Bicycle Rents (Extra Trees Algorithm)

Guido Retegui

1/2/2022

Introduction

Many American cities have communal bike sharing stations where you can rent bicycles by the hour or day. Washington, D.C. is one of these cities. The District collects detailed data on the number of bicycles people rent by the hour and day.

We compiled a dataset like the one produced by Hadi Fanaee-T at the University of Porto which we'll be working with in this project.

First we are going to install the required packages:

```
#Downloading and installing packages, this a could take a few minutes

if(!require(caret)) install.packages("caret",repos ="http://cran.us.r-project.org")
if(!require(tidyverse)) install.packages("tidyverse",repos ="http://cran.us.r-project.org")
if(!require(corrplot)) install.packages("corrplot",repos ="http://cran.us.r-project.org")
install.packages("rJava", repos = "https://cran.rstudio.com")
install.packages("extraTrees",repos ="https://cran.r-project.org/")


# Please, make sure you have the right version of Java (32 or 64 bit) to match architectures with R,
# otherwise the console will show an error

library(rJava)

# This instruction solves memory problems for "extraTrees" package

options( java.parameters = "-Xmx4g" )
library(extraTrees)

library(lubridate)

library(creditmodel)

# Create "temp" file and download database

temp <- tempfile()
download.file("https://raw.githubusercontent.com/GuidoRetegui/bike_rental_hour_saegus/master/bike_rental

brhs <- read.csv(temp)
```

Quick Review

Let's analyse our database. We can see a quick review of it:

```
str(brhs)
```

```
## 'data.frame':    52272 obs. of  18 variables:
##  $ dteday    : chr  "2012-01-01" "2012-01-01" "2012-01-01" "2012-01-01" ...
##  $ hr        : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ casual    : int  5 15 16 11 0 0 1 1 4 13 ...
##  $ registered: int  43 77 57 39 8 5 1 6 10 27 ...
##  $ cnt       : int  48 92 73 50 8 5 2 7 14 40 ...
##  $ season    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ holiday   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ fulldate  : chr  "2012-01-01 00:00:00" "2012-01-01 01:00:00" "2012-01-01 02:00:00" "2012-01-01 03
##  $ temp      : chr  "9Â°C" "7Â°C" "6Â°C" "5Â°C" ...
##  $ atemp     : chr  "8Â°C" "7Â°C" "6Â°C" "4Â°C" ...
##  $ dewpoint  : chr  "3Â°C" "3Â°C" "3Â°C" "2Â°C" ...
##  $ hum       : chr  "66%" "76%" "81%" "81%" ...
##  $ windspeed : chr  "9 km/h" "2 km/h" "2 km/h" "6 km/h" ...
##  $ pressure  : chr  "1020.0mb" "1020.0mb" "1020.0mb" "1020.0mb" ...
##  $ weathersit: chr  "Clair" "Clair" "Clair" "Clair" ...
##  $ fog       : chr  "false" "false" "false" "false" ...
##  $ wea_desc  : chr  " DispersÃ©s Nuages Ã  1200m" " Peu de Nuages Ã  1200m" " Peu de Nuages Ã  1200m
##  $ workingday: chr  "False" "False" "False" "False" ...
```

The file contains 62608 rows, with each row representing the number of bike rentals for a single hour of a single day.

With this function, we can see the first entries of the dataset:

```
head(brhs)
```

```
##       dteday hr casual registered cnt season holiday            fulldate temp
## 1 2012-01-01  0      5         43  48      1       0 2012-01-01 00:00:00 9Â°C
## 2 2012-01-01  1     15         77  92      1       0 2012-01-01 01:00:00 7Â°C
## 3 2012-01-01  2     16         57  73      1       0 2012-01-01 02:00:00 6Â°C
## 4 2012-01-01  3     11         39  50      1       0 2012-01-01 03:00:00 5Â°C
## 5 2012-01-01  4      0          8   8      1       0 2012-01-01 04:00:00 5Â°C
## 6 2012-01-01  5      0          5   5      1       0 2012-01-01 05:00:00 4Â°C
##   atemp dewpoint hum windspeed pressure  weathersit   fog
## 1  8Â°C     3Â°C 66%    9 km/h 1020.0mb       Clair false
## 2  7Â°C     3Â°C 76%    2 km/h 1020.0mb       Clair false
## 3  6Â°C     3Â°C 81%    2 km/h 1020.0mb       Clair false
## 4  4Â°C     2Â°C 81%    6 km/h 1020.0mb       Clair false
## 5  3Â°C     2Â°C 81%    7 km/h 1020.0mb Peu Nuageux false
## 6  2Â°C     3Â°C 93%    9 km/h 1020.0mb Peu Nuageux false
##                       wea_desc workingday
## 1  DispersÃ©s Nuages Ã  1200m      False
## 2      Peu de Nuages Ã  1200m      False
## 3      Peu de Nuages Ã  1200m      False
## 4      Peu de Nuages Ã  1200m      False
## 5       Claire moins de 3700m      False
## 6       Claire moins de 3700m      False
```

Main Variables

In this section, we'll see a description of the variables contained in the dataset with their respective name and type.

- dteday (character) - The date of the rentals
- hr (integer) - rental's Hour
- season (integer) - The season in which the rentals occurred
- holiday (integer) - Whether or not the day was a holiday
- fulldate (character) - date and time (datetime format)
- workingday (character) - Whether or not the day was a working day
- weathersit (character) - The weather
- temp (character) - The temperature (°C)
- atemp (character) - The adjusted temperature (°C)
- hum (character) - The humidity (%)
- windspeed (character) - The wind speed (km/h)
- pressure (character) - Pressure (mb)
- dewpoint (character) - Temperature to which air must be cooled to become saturated with water vapor (in celsius).
- fog (character) - Presence of fog (true/false).
- wea_desc (character) - Description of the weather.
- casual (integer) - The number of casual riders (people who hadn't previously signed up with the bike sharing program)
- registered (integer) - The number of registered riders (people who had already signed up)
- cnt (integer) - The total number of bike rentals (casual + registered)

Data Cleaning

Last day contains many missing values. We can remove them because one day in five years is insignificant. This loop fixes the problem:

```r
#Remove last day from the dataset

for (i in 52249:52272){
  brhs <- brhs[-c(52249),]
}
```

