Reproducible Research: Peer Assessment 1

Carlos Artilez

May 10, 2018

0.- Loading and preprocessing the data

First, in loading the data for this Peer Assessment Project, we proceed to use the *read.csv()* function unto the designated .csv file. To assess the class types within this data set, we use the str() function.

```
activity <- read.csv("activity.csv")
str(activity)
```

```
## '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 ...
```

We can see that the class types are integer and factor, the latter being very inconvenient for further analysis. We proceed to re-load the data adding the colClasses option in order to specify a numeric, Date and numeric columns, respectively.

```
 activity <- read.csv("activity.csv", colClasses = c("numeric", "Date", "numeric"))
str(activity)
```

```
## 'data.frame':    17568 obs. of  3 variables:
##  $ steps   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ date    : Date, format: "2012-10-01" "2012-10-01" ...
##  $ interval: num  0 5 10 15 20 25 30 35 40 45 ...
```

Now the new loaded data set is suitable for analysis, and we are ready to answer the assigned questions.

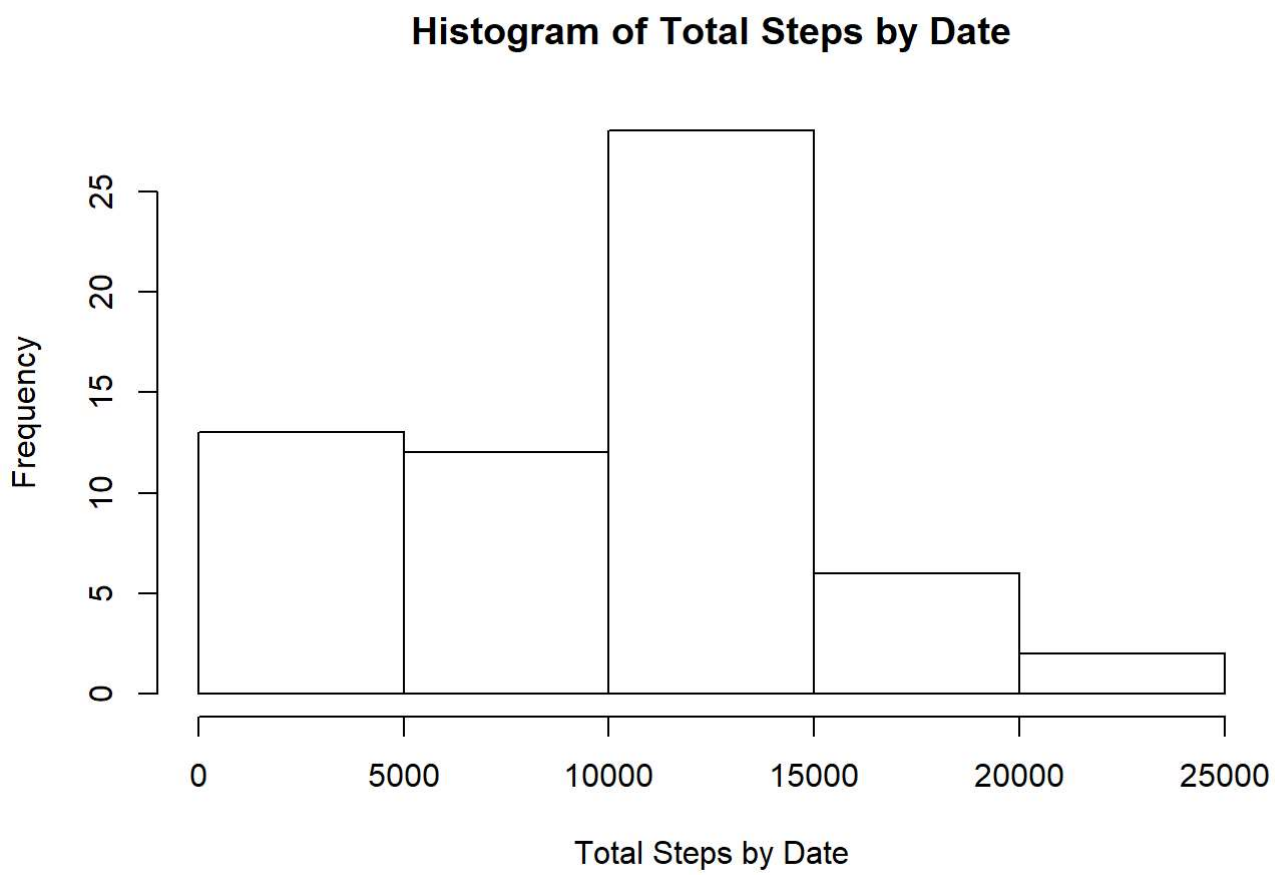
1.- What is mean total number of steps taken per day?

