# R documentation for the use of combined traffic and pollution data

# Description

This production is a tool for R, and for the use of combined traffic and pollution data. This work follows the work of David Carslaw, who created the package Openair. Openair has been developed for the purpose of analyzing air quality data.

The idea behind this work is to be able to work in parallel with traffic and pollution data. The idea came from [Buisson et al., 2018] where the authors decided to find a direct relation linking traffic data (speed, flow) and the pollution concentration (NOx and PM10). An important data has also been used in this work, which is the weather conditions (speed and direction of wind, humidity rate, temperature).

# Dataset

The dataset to use on which all the work is based on is a dataframe which variables are:

* ***The date and time of the data collection***. These data are supposed to be in the *POSIXct format*. The function: as.POSIXct(date, format = ‘%d/%m/%Y %H:%M:%S’)might be useful for that purpose.
* ***The flow on the section.*** This data’s dimension is vehicle/hour. It is a *float* number.
* ***The speed on the section***. This data’s dimension is kilometer/hour. It is a *float* number.
* ***The NOx concentration***. This data’s dimension is µg/m3. It is a *float* number.
* ***The NOx background concentration.*** It is the concentrations from on a background station. This data’s dimension is µg/m3. It is a *float* number.
* ***The wind speed***. This data’s dimension is km/h. It is a *float* number.
* ***The humidity rate.*** This data’s dimension is %. It is a *float* number.
* ***The temperature***. This data’s dimension is °C. It is a *float* number.

The pollution data and the traffic data sensors should be very close in order to study the most pertinent data. All missing data should be NAs in the dataset. The weather sensor however can be in a wider area (same city for instance).



Table 1: Format of the dataset

# Analysis functions

## plot\_quartiles

This function plots the evolution of the median, and two surrounding percentiles.

The arguments of the function are

* DF: the dataframe which has the pollution and traffic data
* percentile: the percentile that is wanted to be drawn. By default, the algorithm will draw 25th and 75th percentiles.
* col\_var: it is the number the column that is wanted to be plotted, Default is NOx column.
* var\_name: it is the name of the variable that is studied. Default is NOx.

Use:

plot\_quartiles(DF, percentile = 25, col\_var=4, var\_name = "NOx")

Based on the dataset of Boulevard Laurent Bonnevay close to Lyon from 2015 to 2017, here are the results:

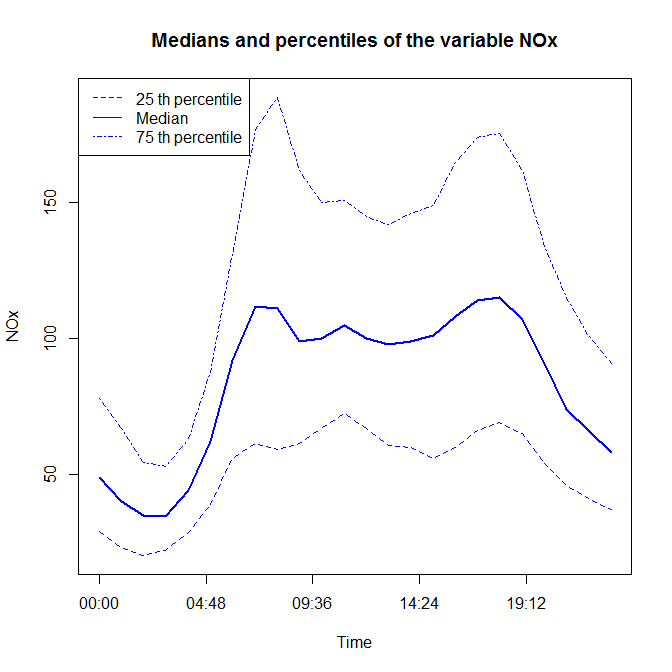


Figure 1: Result to plot\_quartiles(DF, 25, 4, "NOx")

