %>% group_by(id) %>% count())

df_ml <- left_join(df_merged2, trips_hourly, by = "id")

df_ml <- df_ml[!duplicated(df_ml$id), ]
head(df_ml)
```

```
## # A tibble: 6 x 20
##   id    month  year day_of_week hour_of_day week_of_year day_of_month week
##   <chr> <dbl> <dbl> <ord>             <int>        <dbl>        <int> <chr>
## 1 2015~     9  2015 Mon                  14           38           21 week~
## 2 2015~     9  2015 Mon                  15           38           21 week~
## 3 2015~     9  2015 Mon                  16           38           21 week~
## 4 2015~     9  2015 Mon                  17           38           21 week~
## 5 2015~     9  2015 Mon                  18           38           21 week~
## 6 2015~     9  2015 Mon                  19           38           21 week~
## # ... with 12 more variables: temperature <dbl>, humidity <dbl>,
## #   wind_speed <int>, precipitation <chr>, drizzle <dbl>, rain <dbl>,
## #   snow <dbl>, ice_pellets <dbl>, thunder <dbl>, season <chr>,
## #   holiday <chr>, n <int>
```

Continuig Exploratory Data Analysis

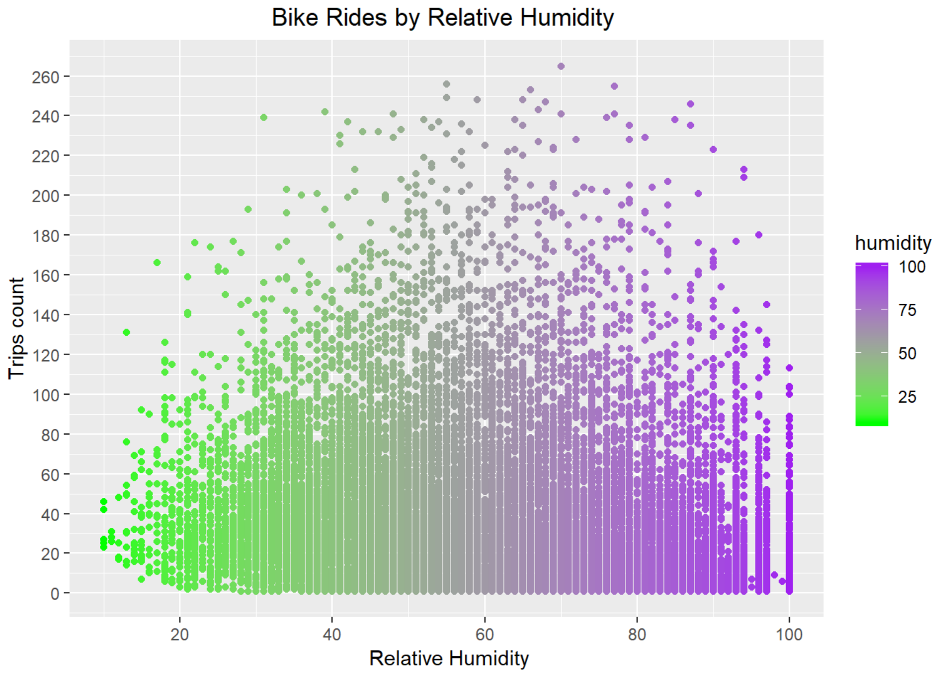
This scatterplot of temperature against bike trips count depicts that temperatures are positively correlated with the amount of trips, as we see that bike journey count increases as temperature increases up to 30ºC. Temperatures higher than that cause the count to decrease.

```
temp_count <- ggplot(df_ml, aes(x = temperature, y = n))+
              geom_point(aes(color = temperature))+
              theme(legend.position = "none", plot.title = element_text(hjust = 0.5))+
              scale_x_continuous("Temperature", breaks = seq(-20, 40, by = 2))+
              scale_y_continuous("Trips count", breaks = seq( 0, 340, by = 20))+
              ggtitle("Bike Rides by Temperature")+
              scale_color_gradientn(colors=c('dark blue','blue','light blue','light green','yellow','orange','red'))