To answer this question, first we use the *tapply()* function to calculate the sum of the activities per day.

```
sumByDate <- tapply(activity$steps, activity$date, sum, na.rm = TRUE)
```

Then, we can create a histogram plot for the total number of steps taken per day:

```
hist(sumByDate, xlab = "Total Steps by Date", main = "Histogram of Total Steps by Date")
```



We can see that the large number of missing values (NAs), produce some skewness to towards the left side of the distribution of what appears to be a gaussian behavior. On the other hand, getting a quick summary of the new array can also help us see the mean and median of the total number of steps taken per day.

```
summary(sumByDate)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0    6778   10395    9354   12811   21194
```

This summary can help us respond to this first question.

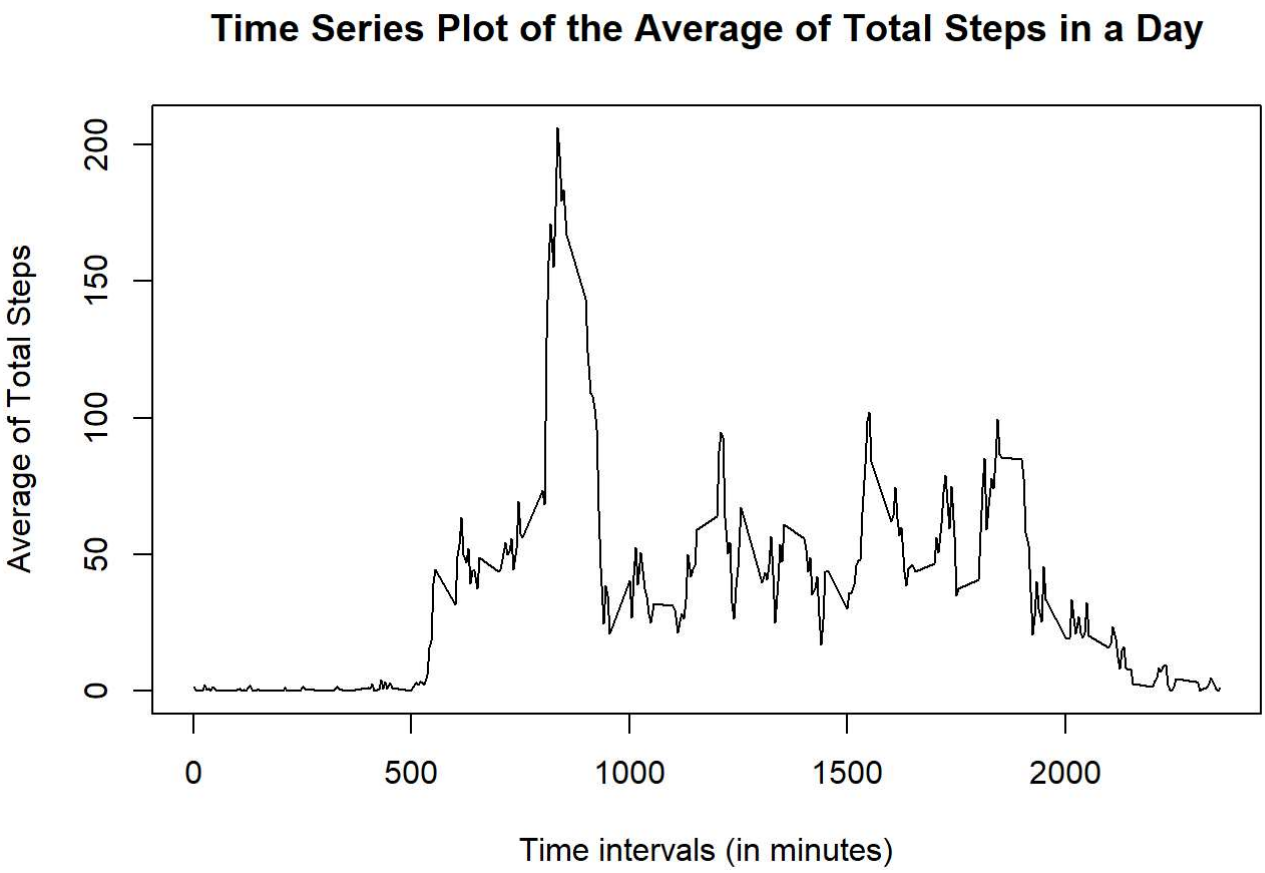
2.- What is the average daily activity pattern?

