# Reproducible Research: Peer Assessment 1

*Michaël Hendrickx*

*March 17, 2017*

## Loading and preprocessing the data

Before starting, I assume that the **activity.zip** file has been downloaded and is located in the working directory.
If the raw data file is not present in the working directory, you can download the raw data file using the following link and then add it to your working directory :

- activity.zip

### Loading data

Once the dataset file is in the working directory, this dataset file is unzipped and read as csv. The content of the dataset is stored in the `data` variable.

```r
unzip("activity.zip")
data <- read.csv("activity.csv")
```

### Examining the data

Let's have a look at the content of theses data.

```r
str(data)
```

```
## 'data.frame':    17568 obs. of  3 variables:
##  $ steps   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ date    : Factor w/ 61 levels "2012-10-01","2012-10-02",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ interval: int  0 5 10 15 20 25 30 35 40 45 ...
```

```r
head(data, 15)
```

```
##    steps       date interval
## 1     NA 2012-10-01        0
## 2     NA 2012-10-01        5
## 3     NA 2012-10-01       10
## 4     NA 2012-10-01       15
## 5     NA 2012-10-01       20
## 6     NA 2012-10-01       25
## 7     NA 2012-10-01       30
## 8     NA 2012-10-01       35
## 9     NA 2012-10-01       40
## 10    NA 2012-10-01       45
## 11    NA 2012-10-01       50
## 12    NA 2012-10-01       55
## 13    NA 2012-10-01      100
## 14    NA 2012-10-01      105
## 15    NA 2012-10-01      110
```

The variables included in this dataset are:

- **steps**: Number of steps taking in a 5-minute interval (missing values are coded as NA)
- **date**: The date on which the measurement was taken in YYYY-MM-DD format
- **interval**: Identifier for the 5-minute interval in which measurement was taken

From the `str` and `head` commands, we can already see some problems regarding the data :

1. There are a number of `NA` values in the steps variable
2. The dates are represented as factors. The date will have to be converted to Date object in order to have proper representations of dates.
3. The identifier of the interval seems to represent the time where the interval begins in a format **HourMinute**. This should also be converted to timestamps in order to have proper representations of the intervals.

### 1. NA values

Which variables have NA values?

```
sapply(data, function(x) sum(is.na(x)))
```

```
##    steps     date interval 
##     2304        0        0
```

Only the variable steps have some NA values.
Let's check the ratio of NA values.

```
mean(is.na(data$steps))
```

```
## [1] 0.1311475
```

13% of the values are missing. This is not negligeable.
For the moment, observations with NA values will be ignored. Later on, an imputing stategy will be implemented in order to provide simulated values for all these NA.

### 2. Convert the dates format

Convert the date to POSIXct format and store the result in `data$date`.

```
data$date <- as.POSIXct(data$date, format="%Y-%m-%d")
```

### 3. Convert the intervals format

The 5-minutes intevals are stored in a **HourMinute** format representing the time of the start of the interval.
For example :

- **Interval value 5** : represents the 5-minutes interval starting at 00:05:00
- **Interval value 105** : represents the 5-minutes interval starting at 01:05:00
- **Interval value 2255** : represents the 5-minutes interval starting at 22:55:00

This representation should be converted to timestamps in order to have a proper representations of intervals.

```
convertIntervalToHMS <- function(x) {
    h <- x %/% 100      # Extract the hour of the startof the interval
    m <- x %% 100       # Extract the minute of the startof the interval
    sprintf("%02d:%02d:00", h, m)
}
data$intervalHMS <- convertIntervalToHMS(data$interval)
```