temp_count
```



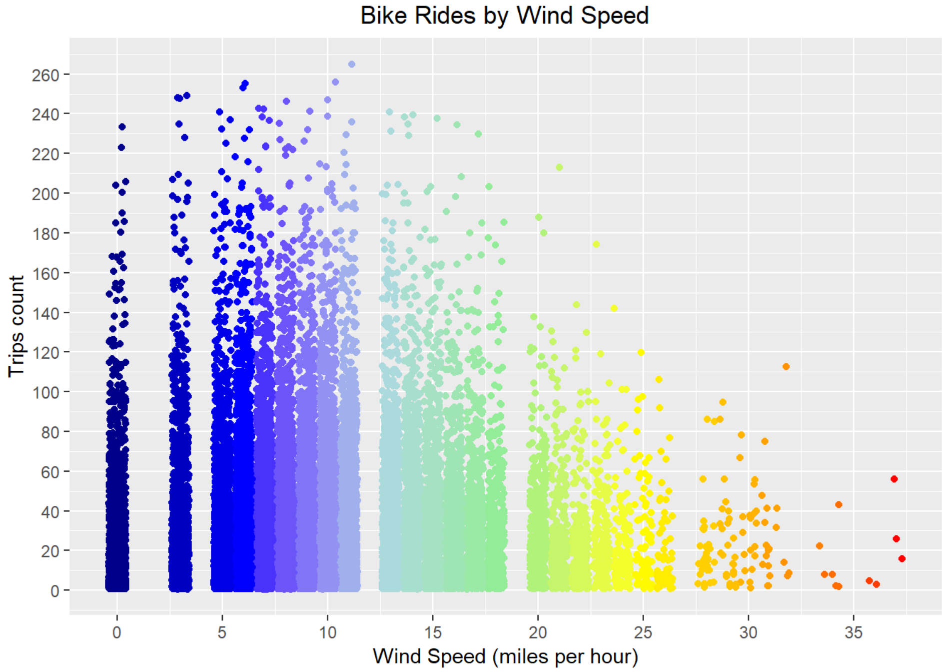
On the other hand, the humidity against trips count plot portrays that there are less trips when humity is either too low or too high, and the count is similar between 40 and 80 relative humidity.

```
humidity_count <- ggplot(df_ml, aes(x = humidity, y = n))+
  geom_point(aes(color = humidity))+
  theme(plot.title = element_text(hjust = 0.5))+
  scale_color_gradient(high='purple',low='green')+
  scale_x_continuous(" Relative Humidity", breaks = seq(0, 100, by = 20))+
  scale_y_continuous("Trips count", breaks = seq( 0, 340, by = 20))+
  ggtitle("Bike Rides by Relative Humidity ")

humidity_count
```



To conclude, our last plot shows that wind speed is negatively correlated with the number of trips.

```
winds_count <- ggplot(df_ml, aes(x = wind_speed, y = n))+
  geom_jitter(aes(color = wind_speed))+
  theme(legend.position = "none", plot.title = element_text(hjust = 0.5))+
  scale_x_continuous("Wind Speed (miles per hour)", breaks = seq(0, 50, by = 5))+
  scale_y_continuous("Trips count", breaks = seq( 0, 340, by = 20))+
  ggtitle("Bike Rides by Wind Speed")+
  scale_color_gradientn(colors=c('dark blue','blue','light blue','light green','yellow','orange','red'))

winds_count
```



Creating a Predictive Model

Machine learning was used to create a linear regression model for the number of trips in a given date and hour (n), which is a continuous outcome. To fit the model, I analyzed both the Multiple and Adjusted R Squared, as well as p values, significance codes and root mean square error.

Firstly, the dataset was split using a 0.75 ratio.

```
set.seed(123)
sample <- sample.split(df_ml, SplitRatio = 0.75)
train <- subset(df_ml,sample==TRUE)
test <- subset(df_ml, sample==FALSE)
```

The first linear regression model (mod1) was created using all predictors in the machine learning dataset, and a R2 equal to 0.6492 was obtained.

```
mod1 <- lm(n ~  month + year + day_of_week + hour_of_day + week_of_year + day_of_month + week + temperature + humidity + win
d_speed + precipitation + drizzle + rain + snow + ice_pellets + thunder + season + holiday, data = trips_train)