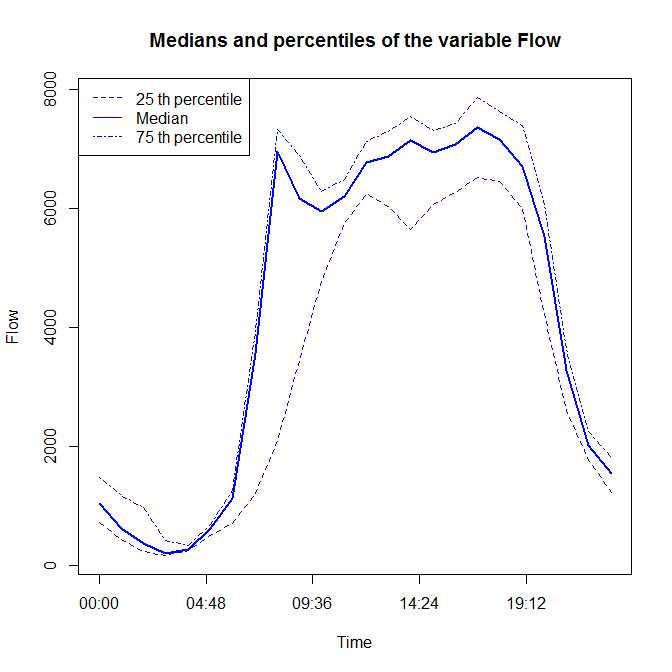


Figure 2: Result to plot\_quartiles(DF, 25, 2, "Flow")

## linear\_regression

This function makes the linear regression of the concentrations data depending on the flow. The purpose is to see if the data can be proportional (y = ax+b with b = 0)

The function returns results of the linear regression (coefficient, p-value and r squared). The function also plots the result, showing the dataset and the regression results

The function allows to do the regression for different dataset. For instance, the user can choose to take only the data where the speed is inferior to a chosen value. Same for the wind speed.

Use:

linear\_regression(DF, min\_speedlimit, max\_speedlimit, min\_windspeed, max\_windspeed)

The variables of the function are:

* DF: the dataset defined earlier
* max / min\_speedlimit: the speed under or over which the data must be for the study. The dimension of the variable is km/h.
* max / min\_windspeed: the minimum wind speed under or over which the data must be for the study. The dimension of the variable is km/h.

Based on the dataset of Boulevard Laurent Bonnevay close to Lyon from 2015 to 2017, here are the results:

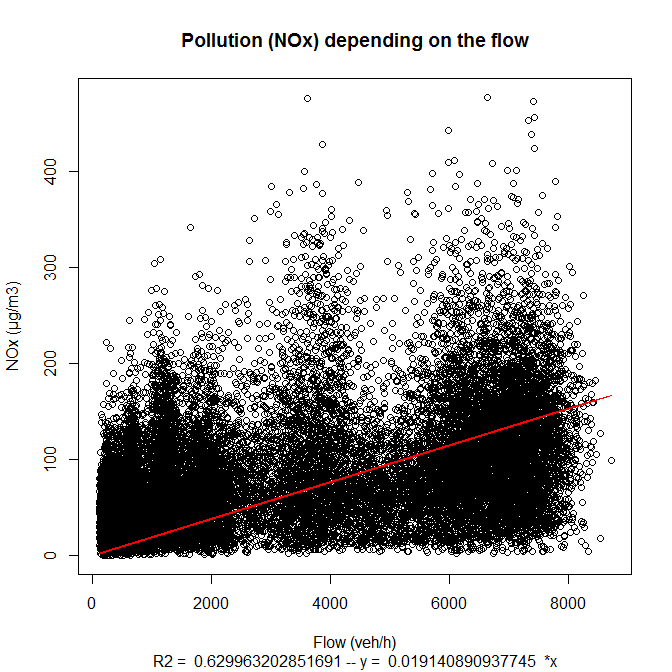


Figure 4: Result to linear\_regression(DF, max\_speelimit = 100, min\_speedlimit = 50)