### Processed data

Let's have a look at the content of the processed data.

```
str(data)
```

```
## 'data.frame':    17568 obs. of  4 variables:
##  $ steps      : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ date       : POSIXct, format: "2012-10-01" "2012-10-01" ...
##  $ interval   : int  0 5 10 15 20 25 30 35 40 45 ...
##  $ intervalHMS: chr  "00:00:00" "00:05:00" "00:10:00" "00:15:00" ...
```

```
head(data, 15)
```

```
##    steps       date interval intervalHMS
## 1     NA 2012-10-01        0    00:00:00
## 2     NA 2012-10-01        5    00:05:00
## 3     NA 2012-10-01       10    00:10:00
## 4     NA 2012-10-01       15    00:15:00
## 5     NA 2012-10-01       20    00:20:00
## 6     NA 2012-10-01       25    00:25:00
## 7     NA 2012-10-01       30    00:30:00
## 8     NA 2012-10-01       35    00:35:00
## 9     NA 2012-10-01       40    00:40:00
## 10    NA 2012-10-01       45    00:45:00
## 11    NA 2012-10-01       50    00:50:00
## 12    NA 2012-10-01       55    00:55:00
## 13    NA 2012-10-01      100    01:00:00
## 14    NA 2012-10-01      105    01:05:00
## 15    NA 2012-10-01      110    01:10:00
```

The date is now a correct date, the intervals are not represented as timestamp.

## What is mean total number of steps taken per day?

First, the total of steps per day is computed by filtering the `NA`values, grouping the values by date and counting the total of steps per day.
`stepsPerDay` contain the total of steps for each day.

```
library(dplyr)
stepsPerDay <- data %>%
    filter(! is.na(steps)) %>%
    group_by(date) %>%
    summarize(stepsPerDay = sum(steps))
head(stepsPerDay)
```

```
## # A tibble: 6 × 2
##         date stepsPerDay
##       <dttm>       <int>
## 1 2012-10-02         126
## 2 2012-10-03       11352
## 3 2012-10-04       12116
## 4 2012-10-05       13294
## 5 2012-10-06       15420
## 6 2012-10-07       11015
```

The mean total number of steps taken per day is the mean of `stepsPerDay` :

```r
meanStepsPerDay <- mean(stepsPerDay$stepsPerDay)
meanStepsPerDay
```

```
## [1] 10766.19
```

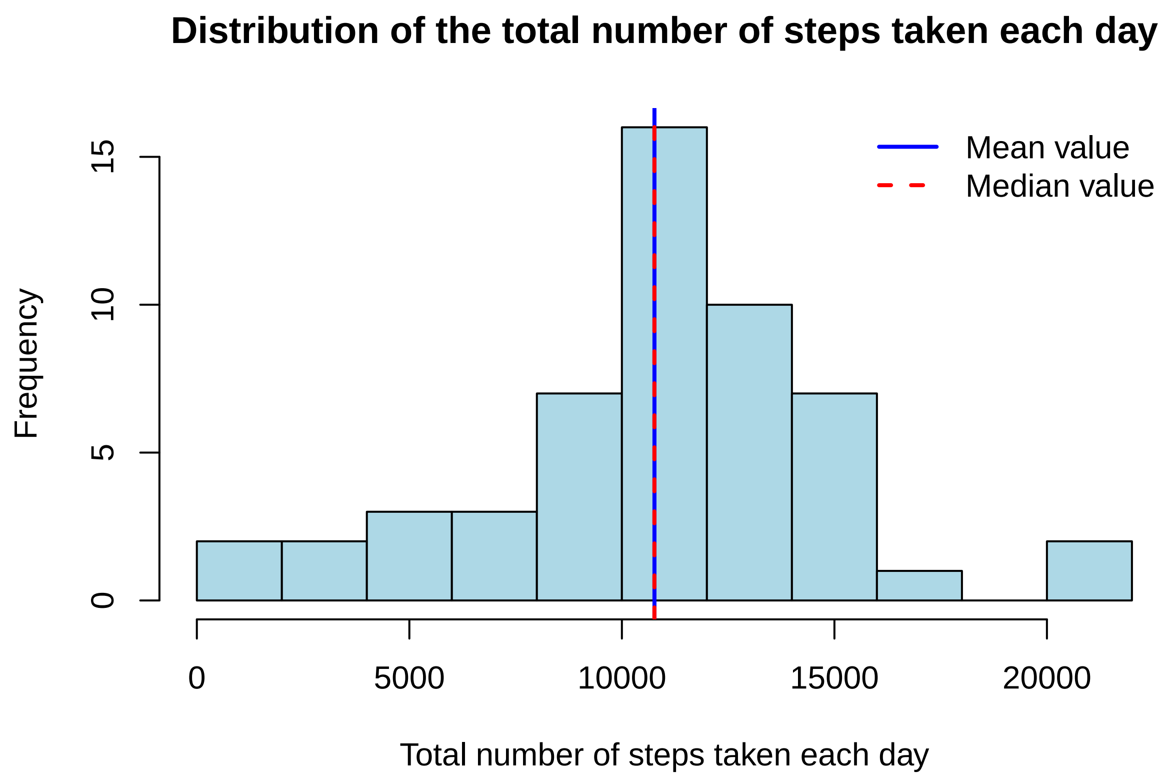
The median total number of steps taken per day is the median of `stepsPerDay` :

```r
medianStepsPerDay <- median(stepsPerDay$stepsPerDay)
medianStepsPerDay
```

```
## [1] 10765
```