summary(mod1)
```

```
##
## Call:
## lm(formula = n ~ month + year + day_of_week + hour_of_day + week_of_year +
##     day_of_month + week + temperature + humidity + wind_speed +
##     precipitation + drizzle + rain + snow + ice_pellets + thunder +
##     season + holiday, data = trips_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -98.001 -11.666  -1.433  10.229 138.993
##
## Coefficients: (1 not defined because of singularities)
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -4.11707    3.53101  -1.166 0.243640
## month           -6.55150    2.59502  -2.525 0.011590 *
## year2016        10.53911    0.66452  15.860  < 2e-16 ***
## year2017        17.62968    0.66024  26.702  < 2e-16 ***
## year2018        26.70902    0.75210  35.513  < 2e-16 ***
## day_of_weekMon   0.72273    0.60106   1.202 0.229213
## day_of_weekSat  -7.54436    0.59596 -12.659  < 2e-16 ***
## day_of_weekSun  -9.73993    0.58949 -16.523  < 2e-16 ***
## day_of_weekThu   2.23446    0.60279   3.707 0.000210 ***
## day_of_weekTue   2.31762    0.59320   3.907 9.38e-05 ***
## day_of_weekWed   2.48661    0.59725   4.163 3.15e-05 ***
## hour_of_day1    -3.02288    1.09688  -2.756 0.005859 **
## hour_of_day2    -5.35762    1.20943  -4.430 9.48e-06 ***
## hour_of_day3    -5.53362    1.24069  -4.460 8.24e-06 ***
## hour_of_day4    -4.54375    1.14435  -3.971 7.20e-05 ***
## hour_of_day5     1.94235    1.08817   1.785 0.074283 .
## hour_of_day6    16.24999    1.08540  14.971  < 2e-16 ***
## hour_of_day7    44.44393    1.08454  40.979  < 2e-16 ***
## hour_of_day8    83.03383    1.08052  76.846  < 2e-16 ***
## hour_of_day9    36.66632    1.08284  33.861  < 2e-16 ***
## hour_of_day10   18.35747    1.09328  16.791  < 2e-16 ***
## hour_of_day11   16.01185    1.11372  14.377  < 2e-16 ***
## hour_of_day12   19.41356    1.11496  17.412  < 2e-16 ***
## hour_of_day13   19.43460    1.10767  17.545  < 2e-16 ***
## hour_of_day14   18.98555    1.11531  17.023  < 2e-16 ***
## hour_of_day15   20.03870    1.11170  18.025  < 2e-16 ***
## hour_of_day16   28.47911    1.10566  25.758  < 2e-16 ***
## hour_of_day17   58.23738    1.09447  53.210  < 2e-16 ***
## hour_of_day18   68.21089    1.08963  62.600  < 2e-16 ***
## hour_of_day19   47.59662    1.08816  43.741  < 2e-16 ***
## hour_of_day20   30.59855    1.08095  28.307  < 2e-16 ***
## hour_of_day21   18.57121    1.07003  17.356  < 2e-16 ***
## hour_of_day22   10.48545    1.08276   9.684  < 2e-16 ***
## hour_of_day23    3.32267    1.07844   3.081 0.002066 **
## week_of_year2    1.70555    1.77260   0.962 0.335976
## week_of_year3    6.73157    2.07242   3.248 0.001164 **
## week_of_year4    4.89140    2.57732   1.898 0.057730 .
## week_of_year5    8.25779    2.89568   2.852 0.004353 **
## week_of_year6    8.50687    3.38956   2.510 0.012091 *
## week_of_year7   11.05485    3.91300   2.825 0.004731 **
## week_of_year8   15.12138    4.50323   3.358 0.000787 ***
## week_of_year9   15.15269    5.12689   2.956 0.003125 **
## week_of_year10  15.79111    5.79875   2.723 0.006472 **
## week_of_year11  18.71411    6.39280   2.927 0.003423 **
## week_of_year12  18.58075    6.97054   2.666 0.007692 **
## week_of_year13  24.16591    7.55141   3.200 0.001376 **
## week_of_year14  23.25879    8.06573   2.884 0.003935 **
## week_of_year15  27.56842    8.65343   3.186 0.001446 **
## week_of_year16  31.63909    9.24741   3.421 0.000624 ***
## week_of_year17  35.88157    9.85747   3.640 0.000273 ***
## week_of_year18  37.38409   10.45351   3.576 0.000350 ***
## week_of_year19  38.84125   11.04403   3.517 0.000438 ***
## week_of_year20  39.86602   11.64906   3.422 0.000622 ***
## week_of_year21  41.84001   12.24784   3.416 0.000637 ***
## week_of_year22  47.07323   12.84403   3.665 0.000248 ***
## week_of_year23  50.82139   13.43176   3.784 0.000155 ***
## week_of_year24  49.44377   14.02640   3.525 0.000424 ***
## week_of_year25  53.42696   14.62870   3.652 0.000261 ***
## week_of_year26  54.58965   15.22377   3.586 0.000337 ***
## week_of_year27  52.00642   15.84086   3.283 0.001029 **
## week_of_year28  58.35365   16.43505   3.551 0.000385 ***
## week_of_year29  57.45125   17.02980   3.374 0.000744 ***
## week_of_year30  60.76677   17.62542   3.448 0.000567 ***
## week_of_year31  67.14532   18.17999   3.693 0.000222 ***
## week_of_year32  68.04894   18.75288   3.629 0.000286 ***
## week_of_year33  69.43149   19.34963   3.588 0.000334 ***
## week_of_year34  72.85501   19.93526   3.655 0.000258 ***
## week_of_year35  73.72862   20.51001   3.595 0.000326 ***
```