Secondly, to optimize the analysis, we are going to convert some variables to numeric. For example, we know how "temp" variable is measured, the important data takes place exclusively in the number. To remove useless parts of these characters and change their types into numeric, we use:

```r
#delete "Â°C" from the variable 'temp'
brhs$temp <- lapply(brhs$temp, gsub, pattern='Â°C', replacement = '')
#change variable's type to numeric
brhs$temp <- as.numeric(brhs$temp)

#delete "Â°C" from the variable 'atemp'
brhs$atemp <- lapply(brhs$atemp, gsub, pattern='Â°C', replacement = '')
#change variable's type to numeric
brhs$atemp <- as.numeric(brhs$atemp)

#delete "Â°C" from the variable 'dewpoint'
```

```r
brhs$dewpoint <- lapply(brhs$dewpoint, gsub, pattern='Â°C', replacement = '')
#change variable's type to numeric
brhs$dewpoint <- as.numeric(brhs$dewpoint)

#delete "%" from the variable 'hum'
brhs$hum <- lapply(brhs$hum, gsub, pattern='%', replacement = '')
#change variable's type to numeric
brhs$hum <- as.numeric(brhs$hum)

#delete " km/h" from the variable 'windspeed'
brhs$windspeed <- lapply(brhs$windspeed, gsub, pattern=' km/h', replacement = '')
#change variable's type to numeric
brhs$windspeed <- as.numeric(brhs$windspeed)

#delete " mb" from the variable 'pressure'
brhs$pressure <- lapply(brhs$pressure, gsub, pattern='mb', replacement = '')
#change variable's type to numeric
brhs$pressure <- as.numeric(brhs$pressure)
```

We can see “fog” variable is a character vector. With this instruction we change the values to 1 in case of fog, and to 0 if it is not. After that, we change the variable’s type to numeric to complete the cleaning.

```r
# Change "fog" to a dummy variable

for (i in 1:52248) {
  if (brhs$fog[i] == "true" & !is.na(brhs$fog[i])) {
    brhs$fog[i] = 1
  }
  else{
    brhs$fog[i] = 0
  }
}

#change variable's type to numeric

brhs$fog <- as.numeric(brhs$fog)
```

Taking the “dteday” variable, we create the “weekdayname” and the “weekdaynum” columns with the “strftime” function. The second one will be important to make the evaluation, so we change its variable type to numeric as well. “Weekdaynum” assigns the weekdays as decimal numbers (1 is for Monday).

```r
# Create the "weekdayname" column

brhs <- mutate(brhs, weekdayname = strftime(brhs$dteday, "%A"))

# Create the "weekdaynum" column. Weekdaynum as a decimal number (1-7, Monday is 1).

brhs <- mutate(brhs, weekdaynum = strftime(brhs$dteday, "%u"))

#change variable's type to numeric

brhs$weekdaynum <- as.numeric(brhs$weekdaynum)
```

Taking the "cnt" variable, we convert it to create the values in their logarithms:

```
# Create the column "lcnt" with the cnt's logarithm

brhs <- mutate(brhs, lcnt = log(brhs$cnt))
```

With the "year" function we can take that value from the date and extract it to create a new column with "mutate":

```
# Create the column "year"

brhs <- mutate(brhs, year = year(brhs$dteday))
```

The last modification of the dataset that we are going to make is to create the column "workingday_dummy" and assign them the value of 0. With the for loop we check if the value containing in the column "workingday" is "True" to assign this information to our new column.

```
# Create the column "workingday_dummy"

brhs <- mutate(brhs, workingday_dummy = 0)

#Assigning "Workingday_dummy" column the value "1" when it is a workingday.

for (i in 1:52248) {
  if (brhs$workingday[i] == "True" & !is.na(brhs$workingday[i])) {
    brhs$workingday_dummy[i] = 1
  }
}
```

We can check these changes we implemented with "head" instruction again:

```
# Create the column "year"

head(brhs)
```

```
##       dteday hr casual registered cnt season holiday            fulldate temp
## 1 2012-01-01  0      5         43  48      1       0 2012-01-01 00:00:00    9
## 2 2012-01-01  1     15         77  92      1       0 2012-01-01 01:00:00    7
## 3 2012-01-01  2     16         57  73      1       0 2012-01-01 02:00:00    6
## 4 2012-01-01  3     11         39  50      1       0 2012-01-01 03:00:00    5
## 5 2012-01-01  4      0          8   8      1       0 2012-01-01 04:00:00    5
## 6 2012-01-01  5      0          5   5      1       0 2012-01-01 05:00:00    4
##   atemp dewpoint hum windspeed pressure  weathersit fog
## 1     8        3  66         9     1020       Clair   0
## 2     7        3  76         2     1020       Clair   0
## 3     6        3  81         2     1020       Clair   0
## 4     4        2  81         6     1020       Clair   0
## 5     3        2  81         7     1020 Peu Nuageux   0
## 6     2        3  93         9     1020 Peu Nuageux   0
##                       wea_desc workingday weekdayname weekdaynum     lcnt year
## 1  DispersÃ©s Nuages Ã  1200m      False     domingo          7 3.871201 2012
## 2      Peu de Nuages Ã  1200m      False     domingo          7 4.521789 2012
## 3      Peu de Nuages Ã  1200m      False     domingo           7 4.290459 2012
## 4      Peu de Nuages Ã  1200m      False     domingo           7 3.912023 2012
## 5       Claire moins de 3700m      False     domingo           7 2.079442 2012
## 6       Claire moins de 3700m      False     domingo           7 1.609438 2012
##   workingday_dummy
## 1                0
## 2                0
## 3                0
## 4                0
## 5                0
## 6                0
```

Once we have a cleaner data, we can proceed to the analysis itself.

Data Analysis

Summary of bicycle rents

With "group_by" we can make the analysis by day. The following code shows the sum of bicycles rented (and its mean per hour) by day

```
# Summary of total rents per day and mean per hour

brhs %>% group_by(dteday) %>% summarize (rents = sum(cnt), mean = mean(cnt))
```

```
## # A tibble: 2,187 x 3
##    dteday     rents  mean
##    <chr>      <int> <dbl>
##  1 2012-01-01  2260  94.2
##  2 2012-01-02  1937  84.2
##  3 2012-01-03  2219  92.5
##  4 2012-01-04  2357  98.2
##  5 2012-01-05  3251 135.
##  6 2012-01-06  4054 169.
##  7 2012-01-07  4480 187.
##  8 2012-01-08  3408 142
##  9 2012-01-09  2325  96.9
## 10 2012-01-10  3573 149.
## # ... with 2,177 more rows
```

We can also differentiate by how many users that rented the bicycles were registered and how many of them were not.

```
# Accumulated rents per day, differentiated by "registered" and "casual" variables

sum_day_cnt <- brhs %>% group_by(dteday) %>%
  summarize(day_reg = sum(registered),
            day_casual = sum(casual),
            day_count = sum(cnt))
sum_day_cnt
```

```
## # A tibble: 2,187 x 4
##    dteday     day_reg day_casual day_count
##    <chr>        <int>      <int>     <int>
##  1 2012-01-01    1572        688      2260
##  2 2012-01-02    1693        244      1937
##  3 2012-01-03    2130         89      2219
##  4 2012-01-04    2262         95      2357
##  5 2012-01-05    3111        140      3251
##  6 2012-01-06    3750        304      4054
##  7 2012-01-07    3406       1074      4480
##  8 2012-01-08    2807        601      3408
##  9 2012-01-09    2219        106      2325
## 10 2012-01-10    3400        173      3573
## # ... with 2,177 more rows
```