The mean and median are really similar.
The following graph displays the distribution of the number of steps per day, along with the mean and median values.

```r
hist(stepsPerDay$stepsPerDay, breaks = 10, col = "lightblue",
     main = "Distribution of the total number of steps taken each day",
     xlab = "Total number of steps taken each day")
abline(v=meanStepsPerDay, lwd=2, col="blue")
abline(v=medianStepsPerDay, lty=2, lwd=2, col="red")
legend("topright", lwd=2, lty=c(1,2), col=c("blue", "red"),
       legend=c("Mean value", "Median value"),
       bty="n")
```



## What is the average daily activity pattern?

In order to get the average daily activity pattern, we first compute the number of steps per 5-minutes time interval for all the days present in the dataset. To do so, the `NA` values are removed, the data are grouped by time intervals and the mean number of steps is computed.

```
meanStepsPerInterval <- data %>%
    filter(! is.na(steps)) %>%
    group_by(intervalHMS) %>%
    summarize(meanStepsPerInterval = mean(steps))
```

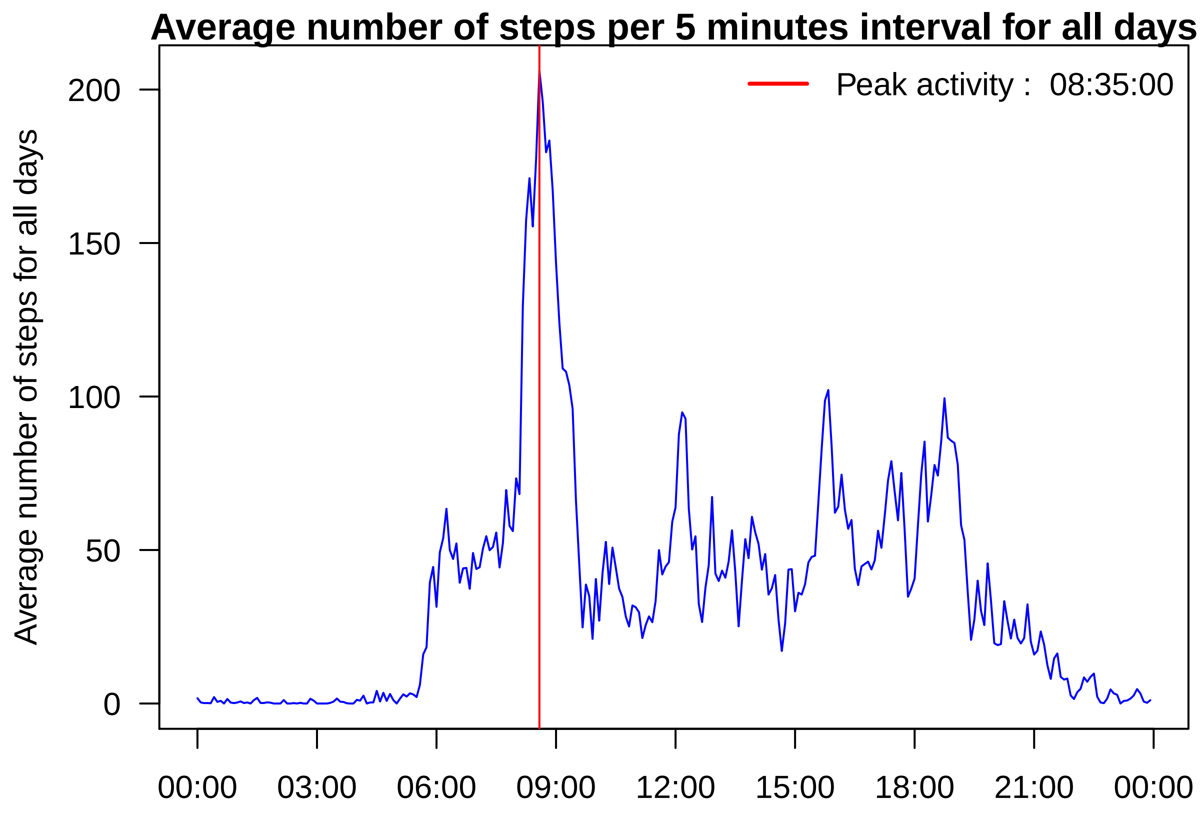
`meanStepsPerInterval` contains the mean number of steps per time interval. When does the mean number of steps is maximal?

```
intervalWithMaxMeanSteps <- meanStepsPerInterval$intervalHMS[
    which.max(meanStepsPerInterval$meanStepsPerInterval)]
intervalWithMaxMeanSteps
```

```
## [1] "08:35:00"
```