```
## week_of_year36  76.99665   21.14120   3.642 0.000271 ***
## week_of_year37  80.81760   21.73441   3.718 0.000201 ***
## week_of_year38  81.35819   22.34487   3.641 0.000272 ***
## week_of_year39  84.46082   22.89896   3.688 0.000226 ***
## week_of_year40  87.54158   23.51721   3.722 0.000198 ***
## week_of_year41  89.15907   24.10721   3.698 0.000218 ***
## week_of_year42  90.39414   24.69280   3.661 0.000252 ***
## week_of_year43  89.36218   25.28418   3.534 0.000410 ***
## week_of_year44  90.35526   25.85052   3.495 0.000475 ***
## week_of_year45  90.46532   26.41691   3.425 0.000617 ***
## week_of_year46  90.21725   27.00358   3.341 0.000837 ***
## week_of_year47  87.00652   27.59544   3.153 0.001619 **
## week_of_year48  88.91560   28.17081   3.156 0.001600 **
## week_of_year49  88.22952   28.74712   3.069 0.002150 **
## week_of_year50  88.41983   29.33903   3.014 0.002584 **
## week_of_year51  87.49260   29.93179   2.923 0.003470 **
## week_of_year52  83.88567   30.52726   2.748 0.006004 **
## week_of_year53  86.36860   30.96301   2.789 0.005286 **
## day_of_month    -0.24824    0.08958  -2.771 0.005591 **
## weekweekend           NA         NA      NA       NA
## temperature      0.82659    0.03504  23.590  < 2e-16 ***
## humidity        -0.19384    0.01071 -18.091  < 2e-16 ***
## wind_speed      -0.16916    0.03274  -5.167 2.40e-07 ***
## precipitation  -33.87781    5.48692  -6.174 6.79e-10 ***
## drizzle          0.47366    1.54508   0.307 0.759181
## rain            -6.71456    0.69035  -9.726  < 2e-16 ***
## snow            -2.44496    1.45881  -1.676 0.093755 .
## ice_pellets     -2.83669    4.30502  -0.659 0.509951
## thunder          6.15551    2.31041   2.664 0.007723 **
## season          -1.35087    0.75386  -1.792 0.073161 .
## holiday         -7.45639    1.00322  -7.432 1.11e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.27 on 18139 degrees of freedom
## Multiple R-squared:  0.651,  Adjusted R-squared:  0.6492
## F-statistic: 348.9 on 97 and 18139 DF,  p-value: < 2.2e-16
```

In order to build a stronger model (mod2), variables month and week_of_year, which are highly correlated, were dropped one at a time, to investigate the model with the highest R2 and to avoid multicollinearity. If the variable month was kept, R2 would equal 0.6432. Whereas if week_of_year was kept, R2 would be 0.6491. So week_of_year was kept. Precipitation, rain, humidity and drizzle are also highly correlated. After performing the same steps, humidity remained in the model (R2 is equal to 0.6453).

Turning to the coefficients table again, there is an error “1 not defined because of singularities”, related to week, which is also highly related to day_of_week. Removing day_of_ week would cause the model to have its R2 reduced, so the week variable was dropped.

Next, insignificant variables were removed, one by one, according to the significance codes on the coefficients table. The one with the highest Pr(>|t|) was dropped first. After dropping thunder, snow and day_of_month, our R2 remains unaltered. The predictor season was also dropped, not only does it have low significance, but it is also correlated to week_of_year. All these modifications created a simpler model, with a slightly smaller Adjusted R2 (mod2 = 0.6452), as verified on the table below. Also, running mod2 to make predictions generated a RMSE equal to 21.37375.

```
mod2 <- lm(n ~ year + day_of_week + hour_of_day + week_of_year +  temperature + humidity + wind_speed + ice_pellets + holida
y, data = trips_train)
summary(mod2)
```