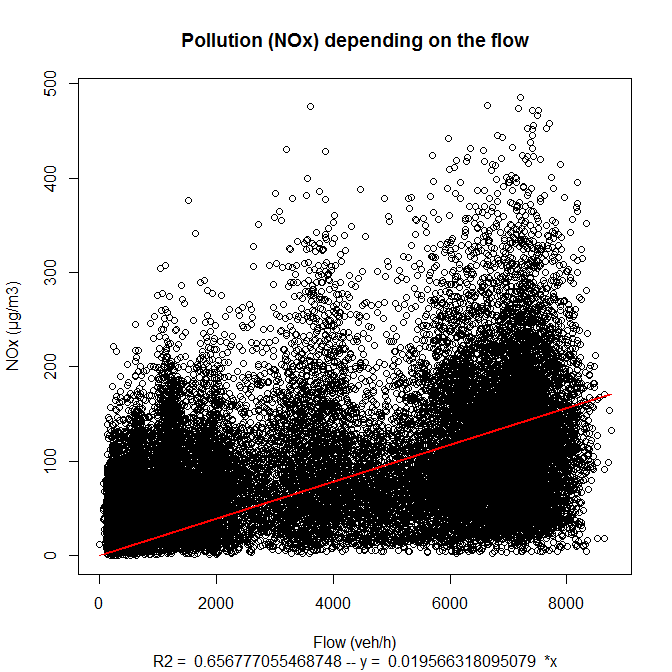


Figure 4: Result to linear\_regression(DF)

## Principal\_Component\_Analysis

This function does a PCA of the dataset, using its different variables, and plots the results of the 2 principal dimensions. Are also given the description of the different dimensions created by the PCA algorithm. The only variable that is taken into account is the original dataset DF.

Use:

Principal\_Component\_Analysis(DF)

Based on the dataset of Boulevard Laurent Bonnevay close to Lyon from 2015 to 2017, here are the results:

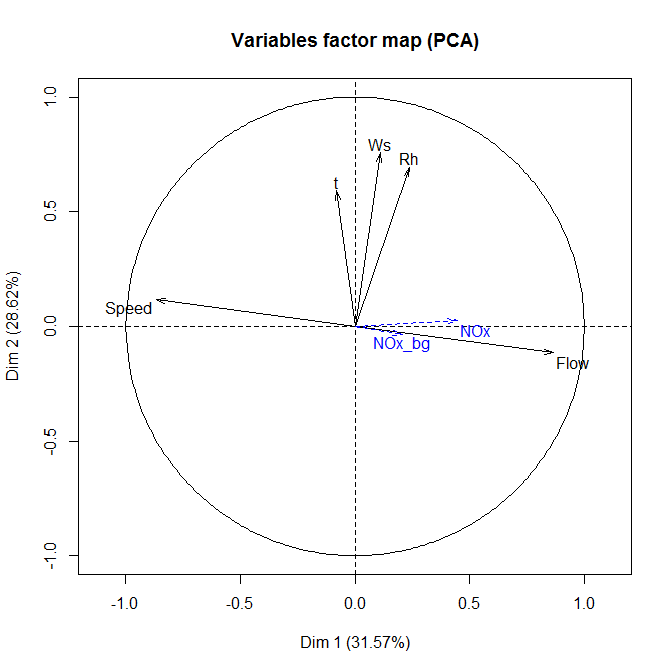
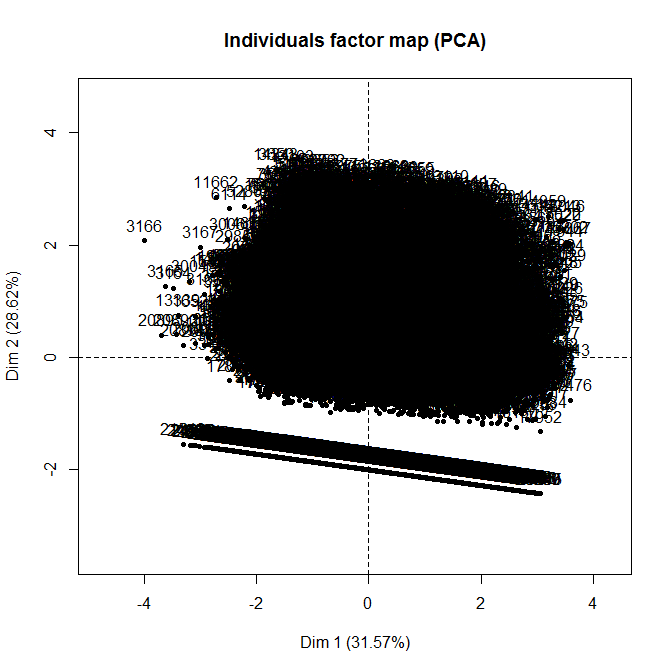


Figure 5: Result to Principal\_Component\_Analysis(DF)

## plot\_boxplot

This functions draws boxplots of the concentrations depending on the speed, for a period (for instance 2 months) and at a precise peak hour.