One good way to respond is creating a time series plot of the 5-minute interval (x-axis) and the average number of steps taken, averaged across all days (y-axis). But first, we use *tapply()* to calculate the mean of the steps per time interval.

```
meanByInterval <- tapply(activity$steps, activity$interval, mean, na.rm = TRUE)
```

With this new array we can create the time series plot as follows:

```
plot(row.names(meanByInterval), meanByInterval, type = "l", xlab = "Time intervals (in minutes)"
, ylab = "Average of Total Steps", main = "Time Series Plot of the Average of Total Steps in a D
ay")
```



We can see which 5-minute interval, on average across all the days in the data set, contains the maximum number of steps. Nevertheless, we can also obtain the specific interval with the maximum number of steps by applying the *max()* funcition:

```
max <- max(meanByInterval)
max_int <- match(max, meanByInterval)
meanByInterval[max_int]
```

```
##      835
## 206.1698
```

Both, the time series plot and this last important data point, can help us understand the average daily activity pattern.

3.- Imputing missing values

We can note that there are a number of days/intervals where there are missing values (coded as NA). The presence of missing days may introduce bias into some calculations or summaries of the data. Therefore, first, we had to calculate and report the total number of missing values in the data set (i.e. the total number of rows with NAs).

```
sum(is.na(activity))
```

```
## [1] 2304
```

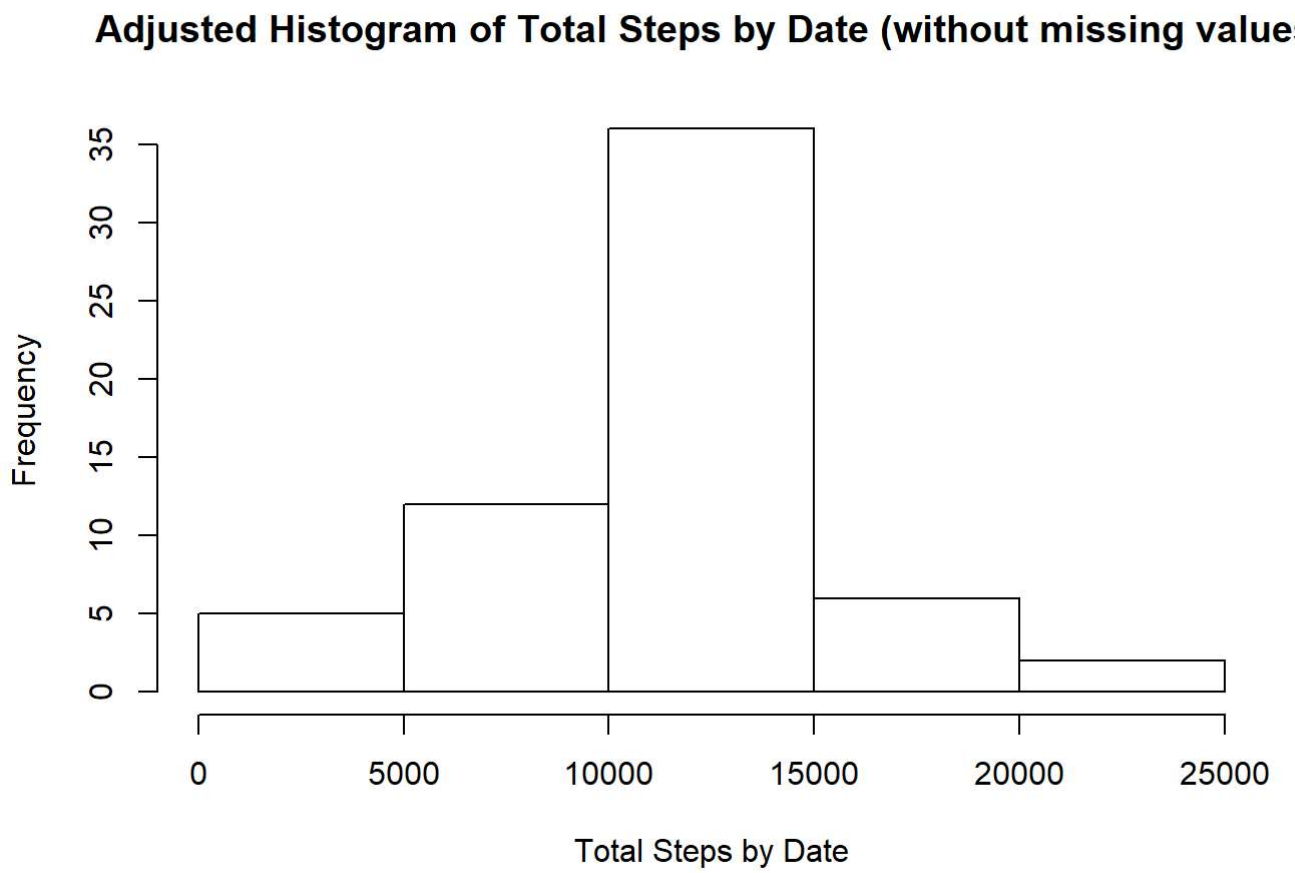
Depending on the type of analysis that we need to apply, this might be quite harmful. So, we devised a non-sophisticated strategy for filling in all of the missing values in the data set, using the mean for the corresponding 5-minute interval.