```
##
## Call:
## lm(formula = n ~ year + day_of_week + hour_of_day + week_of_year +
##     temperature + humidity + wind_speed + ice_pellets + holiday,
##     data = trips_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -98.847 -11.718  -1.403  10.215 140.395
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -9.15293    1.74331  -5.250 1.54e-07 ***
## year2016       10.62158    0.66221  16.040  < 2e-16 ***
## year2017       17.61234    0.66365  26.539  < 2e-16 ***
## year2018       26.62868    0.75583  35.231  < 2e-16 ***
## day_of_weekMon  0.95637    0.59788   1.600 0.109705
## day_of_weekSat -7.78266    0.59257 -13.134  < 2e-16 ***
## day_of_weekSun -9.75183    0.59105 -16.499  < 2e-16 ***
## day_of_weekThu  2.00932    0.59343   3.386 0.000711 ***
## day_of_weekTue  2.32352    0.59567   3.901 9.63e-05 ***
## day_of_weekWed  2.29440    0.59349   3.866 0.000111 ***
## hour_of_day1   -2.85927    1.10281  -2.593 0.009530 **
## hour_of_day2   -5.03617    1.21575  -4.142 3.45e-05 ***
## hour_of_day3   -5.49673    1.24738  -4.407 1.06e-05 ***
## hour_of_day4   -4.30284    1.15039  -3.740 0.000184 ***
## hour_of_day5    2.03889    1.09386   1.864 0.062347 .
## hour_of_day6   16.43155    1.09122  15.058  < 2e-16 ***
## hour_of_day7   44.29039    1.09026  40.624  < 2e-16 ***
## hour_of_day8   82.64211    1.08567  76.121  < 2e-16 ***
## hour_of_day9   35.92569    1.08673  33.058  < 2e-16 ***
## hour_of_day10  17.36734    1.09573  15.850  < 2e-16 ***
## hour_of_day11  14.90816    1.11556  13.364  < 2e-16 ***
## hour_of_day12  18.32981    1.11593  16.426  < 2e-16 ***
## hour_of_day13  18.34522    1.10915  16.540  < 2e-16 ***
## hour_of_day14  17.80258    1.11631  15.948  < 2e-16 ***
## hour_of_day15  18.98643    1.11294  17.060  < 2e-16 ***
## hour_of_day16  27.53640    1.10772  24.859  < 2e-16 ***
## hour_of_day17  57.33826    1.09719  52.259  < 2e-16 ***
## hour_of_day18  67.66259    1.09349  61.878  < 2e-16 ***
## hour_of_day19  47.25074    1.09312  43.226  < 2e-16 ***
## hour_of_day20  30.22046    1.08614  27.824  < 2e-16 ***
## hour_of_day21  18.35307    1.07561  17.063  < 2e-16 ***
## hour_of_day22  10.44152    1.08865   9.591  < 2e-16 ***
## hour_of_day23   3.29740    1.08432   3.041 0.002361 **
## week_of_year2   0.11080    1.66013   0.067 0.946790
## week_of_year3   3.65688    1.64738   2.220 0.026443 *
## week_of_year4  -0.68328    1.75860  -0.389 0.697623
## week_of_year5   1.68140    1.64952   1.019 0.308060
## week_of_year6   0.61642    1.69012   0.365 0.715325
## week_of_year7   1.60300    1.65391   0.969 0.332450
## week_of_year8   3.93046    1.68472   2.333 0.019658 *
## week_of_year9   1.60364    1.65614   0.968 0.332908
## week_of_year10 -0.01327    1.65521  -0.008 0.993601
## week_of_year11  1.21769    1.75414   0.694 0.487576
## week_of_year12 -0.91780    1.67840  -0.547 0.584504
## week_of_year13  3.40126    1.66777   2.039 0.041424 *
## week_of_year14  1.80145    1.66734   1.080 0.279962
## week_of_year15  3.90277    1.69172   2.307 0.021067 *
## week_of_year16  6.30081    1.70650   3.692 0.000223 ***
## week_of_year17  9.12043    1.73420   5.259 1.46e-07 ***
## week_of_year18  9.37933    1.73918   5.393 7.02e-08 ***
## week_of_year19  9.40042    1.72219   5.458 4.87e-08 ***
## week_of_year20  8.48514    1.76026   4.820 1.44e-06 ***
## week_of_year21  8.38594    1.78729   4.692 2.73e-06 ***
## week_of_year22 12.82075    1.80830   7.090 1.39e-12 ***
## week_of_year23 14.24998    1.78800   7.970 1.68e-15 ***
## week_of_year24 11.07818    1.82955   6.055 1.43e-09 ***
## week_of_year25 13.29125    1.86440   7.129 1.05e-12 ***
## week_of_year26 12.89039    1.86342   6.918 4.75e-12 ***
## week_of_year27  9.30769    1.87781   4.957 7.24e-07 ***
## week_of_year28 14.05957    1.88015   7.478 7.90e-14 ***
## week_of_year29 11.23700    1.89682   5.924 3.20e-09 ***
## week_of_year30 12.94716    1.87946   6.889 5.81e-12 ***
## week_of_year31 18.42155    1.87301   9.835  < 2e-16 ***
## week_of_year32 18.04007    1.88202   9.585  < 2e-16 ***
## week_of_year33 17.85461    1.89602   9.417  < 2e-16 ***
## week_of_year34 19.42018    1.85904  10.446  < 2e-16 ***
## week_of_year35 18.24047    1.89174   9.642  < 2e-16 ***
## week_of_year36 20.06737    1.98246  10.122  < 2e-16 ***
## week_of_year37 22.27639    1.99514  11.165  < 2e-16 ***
```