There is a plot for each class of flow (each 1,000 veh/hour)

Use :

plot\_boxplot(DF, period, PH)

The variables are:

* DF: same dataset with flow, speed, pollution and weather values.
* period: the description of the period, it will be used on the plot as a descrition
* the peak hour chosen: same as the period, it will be used on the plot as a descrition

Based on the dataset of Boulevard Laurent Bonnevay close to Lyon from 2015 to 2017, here are the results for the dataset on the two months of May and April, and the morning peak hour (6am-9am):

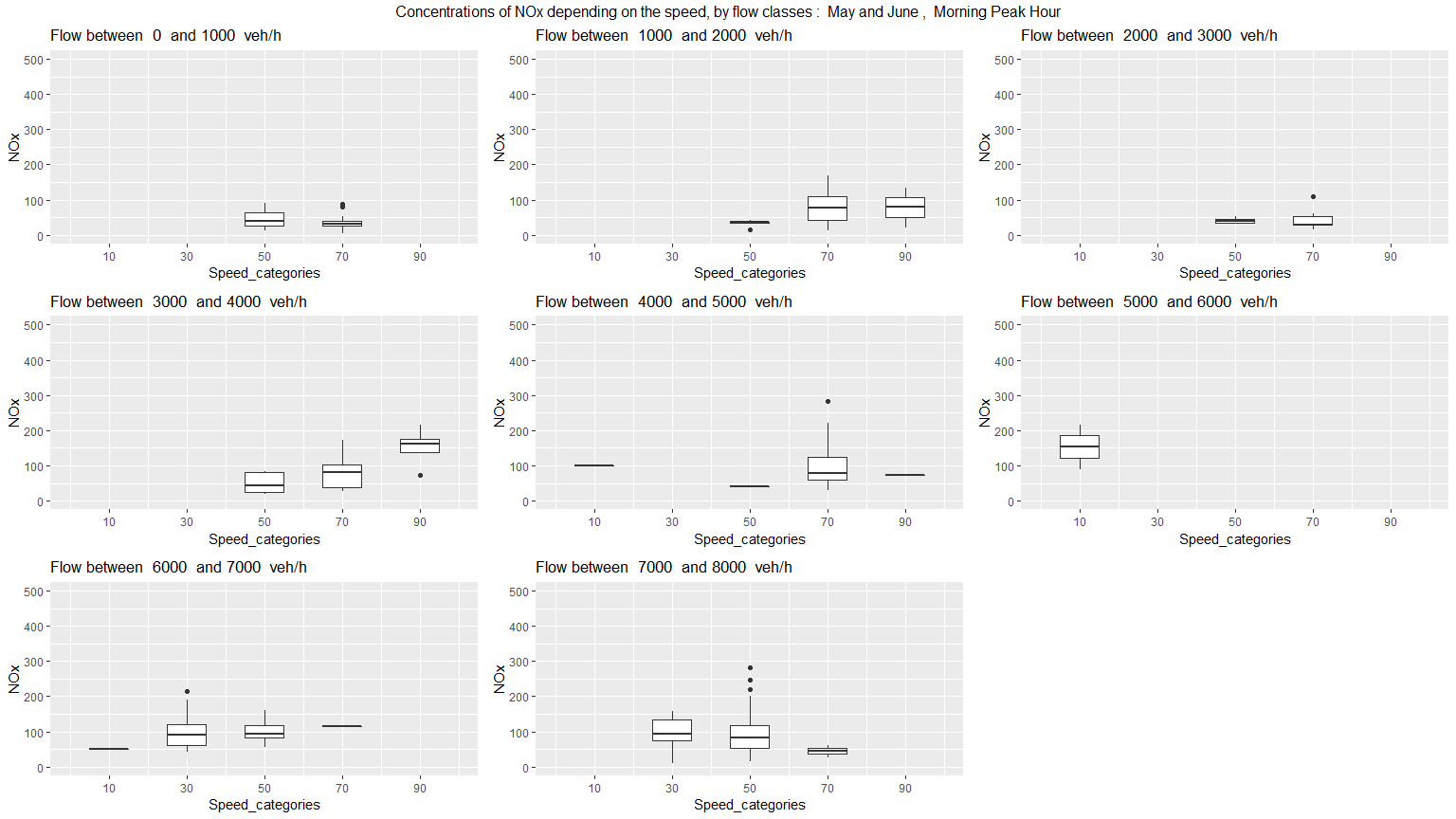


Figure 6: Result to plot\_boxplot(MayJune\_MorningPH\_DF, "May and June", "Morning Peak Hour")

# Tool functions

Some functions were also created in order to apply filters to the data. For instance, to take data for a precise period. It might be useful to study and compare a specific period (traffic regulation, extraordinary weather conditions…)

## date\_period\_selection(DF, beginning\_date, ending\_date)

This function returns a new data frame extracted from the first one, containing only the data that are in between the two dates in argument.