The mean per day of bicycle rents by casual users and registered users are:

```
# Mean of rents per day, differentiated by "registered" and "casual" variables

sum_day_cnt %>% summarize(reg_p_day = mean(day_reg), cas_p_day = mean(day_casual),
                          cnt_p_day = mean(day_count))
```

```
## # A tibble: 1 x 3
##   reg_p_day cas_p_day cnt_p_day
##       <dbl>     <dbl>     <dbl>
## 1     6342.     1785.     8127.
```

The total rents of the entire dataset, differentiated by casual and registered users:

```
# Total rents, differentiated by "registered" and "casual" variables

brhs %>% summarize(registered = sum(registered), casual = sum(casual), cnt= sum(cnt))
```

```
##   registered  casual      cnt
## 1   13869391 3904451 17773842
```

Summary of main variables

We can show the mean per hour of temp, atemp, dewpoint, hum, windspeed, pressure.

```
# Summary of principal variables

brhs %>% summarize (temp = mean(temp), atemp = mean(atemp))
```

```
##       temp    atemp
## 1 15.40417 14.72162
```

```
brhs %>% summarize (dewpoint = mean(dewpoint), hum = mean(hum))
```

```
##   dewpoint      hum
## 1 7.269733 61.41739
```

```
brhs %>% summarize (windspeed = mean(windspeed), pressure = mean(pressure))
```

```
##   windspeed pressure
## 1  13.53367 1017.679
```

How many times "fog" were registered, and the probability of occurence:

```
# Hours with fog vs hours without fog, and its probability

brhs %>% summarize(fog = sum(brhs$fog==1), no_fog = sum(brhs$fog==0),
                   Pr_fog = fog/(fog+no_fog), Pr_no_fog = no_fog/(fog+no_fog))
```

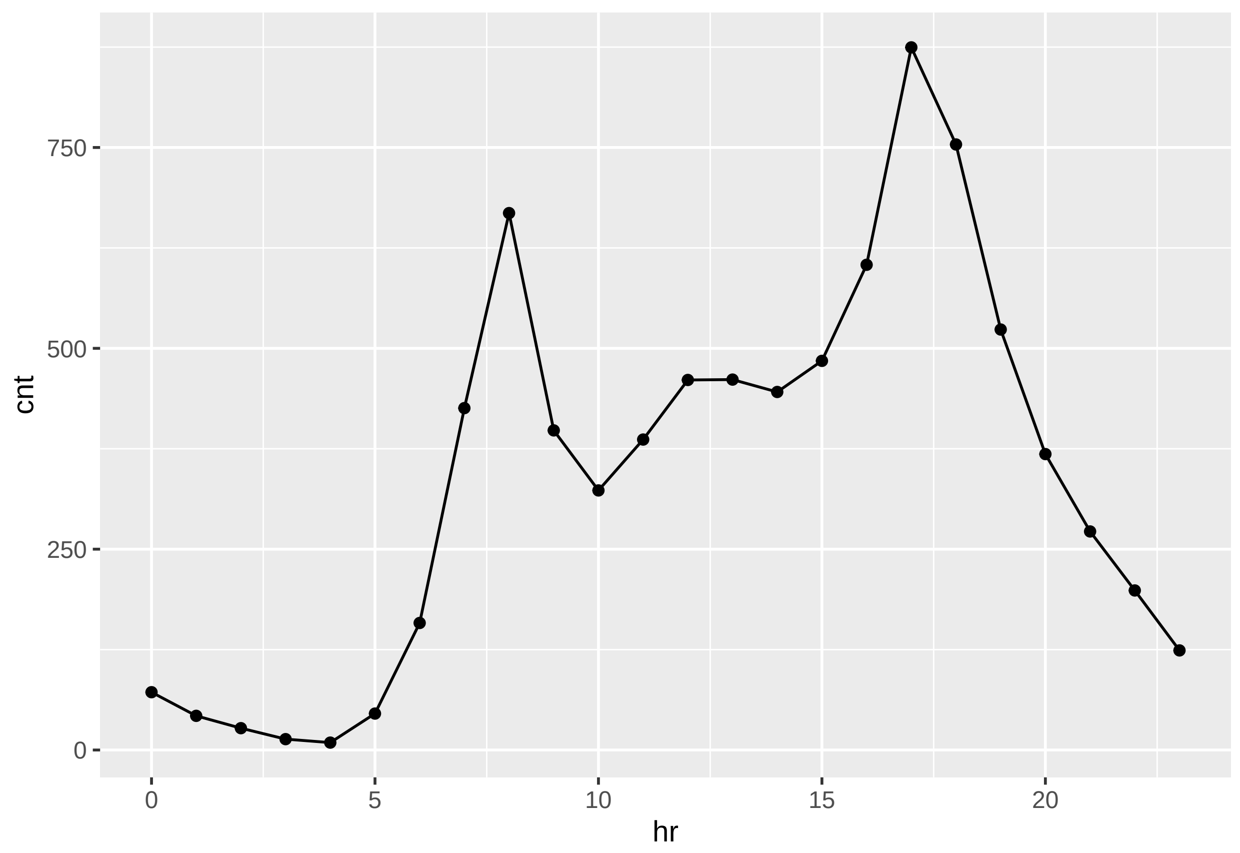
```
##    fog no_fog     Pr_fog Pr_no_fog
## 1 2744  49504 0.05251876 0.9474812
```

Graphs and more analysis

cnt by hour

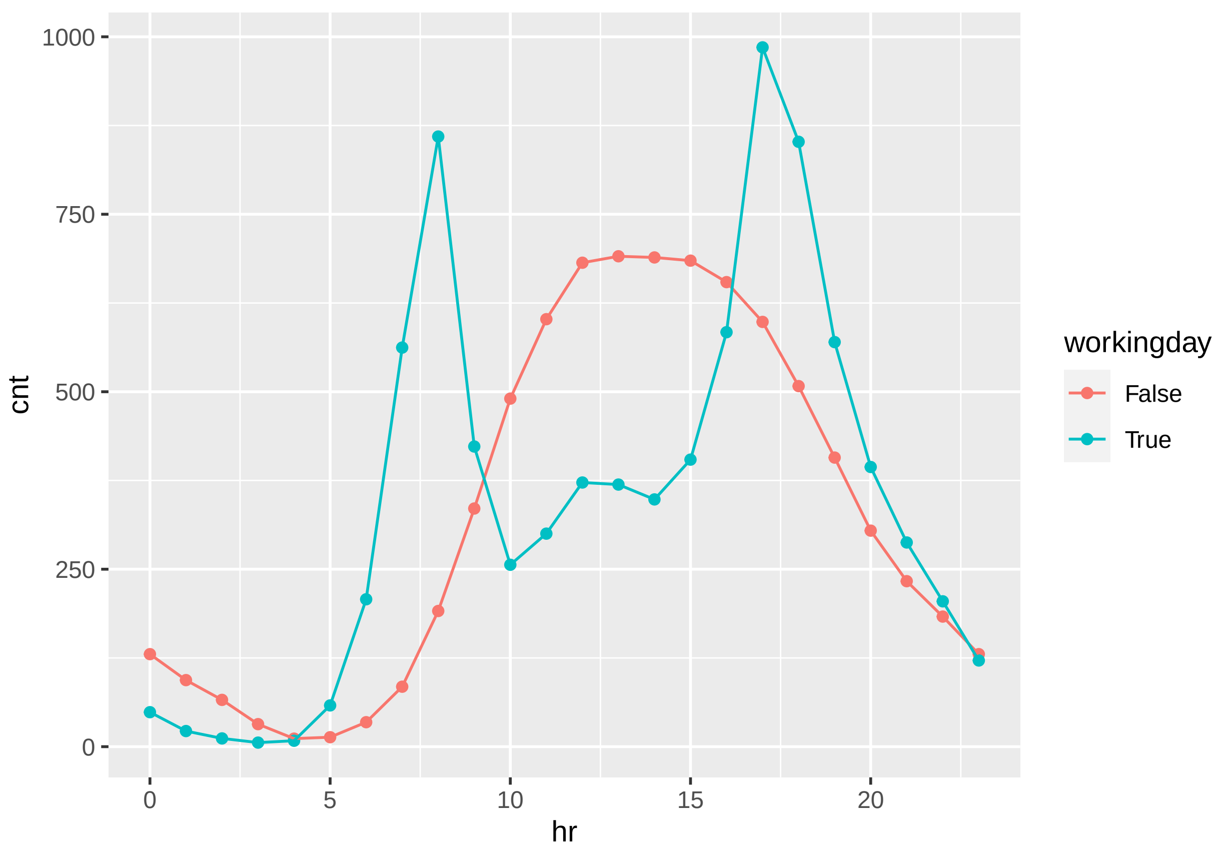
Using the "cnt" variable, we can show a plot taking its mean by hour:

```
# Rents by hour.

brhs %>% group_by(hr) %>%
  summarize(cnt=mean(cnt)) %>%
  ggplot(aes(hr, cnt)) +
  geom_point () +
  geom_line()
```