```
## week_of_year38 20.85769    1.92460  10.837  < 2e-16 ***
## week_of_year39 22.29630    1.80753  12.335  < 2e-16 ***
## week_of_year40 24.57842    1.75982  13.966  < 2e-16 ***
## week_of_year41 24.45649    1.76197  13.880  < 2e-16 ***
## week_of_year42 24.00153    1.73458  13.837  < 2e-16 ***
## week_of_year43 20.70196    1.72355  12.011  < 2e-16 ***
## week_of_year44 20.95590    1.71537  12.217  < 2e-16 ***
## week_of_year45 19.91902    1.69468  11.754  < 2e-16 ***
## week_of_year46 17.63344    1.67904  10.502  < 2e-16 ***
## week_of_year47 12.84037    1.65870   7.741 1.04e-14 ***
## week_of_year48 14.18823    1.68564   8.417  < 2e-16 ***
## week_of_year49 15.88463    1.66569   9.536  < 2e-16 ***
## week_of_year50 14.51245    1.65558   8.766  < 2e-16 ***
## week_of_year51 11.28841    1.65979   6.801 1.07e-11 ***
## week_of_year52  6.28089    1.65736   3.790 0.000151 ***
## week_of_year53  8.28326    2.92937   2.828 0.004694 **
## temperature     0.85379    0.03495  24.426  < 2e-16 ***
## humidity       -0.25202    0.00942 -26.753  < 2e-16 ***
## wind_speed     -0.22408    0.03254  -6.885 5.95e-12 ***
## ice_pellets    -8.07618    4.30185  -1.877 0.060483 .
## holiday        -7.68009    1.00793  -7.620 2.67e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.39 on 18147 degrees of freedom
## Multiple R-squared:  0.647,  Adjusted R-squared:  0.6452
## F-statistic: 373.7 on 89 and 18147 DF,  p-value: < 2.2e-16
```

```
prediction2 <- predict(mod2, newdata = trips_test)
SSE_prediction2 <- sum((prediction2 - trips_test$n)^2)
RMSE_2 <- sqrt(SSE_prediction2/nrow(trips_test))
RMSE_2
```

```
## [1] 21.37375
```

There were many other attempts to build stronger models, but mod2 was the best one so far. To check if an interaction of features could improve the model, mod3 was created with the variable humidity2. This interaction made our R2 go up to 0.6479. Running mod3 to make predictions generated a RMSE equal to 21.29569, slightly better than mod2.

```
trips_train$humidity2 <- trips_train$humidity*trips_train$humidity
trips_test$humidity2 <- trips_test$humidity*trips_test$humidity

mod3 <- lm(n ~  year + day_of_week + hour_of_day + week_of_year + temperature + humidity2 + wind_speed + ice_pellets + holid
ay, data = trips_train)
summary(mod3)
```

