Eduardo B. Diez

Professors Roger D. Peng, Jeff Leek, and Brian Caffo

Reproducible Research

13 July 2014

Reproducible Research: Peer Assessment 1

"…walker, there is no path, you make the path as you go . . ."

Antonio Machado (1875 to 1939)

Introduction

This entire assignment is done in a single R markdown document answering all questions stated.

The plotting of this assignment used the base plotting system but only one exception was made for the last question, in order to match exactly the example given in the instructions file and so lattice package is used for this part.

This file is allocated in my GitHub repository inside the forked http://github.com/rdpeng/RepData_PeerAssessment1 done as indicated the instructions. My repository and the SHA-1 commit ID for it are the submission to do.

We also have commented the code and it's part of the documentation.

Loading and Preprocessing the data

All files should be allocated in one directory. The first is to keep the actual directory to restore it at the end of the session, set the working directory and load the file.

Setting the working directory and loading the data

```
# save the actual directory to restore it at the end of the sesion
curdir <- getwd()

# set the pointer to the working directory where the original
# dataset is alocated. Change it to fit your particular setting
workingdirectory <- "D:/Cursos/Hopkin/5-Reproducible Research/Project 1/submission"

# set the new working directory
setwd(workingdirectory)

# load the csv file
mydata <- read.csv("activity.csv")
```

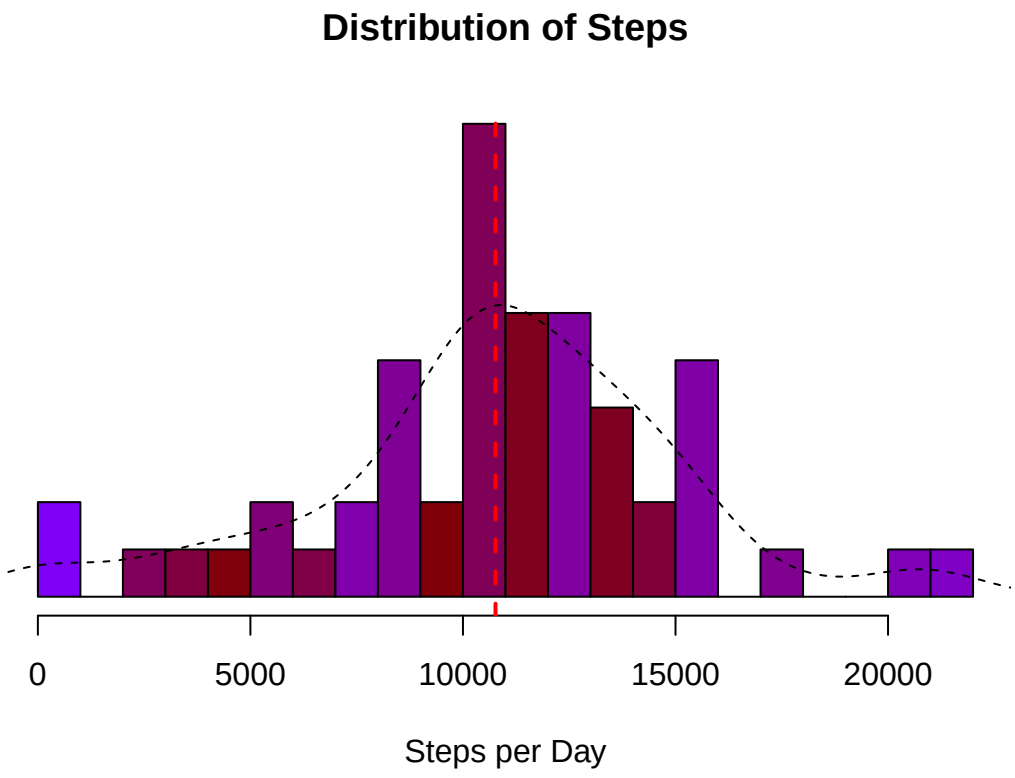
Preprocessing the data The only general preprocessing is to change from the original factor type of the date variable to date type because we prefer it. The conditioning of the data will be done each time we require it.

```
# it's prefered the date type as date type and not factor (...)
mydata$date <- as.Date(mydata$date, "%Y-%m-%d")
```

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

Histogram of the total number of steps taken each day We start making the histogram of the total number of steps taken each day

```
# make the plot with base package

# get the x to be ploted
x <- aggregate(steps ~ date, mydata, sum)
x <- x$steps

# calculate psychedelic colors to avoid a boring histogram with only one color
relative.steps <- rank(x) / length(x)
mypsycos <- rgb(1 - relative.steps, red=.5, green=0)

# plot it
hist(x, col=mypsycos,
     main = "Distribution of Steps",
     xlab = "Steps per Day",
     breaks=21,
     prob=T,
     yaxt="n",
     ylab="")

# add a density line
lines(density(x,
              na.rm=TRUE),
      col="black",
      lty=2)

# add a vertical line at the mean position
abline(v = mean(x,na.rm=TRUE),
       lty = 2,
       lwd=2,
       col = "red" )
```