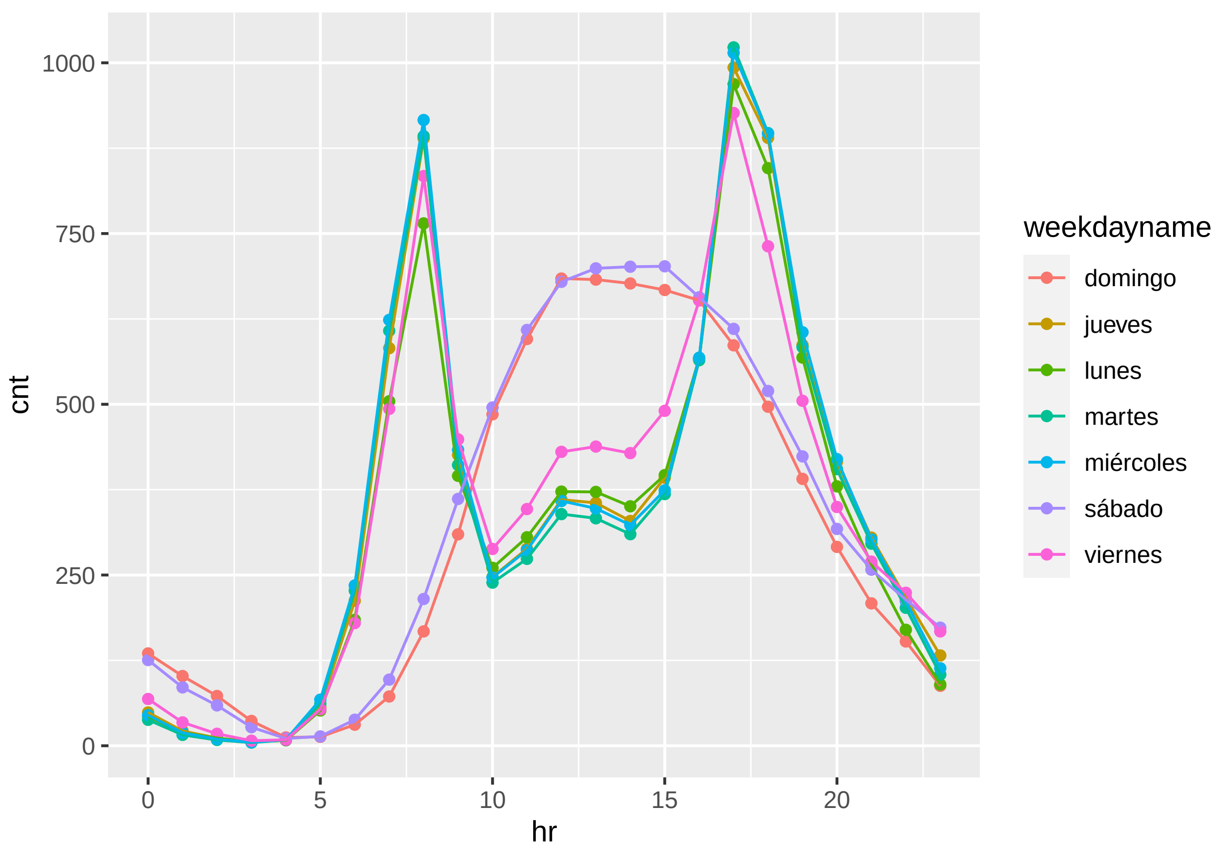
We can easily observe how the most popular hours to rent are between 8 and 9 am, and between 5 and 6 pm. We can even improve the graph by adding the "workingday" variable:

```
# Rents by hour, differentiated by workingday

brhs %>% group_by(hr, workingday) %>%
  summarize(cnt=mean(cnt), .groups = 'drop') %>%
  ggplot(aes(hr, cnt, col = workingday, group = workingday)) +
  geom_point () +
  geom_line()
```



The graph shows how on weekends these hours become less relevant. The last graph will show the differentiation by day:

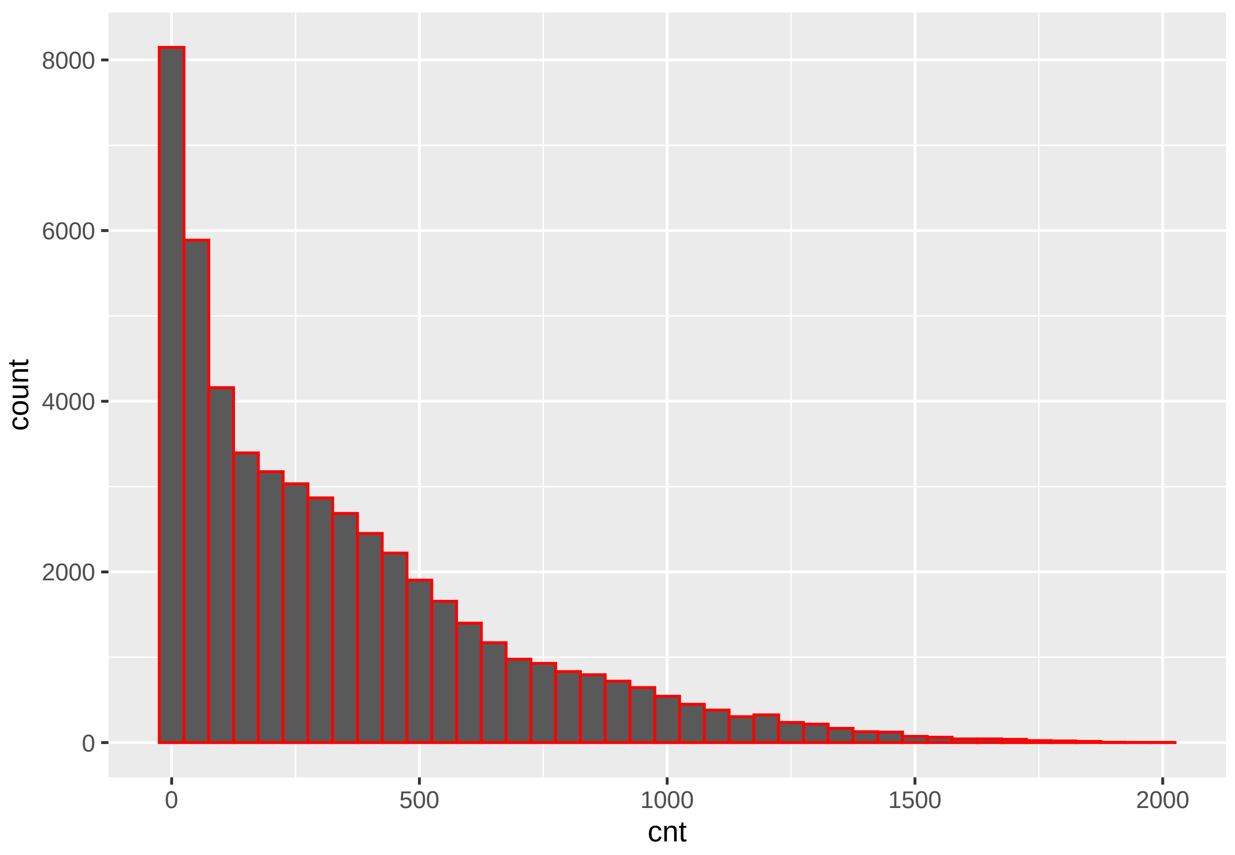
```
# Rents by hour, differentiated by day

brhs %>% group_by(hr, weekdayname) %>%
  summarize(cnt=mean(cnt), .groups = 'drop') %>%
  ggplot(aes(hr, cnt, col = weekdayname)) +
  geom_point () +
  geom_line()
```



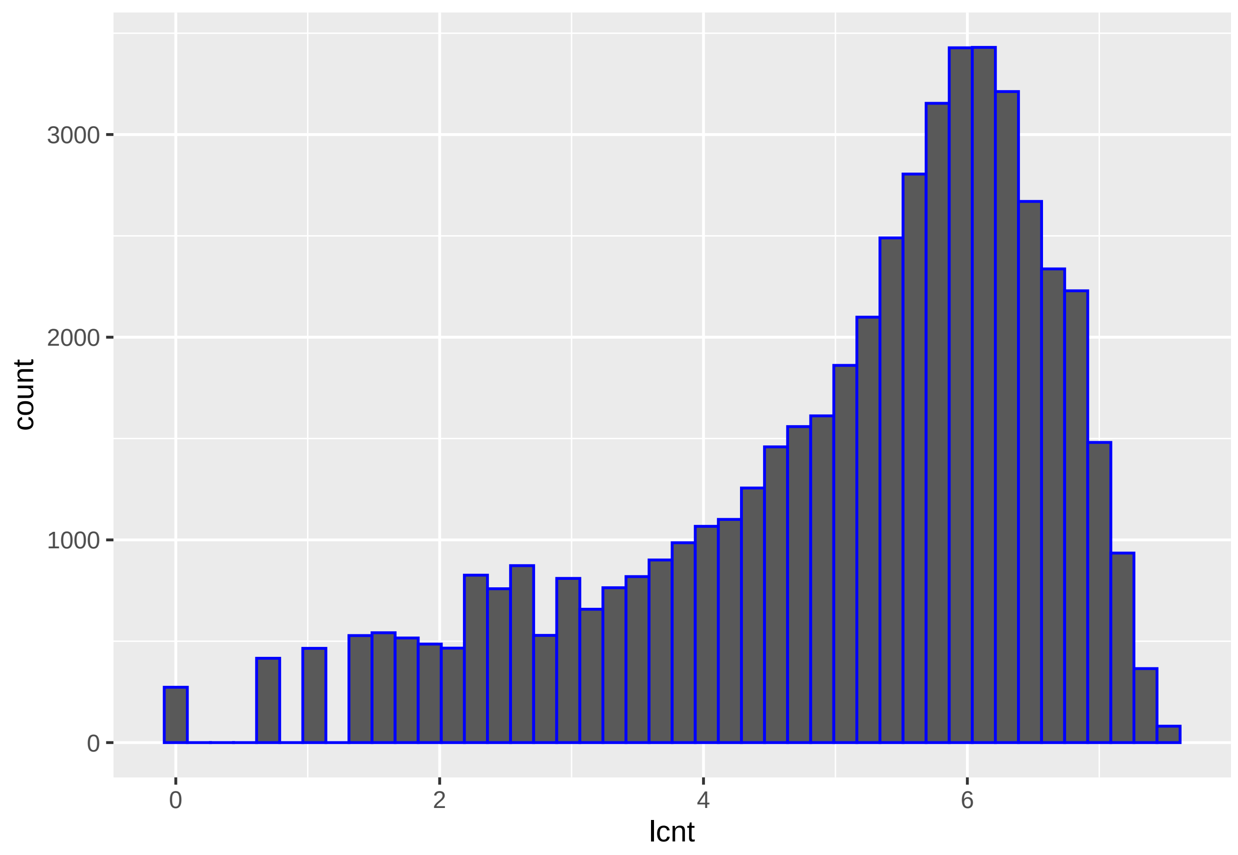
cnt and lcnt Distributions

These two following graphs show the distribution of "cnt" and "lcnt" variables respectively. We can see an approximately normal distribution when transform "cnt" to a logarithmic form.

```
# cnt histogram

brhs %>% ggplot(aes(cnt)) + geom_histogram(binwidth = 50, color="red")
```