The monitored person walks the most between 8:35 and 8:40.
The following diagram represents the mean daily activity for all the observations. In order to plot the time evolution, the interval are first transformed into POSIXct.

```
par(mar=c(0.6, 5.1, 1, 0.6), oma=c(2.1, 0, 1, 0))
plot(as.POSIXct(strptime(meanStepsPerInterval$intervalHMS, "%H:%M:%S")),
     meanStepsPerInterval$meanStepsPerInterval,
     type="l",
     col = "blue",
     main = "Average number of steps per 5 minutes interval for all days",
     ylab= "Average number of steps for all days",
     xlab = "Period of the day",
     las = 2,
     xaxt='n'
)
ticks <- pretty(as.POSIXct(strptime(meanStepsPerInterval$intervalHMS, "%H:%M:%S")), n = 12)
lbl <- strftime(ticks, '%H:%M')
axis(side=1, outer=FALSE, at=ticks, labels=lbl)
abline(v=as.POSIXct(strptime(intervalWithMaxMeanSteps, "%H:%M:%S")), col="red", lwd=1)
legend("topright", lwd=2, lty=1, col="red",
       legend=c(paste("Peak activity : ", intervalWithMaxMeanSteps)),
       bty="n")
```



## Imputing missing values

We saw earlier that 13% of the observations have a `NA` value. Here we'll implement an imputing method for replacing those `NA` values and see if it impacts the distribution of steps per day.
The imputing method chosen consists of replacing a `NA` number of steps by the mean number of steps for the same interval. Remember that `meanStepsPerInterval` contains the mean number of steps per interval. The result will be stored in the `dataImputed` variable.

```
dataImputed <- data %>%
        merge(meanStepsPerInterval, "intervalHMS") %>%
        mutate(steps = ifelse(is.na(steps), meanStepsPerInterval, steps)) %>%
        select(-meanStepsPerInterval)
```

```
    stepsPerDayImputed <- dataImputed %>%
        filter(! is.na(steps)) %>%
        group_by(date) %>%
        summarize(stepsPerDay = sum(steps))
```

The mean total number of steps taken per day is the mean of `stepsPerDay` :

```
meanStepsPerDayImputed <- mean(stepsPerDayImputed$stepsPerDay)
meanStepsPerDayImputed
```