To accomplish this task, we created two separate data frames: one without NAs and one with all rows with NA values for the steps vector. Then, inserting the *meanByInterval* vector (created in the previous question) into the data frame that contains all NAs for the steps variable, we can proceed to bind the data frame with no NAs with the data frame that now contains all of the mean values for the corresponding time intervals. Finally, a new data set is created, equal to the original data set but with the missing data filled in.

```
activityNAs <- activity[is.na(activity), ]
activityNoNAs <- activity[complete.cases(activity), ]
activityNAs$steps <- as.numeric(meanByInterval)
completedActivity <- rbind(activityNAs,activityNoNAs)
completedActivity <- completedActivity[order(completedActivity[ , 2], completedActivity[ , 3]),
 ]
```

Now we can proceed to create a histogram of the total number of steps taken each day, and calculate and report the mean and median total number of steps taken per day.

```
completedsumByDate <- tapply(completedActivity$steps, completedActivity$date, sum)
hist(completedsumByDate, xlab = "Total Steps by Date", main = "Adjusted Histogram of Total Steps
 by Date (without missing values)")
```



We can see a subtle but important change in the behavior of the distribution. The skewness seen in the previous histogram (refer to section 1) was adjusted, further showing how it fits on a Normal Distribution curve. This would allow researchers to get a better representation of the amount of steps that are taken during specific intervals during the day, especially for prediction purposes.

Furthermore, we compute a summary of this new completed data set, where an important change also happened:

```
summary(completedsumByDate)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      41    9819   10766   10766   12811   21194
```

We can see that both the mean and the median have the same value. This could confirm that the adjustment made to the missing values further contributed on fitting the measured behavior into a Normal Distribution.

4.- Are there differences in activity patterns between weekdays and weekends?

To answer this question, we have to create a new factor variable -in the new completed data set- with two levels: “weekday” and “weekend”; we proceed to do this by indicating whether a given date is either a weekday or a weekend day.