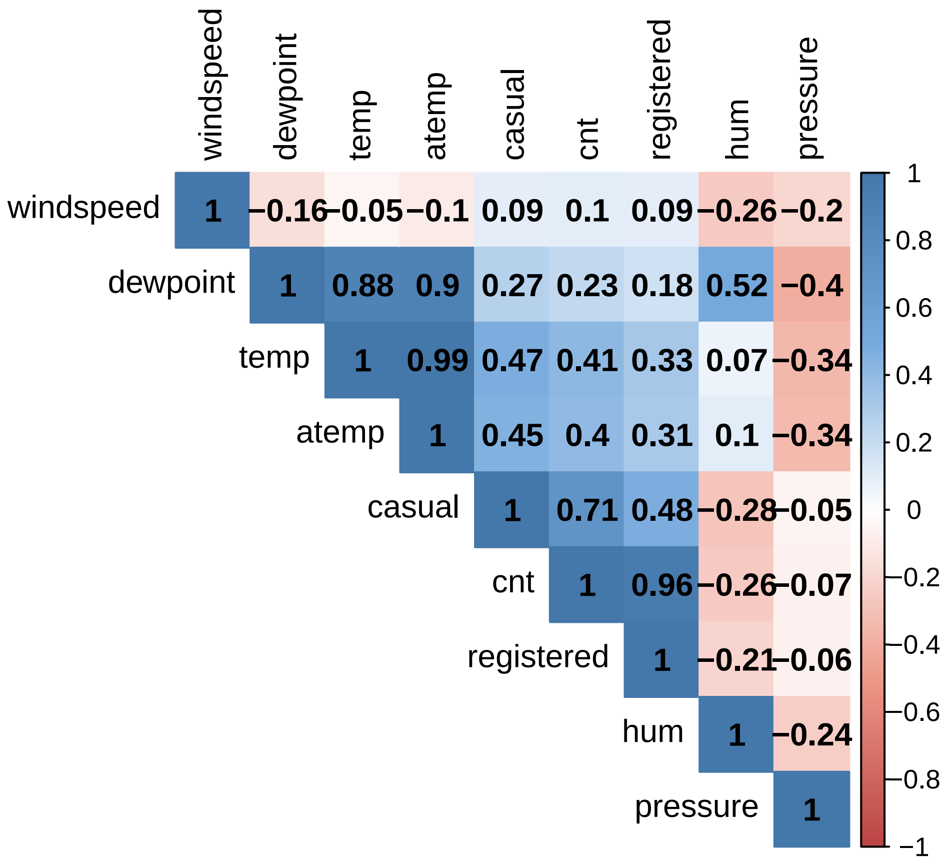
```r
# Logarithm of cnt histogram

brhs %>% ggplot(aes(lcnt)) + geom_histogram(binwidth = 0.175, color="blue")
```



Correlation between main variables

With these instructions we are going to select the main variables, make the correlations between them and show with the corrplot package, the results we obtained.

```r
#Select variables to analyse

M <- brhs %>% select(cnt, temp, atemp,
                     dewpoint, hum, windspeed, pressure, casual, registered)

# Make the correlation of the selected variables

A <- cor(M)

# Show the correlation graph

col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
corrplot(A, method="color", col=col(200),
         type="upper", order="hclust",
         addCoef.col = "black", # Add coefficient of correlation
         tl.col="black", tl.srt=90, #Text label color and rotation
         # Combine with significance
         sig.level = 0.01, insig = "blank",
)
```



	windspeed	dewpoint	temp	atemp	casual	cnt	registered	hum	pressure
windspeed	1	−0.16	−0.05	−0.1	0.09	0.1	0.09	−0.26	−0.2
dewpoint		1	0.88	0.9	0.27	0.23	0.18	0.52	−0.4
temp			1	0.99	0.47	0.41	0.33	0.07	−0.34
atemp				1	0.45	0.4	0.31	0.1	−0.34
casual					1	0.71	0.48	−0.28	−0.05
cnt						1	0.96	−0.26	−0.07
registered							1	−0.21	−0.06
hum								1	−0.24
pressure									1

Model Performance Evaluation

Root Mean Squared Logarithmic Error - RMSLE

The evaluation will consist in comparing the predicted value with the actual outcome. To compare this relationship, we need a loss function: The Root Mean Squared Logarithmic Error. To make an understandable example, when the predicted rents are consistently selected, the error is equal to zero and the algorithm is perfect. The RMSLE is defined by the formula:

$$RMSLE = \sqrt{\frac{1}{N}\sum_{u,i}(Log(\hat{y}_i + 1) - Log(y_i + 1))^2}$$

Where N is the number of observations; y_i is the observation i; $\hat{y}_i$ is the estimated value of the observation i.

```
RMSLE <- function(true, pred){
  sqrt(mean((log(pred+1) - log(true+1))^2))
}
```

Extra Trees

“Extra Trees” is an ensemble machine learning algorithm that combines the predictions from many decision trees. It is related to the widely used “random forest” algorithm, but it can often achieve as-good or better performance, although it uses a simpler algorithm to construct the decision trees used as members of the ensemble.

The Extra Trees works by creating a large number of decision trees from the training dataset. Predictions are made by averaging the prediction of the decision trees in the case of regression or using majority voting in the case of classification.

Results

In this section we make the partition of the dataset, and evaluate how the model works with different numeric vectors. Our goal is to minimize the Root Mean Squared Logarithmic Error. But first we will take the simplest model to evaluate it and compare with the other ones.

Before starting, we copy the values of "cnt" in "Y", and then we split (70% for train set and the rest for test set) with the next instruction:

```r
# Copy CNT values in "Y" and split to train (0.7) and test (0.3)

Y <- as.data.frame(brhs$cnt)

Y <- train_test_split(Y, prop = 0.7, split_type = "Random")
```

The simplest model

The first model simply uses the average "cnt" from the train set:

```r
mu <- mean(Y$train)
mu
```

```
## [1] 338.9496
```

And its RMSLE give us a value of:

```r
RMSLE_SM <- RMSLE(Y$test, mu)
RMSLE_SM
```

```
## [1] 1.712512
```

From now on we are going to use the Extra Trees algorithm, let's see the difference:

Model 2 (Extra Trees)

We assign the data base to the variable "X":

```r
X <- brhs # Assign database to X
```

Then we delete some non-numeric columns, and some numeric columns like "weekdaynum" and "workingday_dummy":

```
#Delete some non-numeric columns of X

X <- select(X, -c(dteday, cnt, lcnt, fulldate,
                  weathersit, wea_desc, workingday,
                  weekdayname, weekdaynum, workingday_dummy))
```

Next step is to make the partition of the assigned datasets, 70% will correspond to train set and the rest to the test set.

```
#Split to train (0.7) and test (0.3)

X <- train_test_split(X, prop = 0.7, split_type = "Random")
```

Here we use the ExtraTrees function, that we describe before:

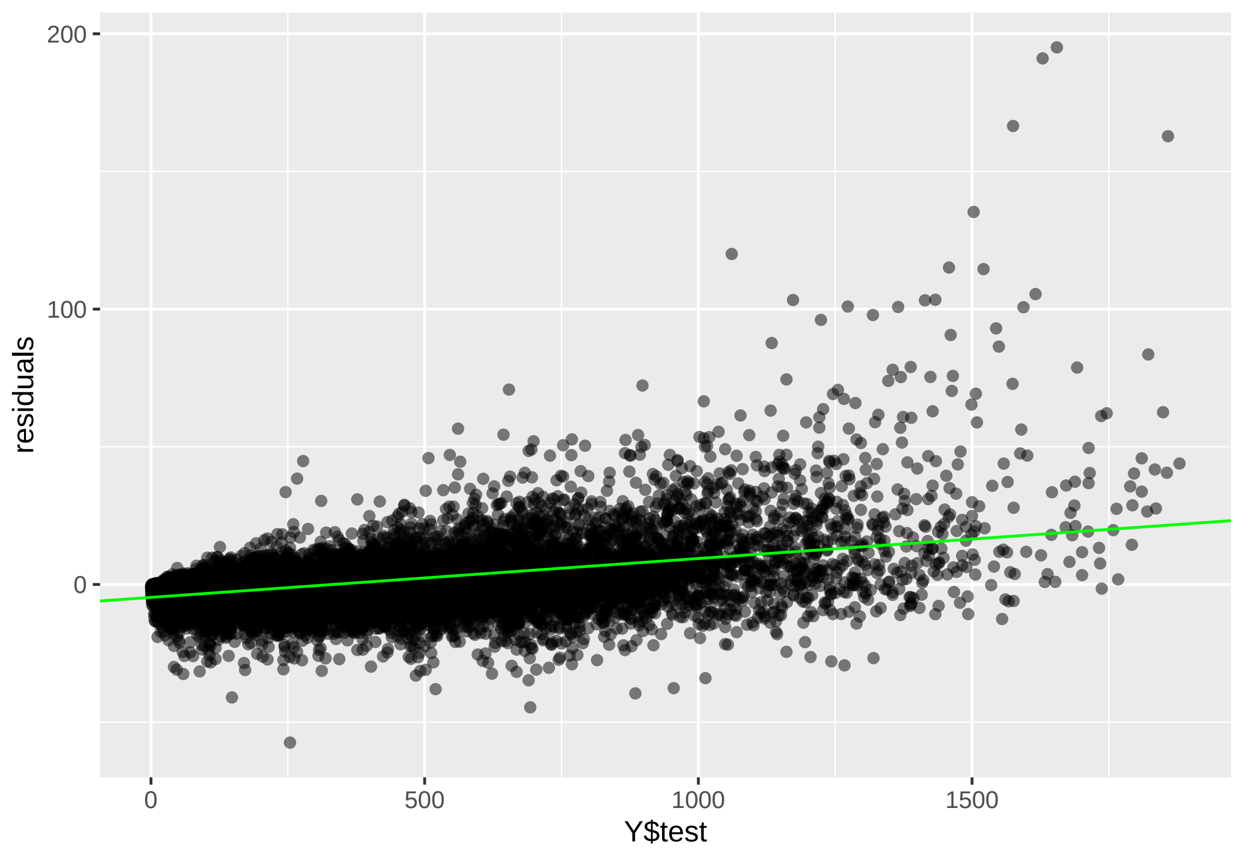
```
# Extra Trees function

ex <- extraTrees(X$train,Y$train)

y_predict <- predict(ex, X$test)
```

We calculate and prepare the variables needed to show the residuals graph:

```
#Calculate the residuals

residuals <- Y$test-y_predict

# Calculate the correlation between Y$test and residuals

r <- cor(Y$test, residuals)

# Calculate standard deviation of Y$test

s <- sd(Y$test)

# Calculate standard deviation of residuals

u <- sd(residuals)
```

Here is the plot that shows the residuals:

```
#Graph of Number of observations with residuals

ggplot(data = NULL, aes(Y$test, residuals)) +
  geom_point (alpha = 0.5) +
  geom_abline(intercept = mean(residuals)-((r*u/s)*mean(Y$test)),
              slope = r*u/s, color = "green")
```



Our first Extra Trees' RMSLE give us a value of:

```
## [1] 0.09336115
```

Model 3 (Extra Trees)

We are going to make the same process, but this time we'll select other variables to evaluate them. Let's keep "workingday_dummy" in our dataset:

```
X <- brhs # Assign database to X

#Delete the same non-numeric columns of X as the first model,
#except for "workingday_dummy"

X <- select(X, -c(dteday, cnt, lcnt,
                  fulldate, weathersit, wea_desc,
                  weekdayname, weekdaynum, workingday))

#Split to train (0.7) and test (0.3)

X <- train_test_split(X, prop = 0.7, split_type = "Random")

# Extra Trees function

ex <- extraTrees(X$train,Y$train)
```

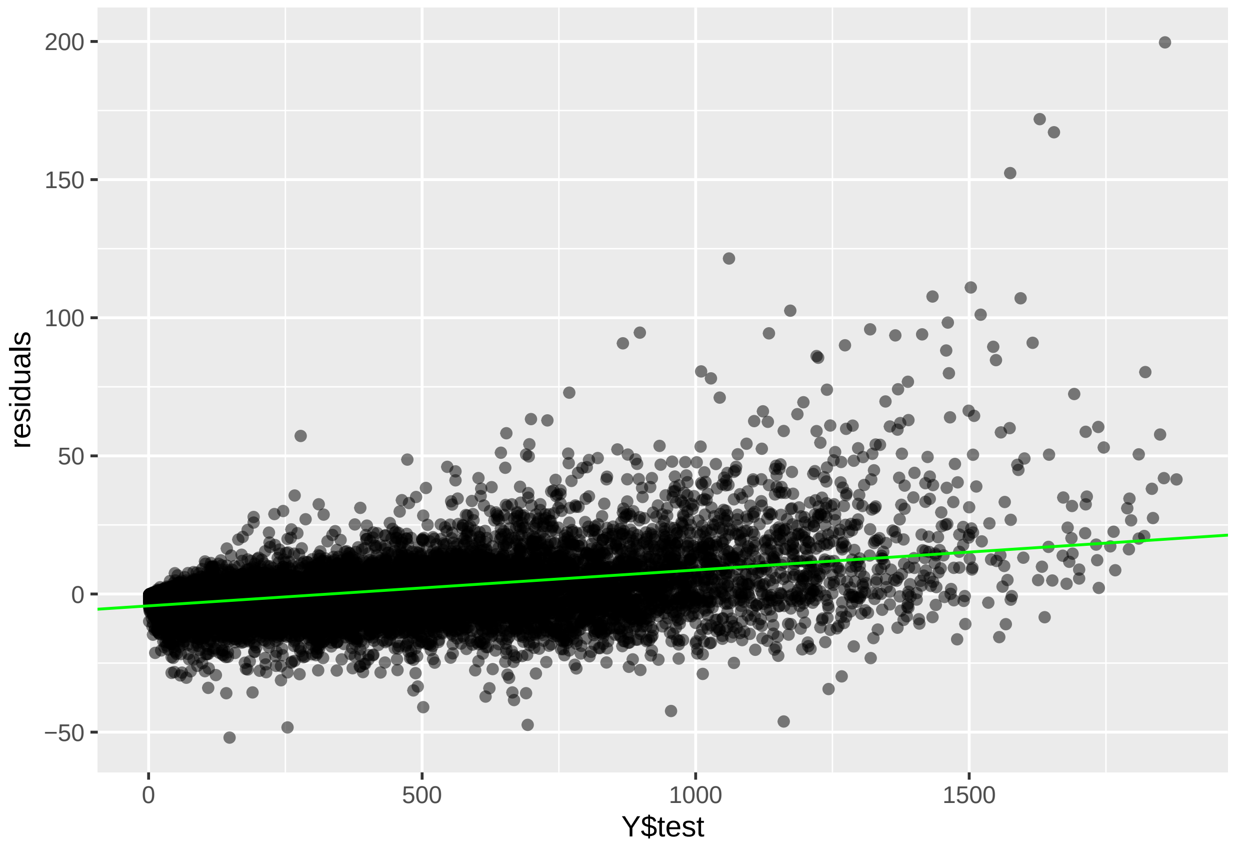
```r
y_predict <- predict(ex, X$test)
```