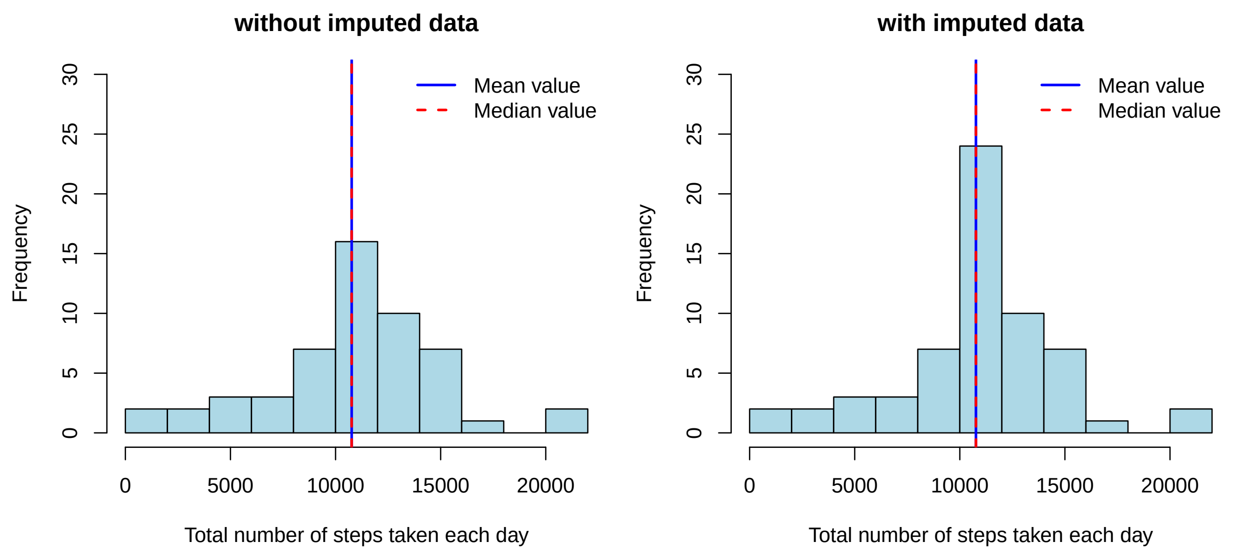
```
## [1] 10766.19
```

The median total number of steps taken per day is the median of `stepsPerDay` :

```r
medianStepsPerDayImputed <- median(stepsPerDayImputed$stepsPerDay)
medianStepsPerDayImputed
```

```
## [1] 10766.19
```

```r
par(mfrow=c(1,2), mar=c(4,4,3,1))
# Plot without imputing
hist(stepsPerDay$stepsPerDay, breaks = 10, col = "lightblue",
     main = "without imputed data",
     xlab = "Total number of steps taken each day",
     ylim = c(0,30))
abline(v=meanStepsPerDay, lwd=2, col="blue")
abline(v=medianStepsPerDay, lty=2, lwd=2, col="red")
legend("topright", lwd=2, lty=c(1,2), col=c("blue", "red"),
       legend=c("Mean value", "Median value"),
       bty="n")
# Plot with imputing
hist(stepsPerDayImputed$stepsPerDay, breaks = 10, col = "lightblue",
     main = "with imputed data",
     xlab = "Total number of steps taken each day",
     ylim =  c(0,30))
abline(v=meanStepsPerDayImputed, lwd=2, col="blue")
abline(v=medianStepsPerDayImputed, lty=2, lwd=2, col="red")
legend("topright", lwd=2, lty=c(1,2), col=c("blue", "red"),
       legend=c("Mean value", "Median value"),
       bty="n")
```



There is no difference in the mean value of the steps per day of both datasets.
This is quite expected since the imputed the `NA` values with the mean of the corresponding interval. The effect of the data imputation is that there are more occurences of the mean value in the distribution. The peak of the mean value block is slightly under 25 with the imputated data, while it is slightly above 15 without imputed data. The distribution of the other number of steps is virtually unchanged.

So imputing the data reinforces the mean value of the dataset.

## Are there differences in activity patterns between weekdays and weekends?

First, add a factor indicating the type of day : “weekday” or “weekend”.