The format of the 2 dates are POSIXct. In order to create a POSIXct date, use the function as.POSIXct(date, format = ‘%d/%m/%Y %H:%M:%S’)

## time\_period\_selection(DF, beginning\_date, ending\_date)

This function returns a new data frame extracted from the first one, containing only the data in a specified time period. The format of the 2 hours are integers.

# Codes:

1. plot\_quartiles

plot\_quartiles = function(DF, percentile = 25, col\_var=4, var\_name = "NOx") {

# This function plots the evolution of the median, and two surrounding percentiles.

# The arguments of the function are DF the dataframe which has the pollution and traffic data,

# and the percentile that is wanted to be drawn. By default, the algorithm will draw 25th and 75th percentiles.

# and the col\_var which is the number the column that is wanted to be plotted, Default is NOx column.

# and the var\_name is the name of the variable that is studied. Default is NOx.

#install.packages("chron")

#library("chron")

# The plot is realized over a day

hours = unique(strftime(DF$Date, format = "%H:%M:%S"))

# The values to plot are in this new Table :

variable = data.frame(matrix(nrow = 2\*length(hours), ncol = 4))

colnames(variable) = c("Time", "Median", "first\_percentile", "sec\_percentile")

# Let's fill the table, while going through the hours vector.

for (i in 1:length(hours)) {

variable$Time[i] = hours[i]

variable$Median[i] = median(DF[which(strftime(DF$Date, format = "%H:%M:%S") == hours[i]), col\_var], na.rm = TRUE)

variable$first\_percentile[i] = quantile(DF[which(strftime(DF$Date, format = "%H:%M:%S") == hours[i]), col\_var], percentile\*0.01, na.rm = TRUE)

variable$sec\_percentile[i] = quantile(DF[which(strftime(DF$Date, format = "%H:%M:%S") == hours[i]), col\_var], 1-(percentile\*0.01), na.rm = TRUE)

}

variable$Time = as.POSIXct(strptime(variable$Time, format = '%H:%M:%S'))

# Now to plot the graph:

x11()

plot(times(format(variable$Time, "%H:%M:%S")), variable$Median, type = "l", lwd = 2, col = "blue",

ylim = c(min(variable$first\_percentile, na.rm = TRUE), max(variable$sec\_percentile, na.rm = TRUE)), ylab = "", xlab = "Time")

lines(times(format(variable$Time, "%H:%M:%S")), variable$first\_percentile, lty = 2, col = "blue")

lines(times(format(variable$Time, "%H:%M:%S")), variable$sec\_percentile, lty = 4, col = "blue")

legend('topleft', c(paste(percentile, "th percentile"), "Median", paste(100-percentile, "th percentile")), lty = c(2,1,4), col = c("blue", "blue", "blue"))

title(main = paste("Medians and percentiles of the variable", var\_name), ylab = paste(var\_name))

}

1. linear\_regression

linear\_regression = function(DF, max\_speedlimit = 'None', min\_speedlimit = 'None',

min\_windspeed = 'None', max\_windspeed = 'None') {

# This function makes the linear regression of the concentrations data depending on the flow.

# The purpose is to see if the data can be proportional (y = ax+b with b = 0)

# The function returns results of the linear regression (coefficient, p-value and r squared)

# The function also plots the result, showing the dataset and the regression results

# Gestion of the limit conditions:

reg\_DF = DF

if (max\_speedlimit != 'None') {

reg\_DF = DF[which(DF$Speed <= max\_speedlimit),]

}

if (min\_windspeed != 'None') {

reg\_DF = DF[which(DF$Speed >= min\_windspeed),]

}

if (min\_speedlimit != 'None') {

reg\_DF = DF[which(DF$Speed >= min\_speedlimit),]

}

if (max\_windspeed != 'None') {

reg\_DF = DF[which(DF$Speed <= max\_windspeed),]