```
##
## Call:
## lm(formula = n ~ year + day_of_week + hour_of_day + week_of_year +
##     temperature + humidity2 + wind_speed + ice_pellets + holiday,
##     data = trips_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -96.783 -11.725  -1.465  10.195 139.942
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -1.564e+01  1.652e+00  -9.472  < 2e-16 ***
## year2016        1.055e+01  6.597e-01  15.985  < 2e-16 ***
## year2017        1.761e+01  6.611e-01  26.629  < 2e-16 ***
## year2018        2.663e+01  7.530e-01  35.365  < 2e-16 ***
## day_of_weekMon  9.378e-01  5.956e-01   1.574 0.115415
## day_of_weekSat -7.797e+00  5.903e-01 -13.207  < 2e-16 ***
## day_of_weekSun -9.743e+00  5.888e-01 -16.547  < 2e-16 ***
## day_of_weekThu  1.977e+00  5.912e-01   3.343 0.000829 ***
## day_of_weekTue  2.388e+00  5.934e-01   4.025 5.73e-05 ***
## day_of_weekWed  2.354e+00  5.913e-01   3.981 6.90e-05 ***
## hour_of_day1   -2.852e+00  1.099e+00  -2.596 0.009434 **
## hour_of_day2   -4.972e+00  1.211e+00  -4.105 4.06e-05 ***
## hour_of_day3   -5.369e+00  1.243e+00  -4.320 1.57e-05 ***
## hour_of_day4   -4.131e+00  1.146e+00  -3.605 0.000313 ***
## hour_of_day5    2.163e+00  1.090e+00   1.985 0.047144 *
## hour_of_day6    1.642e+01  1.087e+00  15.105  < 2e-16 ***
## hour_of_day7    4.419e+01  1.086e+00  40.681  < 2e-16 ***
## hour_of_day8    8.249e+01  1.082e+00  76.269  < 2e-16 ***
## hour_of_day9    3.579e+01  1.082e+00  33.070  < 2e-16 ***
## hour_of_day10   1.731e+01  1.091e+00  15.867  < 2e-16 ***
## hour_of_day11   1.496e+01  1.110e+00  13.481  < 2e-16 ***
## hour_of_day12   1.842e+01  1.110e+00  16.603  < 2e-16 ***
## hour_of_day13   1.845e+01  1.103e+00  16.730  < 2e-16 ***
## hour_of_day14   1.796e+01  1.110e+00  16.186  < 2e-16 ***
## hour_of_day15   1.913e+01  1.106e+00  17.285  < 2e-16 ***
## hour_of_day16   2.757e+01  1.102e+00  25.016  < 2e-16 ***
## hour_of_day17   5.733e+01  1.092e+00  52.498  < 2e-16 ***
## hour_of_day18   6.761e+01  1.089e+00  62.094  < 2e-16 ***
## hour_of_day19   4.713e+01  1.089e+00  43.285  < 2e-16 ***
## hour_of_day20   3.010e+01  1.082e+00  27.817  < 2e-16 ***
## hour_of_day21   1.825e+01  1.072e+00  17.031  < 2e-16 ***
## hour_of_day22   1.030e+01  1.085e+00   9.499  < 2e-16 ***
## hour_of_day23   3.233e+00  1.080e+00   2.993 0.002770 **
## week_of_year2  -2.739e-02  1.653e+00  -0.017 0.986782
## week_of_year3   3.462e+00  1.640e+00   2.110 0.034852 *
## week_of_year4  -7.903e-01  1.752e+00  -0.451 0.651856
## week_of_year5   1.460e+00  1.643e+00   0.888 0.374361
## week_of_year6   5.885e-01  1.683e+00   0.350 0.726620
## week_of_year7   1.571e+00  1.648e+00   0.954 0.340182
## week_of_year8   4.047e+00  1.678e+00   2.413 0.015852 *
## week_of_year9   1.562e+00  1.650e+00   0.947 0.343669
## week_of_year10  6.418e-02  1.649e+00   0.039 0.968954
## week_of_year11  1.234e+00  1.748e+00   0.706 0.480172
## week_of_year12 -7.036e-01  1.671e+00  -0.421 0.673786
## week_of_year13  3.626e+00  1.661e+00   2.182 0.029092 *
## week_of_year14  2.145e+00  1.661e+00   1.291 0.196567
## week_of_year15  3.685e+00  1.685e+00   2.186 0.028822 *
## week_of_year16  6.523e+00  1.700e+00   3.837 0.000125 ***
## week_of_year17  9.314e+00  1.727e+00   5.391 7.08e-08 ***
## week_of_year18  9.610e+00  1.733e+00   5.546 2.96e-08 ***
## week_of_year19  9.503e+00  1.716e+00   5.539 3.08e-08 ***
## week_of_year20  8.417e+00  1.753e+00   4.801 1.59e-06 ***
## week_of_year21  8.357e+00  1.780e+00   4.694 2.70e-06 ***
## week_of_year22  1.284e+01  1.801e+00   7.132 1.03e-12 ***
## week_of_year23  1.400e+01  1.780e+00   7.861 4.02e-15 ***
## week_of_year24  1.089e+01  1.823e+00   5.974 2.36e-09 ***
## week_of_year25  1.305e+01  1.857e+00   7.027 2.18e-12 ***
## week_of_year26  1.246e+01  1.856e+00   6.714 1.94e-11 ***
## week_of_year27  8.879e+00  1.870e+00   4.748 2.07e-06 ***
## week_of_year28  1.372e+01  1.872e+00   7.330 2.39e-13 ***
## week_of_year29  1.083e+01  1.889e+00   5.731 1.01e-08 ***
## week_of_year30  1.278e+01  1.871e+00   6.832 8.62e-12 ***
## week_of_year31  1.814e+01  1.864e+00   9.729  < 2e-16 ***
## week_of_year32  1.760e+01  1.874e+00   9.394  < 2e-16 ***
## week_of_year33  1.764e+01  1.887e+00   9.347  < 2e-16 ***
## week_of_year34  1.886e+01  1.851e+00  10.191  < 2e-16 ***
## week_of_year35  1.767e+01  1.884e+00   9.382  < 2e-16 ***
## week_of_year36  1.963e+01  1.974e+00   9.945  < 2e-16 ***
## week_of_year37  2.193e+01  1.987e+00  11.036  < 2e-16 ***
```