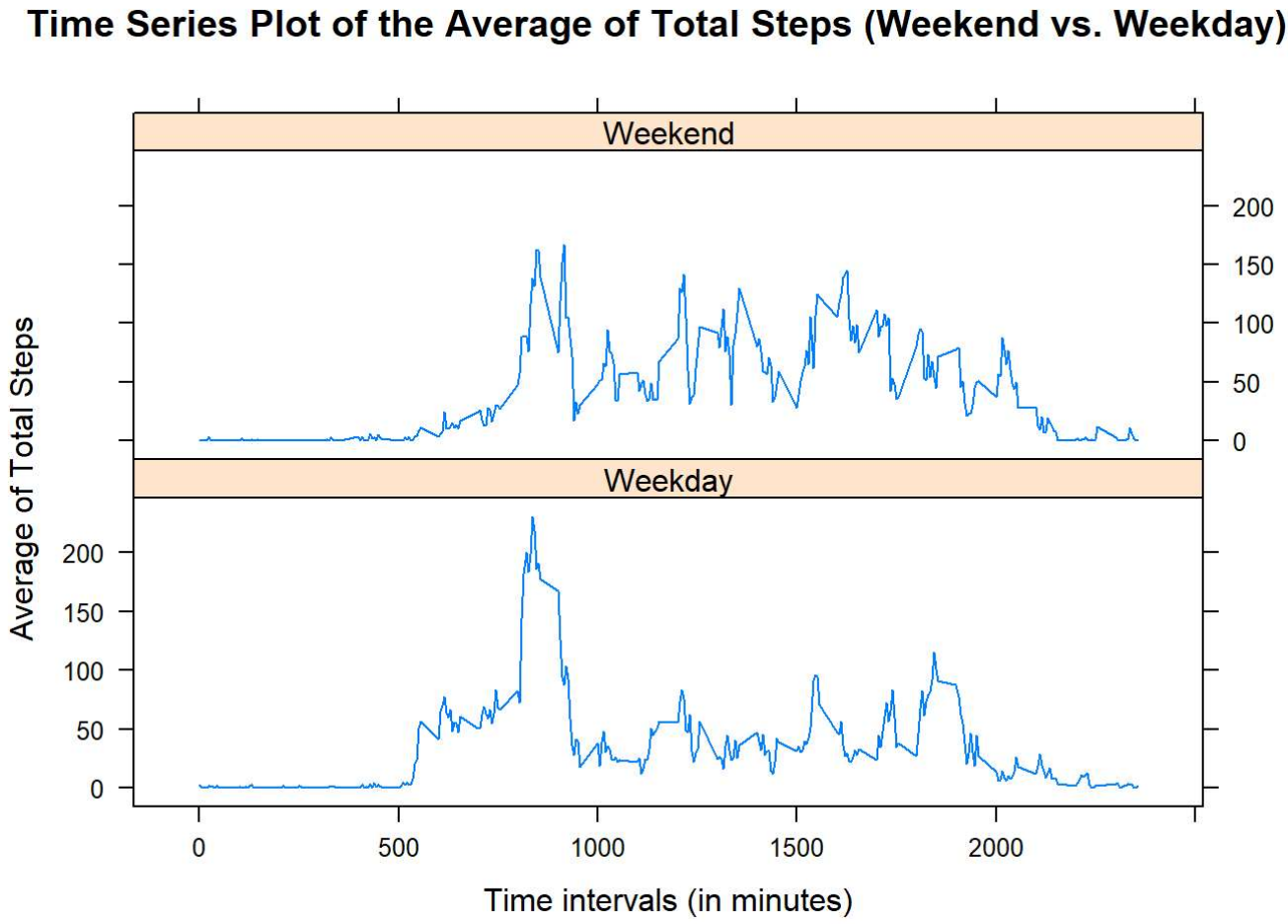
```
days <- weekdays(completedActivity[ , 2])
completedActivity <- cbind(completedActivity, days)
library(plyr)
completedActivity$days <- revalue(completedActivity$days, c("Monday" = "Weekday", "Tuesday" = "W
eekday","Wednesday" = "Weekday", "Thursday" = "Weekday", "Friday"="Weekday", "Saturday" = "Weeke
nd", "Sunday"= "Weekend"))
```

With this new data set, we proceed to create a panel plot containing a time series plot of the 5-minute interval (x-axis) and the average number of steps taken, averaged across all weekday days or weekend days (y-axis). To do this, we need first to create a new data set with the average number of steps taken, averaged across both day levels.

```
newMeanByInterval <- tapply(completedActivity$steps, list(completedActivity$interval, completedA
ctivity$days), mean)
library(reshape2)
newMeanByInterval <- melt(newMeanByInterval)
colnames(newMeanByInterval) <- c("interval", "day", "steps")
```

With this new data set, we proceed to create the necessary time series plot of the mean total steps by the intervals for the weekday and the weekend.

```
library(lattice)
xyplot(newMeanByInterval$steps ~ newMeanByInterval$interval | newMeanByInterval$day, layout = c(
1, 2), type = "l", main = "Time Series Plot of the Average of Total Steps (Weekend vs. Weekday)"
, xlab = "Time intervals (in minutes)", ylab = "Average of Total Steps")
```



With this plot we can see that, in the weekend, all activity (average number of steps) is more distributed between (more or less) the 835 and 2000 time intervals; while in the weekdays, all activity tends to be more concentrated between the 500 and 1000 time intervals.