Now we are going to calculate the residuals again to make the plot for model 3. Note that the standard deviation of the "Y" test corresponds always to the same value because it does not depend on the new model.

```r
#Calculate the residuals

residuals <- Y$test-y_predict

# Calculate the correlation between Y$test and residuals

r <- cor(Y$test, residuals)

# Calculate standard deviation of Y$test

s <- sd(Y$test)

# Calculate standard deviation of residuals

u <- sd(residuals)
```

Here is the plot of the model 3:

```r
#Graph of Number of observations with residuals

ggplot(data = NULL, aes(Y$test, residuals)) +
  geom_point (alpha = 0.5) +
  geom_abline(intercept = mean(residuals)-((r*u/s)*mean(Y$test)),
              slope = r*u/s, color = "green")
```



Our third RMSLE:

```
## [1] 0.09055645
```

Model 4 (Extra Trees)

To finalize, in our forth model we are going to keep in the dataset the “workingday_dummy” (as our last model) and the “weekdaynum” variables. Let’s see what the result shows:

```
X <- brhs # Assign database to X

#Delete the same non-numeric columns of X as the first model,
#except for "workingday_dummy" and "weekdaynum"

X <- select(X, -c(dteday, cnt, lcnt,
                  fulldate, weathersit, wea_desc,
                  weekdayname, workingday))


#Split to train (0.7) and test (0.3)

X <- train_test_split(X, prop = 0.7, split_type = "Random")

# Extra Trees function
```

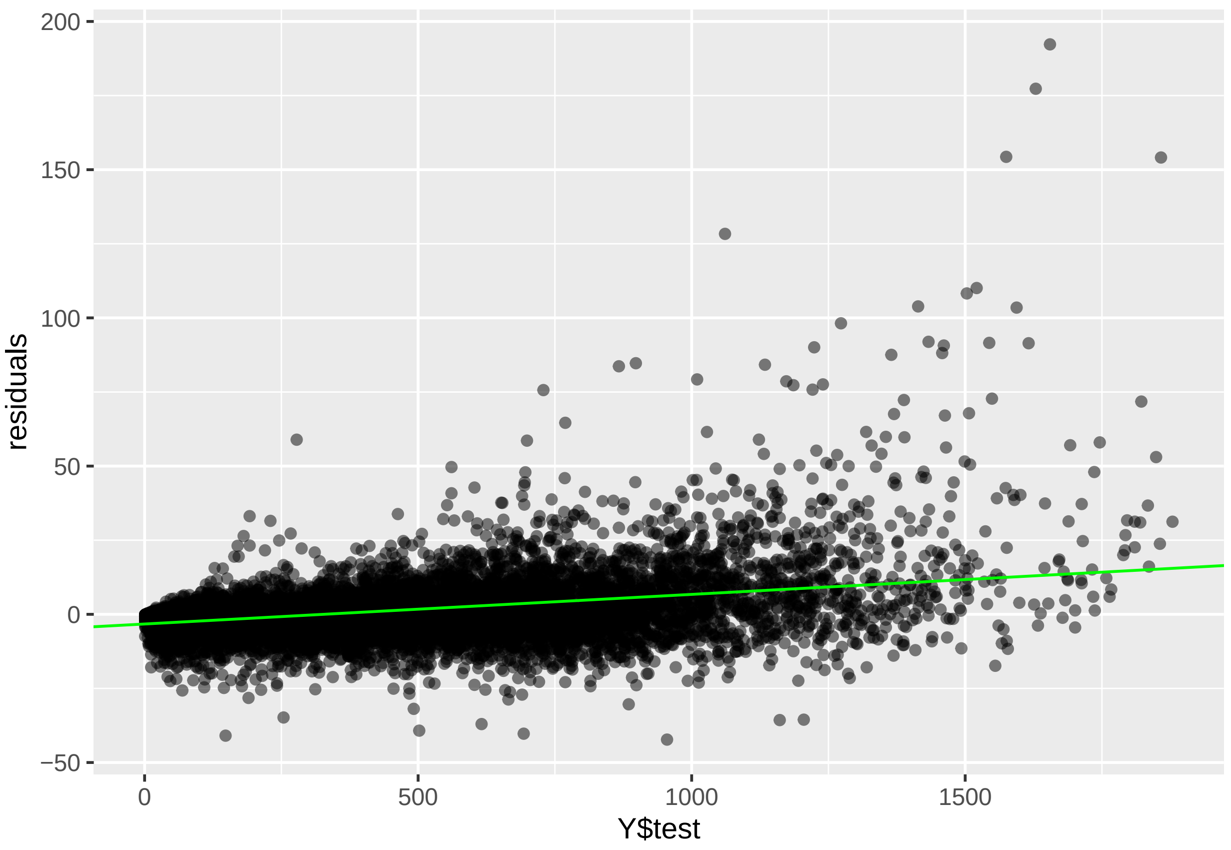
```
ex <- extraTrees(X$train,Y$train)

y_predict <- predict(ex, X$test)
```

Calculating the variables needed to make the graph:

```
#Calculate the residuals

residuals <- Y$test-y_predict

# Calculate the correlation between Y$test and residuals

r <- cor(Y$test, residuals)

# Calculate standard deviation of Y$test

s <- sd(Y$test)

# Calculate standard deviation of residuals

u <- sd(residuals)
```

Last plot:

```
#Graph of Number of observations with residuals

ggplot(data = NULL, aes(Y$test, residuals)) +
  geom_point (alpha = 0.5) +
  geom_abline(intercept = mean(residuals)-((r*u/s)*mean(Y$test)),
              slope = r*u/s, color = "green")
```



And here it is our final RMSLE:

```
## [1] 0.06680875
```

Conclusion

Once we have downloaded the database with all the respective required packages, we proceeded to clean the data deleting some rows, creating some vectors and changing the variable's type to numeric (to make them more analyzable). In the Data Analysis section, we showed how the rents vary (depending on the hour and the day), what the mean of the main variables is, and the correlation between them. After that, we explained how the algorithm "Extra Trees" works, and what formula we are going to use to evaluate its performance. Finally, in the Results Section, we evaluated the simplest model that gave us a RMSLE of 1.71. Once we implemented "Extra Trees" algorithm, we showed how the results improve to 0.0934, 0.0906 and 0.0668 respectively (an improvement of 96% from the first model).

```
## # A tibble: 1 x 4
##   Model_1 Model_2 Model_3 Model_4
##     <dbl>   <dbl>   <dbl>   <dbl>
## 1    1.71  0.0934  0.0906  0.0668
```