```
## week_of_year38  2.053e+01  1.916e+00  10.713  < 2e-16 ***
## week_of_year39  2.189e+01  1.799e+00  12.164  < 2e-16 ***
## week_of_year40  2.440e+01  1.751e+00  13.932  < 2e-16 ***
## week_of_year41  2.428e+01  1.754e+00  13.843  < 2e-16 ***
## week_of_year42  2.367e+01  1.728e+00  13.701  < 2e-16 ***
## week_of_year43  2.051e+01  1.716e+00  11.949  < 2e-16 ***
## week_of_year44  2.060e+01  1.708e+00  12.061  < 2e-16 ***
## week_of_year45  1.984e+01  1.688e+00  11.752  < 2e-16 ***
## week_of_year46  1.736e+01  1.673e+00  10.381  < 2e-16 ***
## week_of_year47  1.250e+01  1.652e+00   7.568 3.98e-14 ***
## week_of_year48  1.419e+01  1.678e+00   8.457  < 2e-16 ***
## week_of_year49  1.559e+01  1.659e+00   9.396  < 2e-16 ***
## week_of_year50  1.424e+01  1.649e+00   8.635  < 2e-16 ***
## week_of_year51  1.129e+01  1.653e+00   6.830 8.78e-12 ***
## week_of_year52  6.326e+00  1.651e+00   3.832 0.000127 ***
## week_of_year53  7.910e+00  2.918e+00   2.711 0.006716 **
## temperature     8.630e-01  3.481e-02  24.788  < 2e-16 ***
## humidity2      -2.131e-03  7.273e-05 -29.301  < 2e-16 ***
## wind_speed     -2.286e-01  3.234e-02  -7.070 1.61e-12 ***
## ice_pellets    -7.735e+00  4.286e+00  -1.805 0.071116 .
## holiday        -7.593e+00  1.004e+00  -7.562 4.16e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.31 on 18147 degrees of freedom
## Multiple R-squared:  0.6496, Adjusted R-squared:  0.6479
## F-statistic:   378 on 89 and 18147 DF,  p-value: < 2.2e-16
```

```
prediction3 <- predict(mod3, newdata = trips_test)
SSE_prediction3 <- sum((prediction3 - trips_test$n)^2)
RMSE_3 <- sqrt(SSE_prediction3/nrow(trips_test))
RMSE_3
```

```
## [1] 21.29569
```

Even though using interaction slightly improved our linear regression model, let’s try to fit a Random Forest model, and check if this classification algorithm might improve our prediction accuracy.

```
model_1 <- randomForest(n ~ ., data=trips_train)


prediction <- predict(model_1, trips_test)
model_output <- cbind(trips_test, prediction)
model_output$log_prediction <- log(model_output$prediction)
model_output$log_n <- log(model_output$n)

rmse(model_output$log_n,model_output$log_prediction)
```

```
## [1] 0.483301
```

All of the independent variables were used in the model, as the algorithm is smart to only select the ones that will help with our prediction. A RMSE of 0.48 was obtained, which is still very high, but better than the linear regression models presented above.

Conclusion and Recommendations

The model created on this study generates a prediction that does not get the very exact number of trips in a given time a date most of the times. However, it clearly understands general demand trends, such as if there will be low, medium, high or extreme demand at a certain moment. Maybe a more refined weather dataset as well as adding other elements, such as traffic-related variables, could help improve its accuracy.

Yet, this analysis exemplifies how automatically generated data can depict trends and variations on the service usage, and it is a relevant content for business decision making. Identifying elements that influence demand may help better allocate resources and leverage ROI.

Seasonality analysis can be used to plan preventive maintenance. Our model can also be used to help predict daily demand, avoiding the shortage of bikes in peak hours (and vice versa). Foreseeing times of high demand is important to work on the expansion of the network.

Identifying the most frequent journeys is valuable to plan expansion in these regions. And further analysis into these journeys might also reveal not so obvious factors that influence demand, such as road conditions for bicycle traffic between them.

The existing gap between male and female ridership shows that there are barriers for women to use the service. Identifying such barriers - which I suppose might be related to safety and bike path conditions around the city - could be arranged by the company alongside third parties, in order to provide a service that meets the needs of women as well, which could lead to thousands of new customers.