}

XObs=reg\_DF$Flow

YObs=reg\_DF$NOx-reg\_DF$NOx\_bg

# Linear model : y = ax+b where b = 0

reg = summary(lm(YObs ~ 0 + XObs))

x11()

plot(XObs, YObs, ylab = "traffic NOx - background NOx (µg/m3)", xlab = "Flow (veh/h)",

main = "Pollution (NOx) depending on the flow",

sub = paste("R2 = ", reg$r.squared, "-- y = ", reg$coefficients[1]," \*x"))

lines(XObs, XObs\*reg$coefficients[1], col = "red")

# To plot the results

summary(lm(YObs ~ 0 + XObs))

}

1. Principal\_Component\_Analysis

Principal\_Component\_Analysis = function(DF) {

# This function does a PCA of the dataset, using its different variables.

install.packages("FactoMineR")

library(FactoMineR)

# PCA

result\_pca = PCA(DF[,c(2,3,4,6,7,8,9)], scale.unit = TRUE, graph = TRUE, quanti.sup = c(3,7))

x11()

plot.PCA(result\_pca, axes = c(1,2), choix ="var")

x11()

plot.PCA(result\_pca, axes = c(1,2), choix ="ind")

# Description of the dimensions

dimdesc(result\_pca, axes = 1:5)

}

1. plot\_boxplot

plot\_boxplot = function(DF, period, PH) {

# This functions draws boxplots of the concentrations depending on the speed, for

# a period (for instance 2 months) and at a precise peak hour

# There is a plot for each class of flow (each 1,000 veh/hour)

# The variables are:

# DF: same dataset with flow, speed, pollution and weather values.

# period : the description of the period, it will be used on the plot as a descrition

# the peak hour chosen. same as earlier, it will be used on the plot as a descrition

library('ggplot2')

library('gridExtra')

i = 0 # i: flow counter

flows\_list = list()

# The classes of flow are from 0 to 1,000 vehicles, from 1,000 to 2,000, etc...

# The flows\_list contains for each class of flow, the corresponding values from DF

while (nrow(DF[which(DF$Flow > i\*1000),]) != 0 ){

flows\_list[i+1] = list(i=DF[which((DF$Flow > i\*1000) & (DF$Flow < ((i+1)\*1000))),])

i=i+1

}

# creation of the list of plots

plot\_list = list()

for (j in 0:(length(flows\_list)-1)) {

# in order to plot the boxes, the speeds are put in categories of 20 km/h

flows\_list[[j+1]]$Speed\_categories = floor(flows\_list[[j+1]]$Speed/20)\*20+10

plot\_list[[j+1]] = ggplot(flows\_list[[j+1]], aes(x=Speed\_categories, y=NOx, group=Speed\_categories)) +

geom\_boxplot(width = 10) +

ggtitle(paste("Flow between ", j\*1000, " and", (j+1)\*1000, " veh/h")) +

scale\_x\_continuous(limits = c(0,100), breaks = seq(10,90,20)) +

scale\_y\_continuous(limits = c(0,500))

}

# Opening of the plotting window

x11(width = 16, height = 9)

n <- length(plot\_list)

nCol <- floor(sqrt(n))

nCol <- 3

do.call("grid.arrange", c(plot\_list, ncol=nCol,

top=paste("Concentrations of NOx depending on the speed, by flow classes : ", period, ", ", PH)))

}

1. Tool functions

date\_period\_selection = function(DF, beginning\_date, ending\_date){

# This function returns a new data frame extracted from the first one, containing only the data

# that are in between the two dates in argument.

# The format of the 2 dates are POSIXct. In order to create a POSIXct date, use the function

# as.POSIXct(date, format = ‘%d/%m/%Y %H:%M:%S’)

new\_DF = DF[which((DF$Date >= beginning\_date) & (DF$Date <= ending\_date)),]

return(new\_DF)

}

time\_period\_selection = function(DF, beginning\_hour, ending\_hour){

# This function returns a new data frame extracted from the first one, containing only the data

# in a specified time period.

# The format of the 2 hours are integers.

install.packages("lubridate")

library("lubridate")

if (beginning\_hour < ending\_hour) {

new\_DF = DF[which((hour(DF$Date) >= beginning\_hour) & (hour(DF$Date) <= ending\_hour)),]

}

if (beginning\_hour > ending\_hour) {

new\_DF = DF[which((hour(DF$Date) <= beginning\_hour) & (hour(DF$Date) >= ending\_hour)),]

}

return(new\_DF)

}