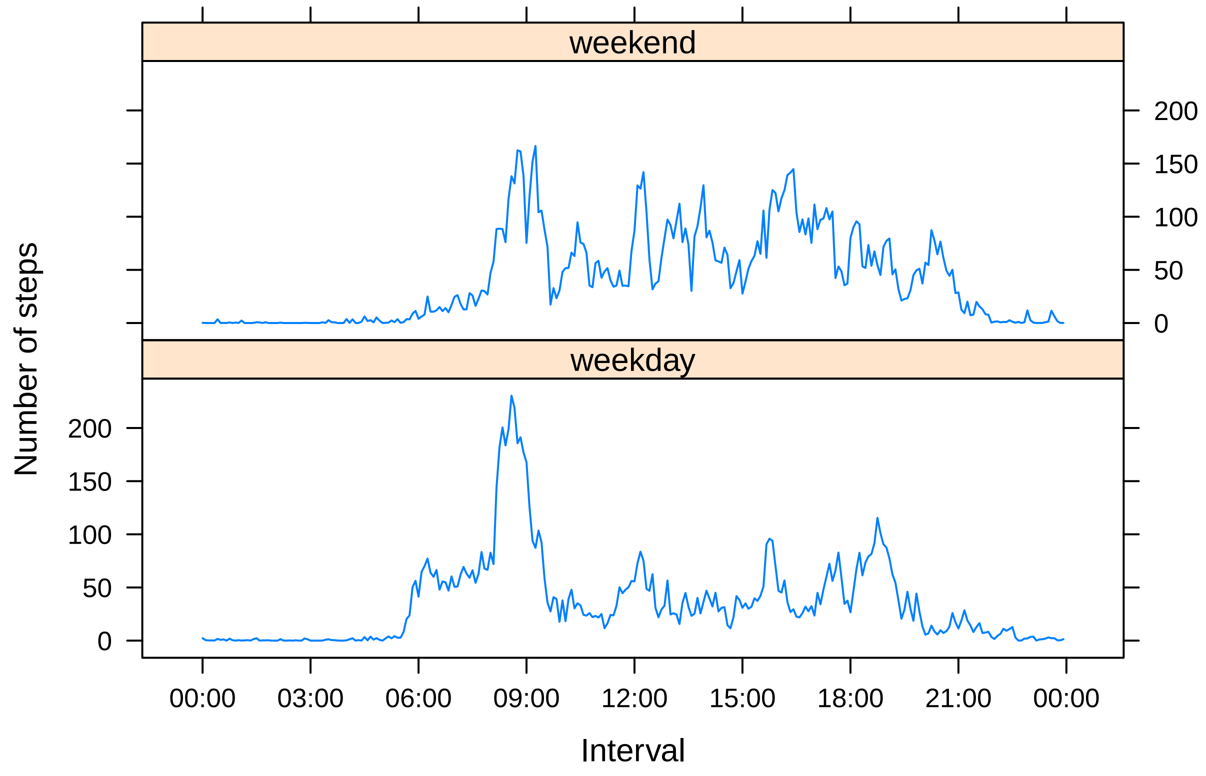
```
meanStepsPerIntervalDayType <- dataImputed %>%
    mutate(dayType =  ifelse(weekdays(date) %in% c("Sunday", "Saturday"),
                            "weekend", "weekday") ) %>%
    mutate(dayType = factor(dayType))
```

Then, group the data by interval and type of day. Compute the mean number of steps per interval and per type of day.

```
meanStepsPerIntervalDayType <- meanStepsPerIntervalDayType %>%
    mutate(intervalHMS = as.POSIXct(strptime(intervalHMS, "%H:%M:%S"))) %>%
    group_by(intervalHMS, dayType) %>%
    summarize(meanStepsPerInterval = mean(steps))
```

Create a plot with the mean number of steps per interval in a weekday and on a weekend.

```
library(lattice)
xyplot(meanStepsPerInterval ~ intervalHMS | dayType,
       data=meanStepsPerIntervalDayType, type="l",
       xlab = "Interval", ylab = "Number of steps",
       layout = c(1, 2),
       scales=list (x=list(tick.number=15, cex=0.8) )
)
```


weekend
weekday
Number of steps
00:00
03:00
06:00
09:00
12:00
15:00
18:00
21:00
00:00
Interval