The mean and the median Follows to calculate and report the mean and median total number of steps taken per day.

```
# the mean
mean(x, na.rm = TRUE)
```

```
## [1] 10766
```

```
# the median
median(x, na.rm = TRUE)
```

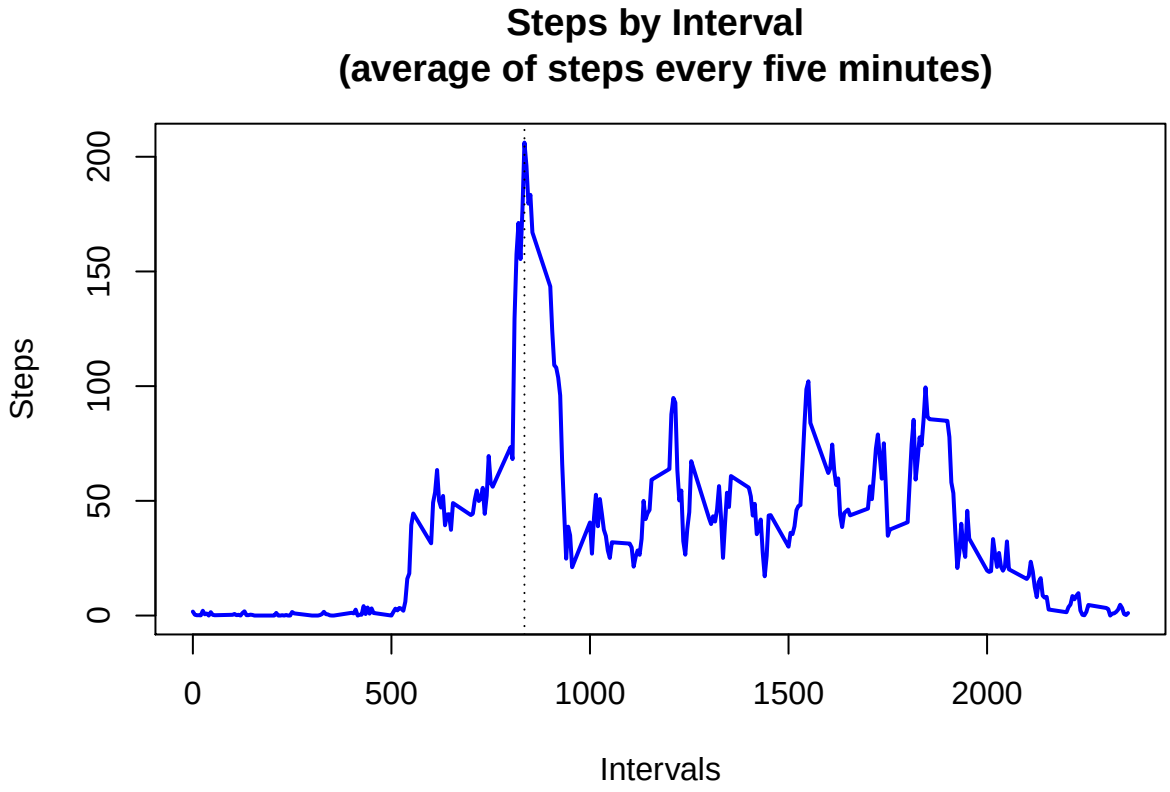
```
## [1] 10765
```

3. The average daily activity

Time series plot At this point we make a time series plot of the intervals in the x-axis and the average number of steps taken, averaged across all days, at the y-axis.

```
# get the x and the y to be plotted
tmp <- aggregate(steps ~ interval, mydata, mean)
x <- tmp$interval
y <- tmp$step
```

```r
# plot y as a funtion of x
plot(y ~ x,
     type = "l",
     col = "blue",
      lwd = 2,
     main = paste("Steps by Interval",
                  "\n",
                  "(average of steps every five minutes)"),
     xlab = "Intervals",
     ylab = "Steps"
     )

# show a vertical line at the max value of the ploted line,
# looking for the next answer to be given (...)
abline(v = x[which(y == max(y))],
       lty = 3,
       col = "black" )
```



Maximum number of steps If we worder about which 5-minute interval, on average across all the days in the data set, contains the maximum number of steps; we can give the answer as follow;

```r
# find the x (the number of the interval) in which y (mean of step) is a maximum
x[which(y == max(y))]
```

```
## [1] 835
```

```r
# that contain a maximum of
max(y)
```

```
## [1] 206.2
```

4. Imputing missing values

Now is the turn to create a new data set that is equal to the original data set but with the missing data filled in. After we get the data set so conformed, we make 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.

Missing values in dataset The total number of missing values in the data set id given by:

```r
sum(is.na(mydata$steps))
```

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

Strategy for filling in missing values The strategy for filling in all of the missing values in the data set will be to use the mean for that day. But before we need to adequate the data in order to do the calculation and then save them in the original variable that has NAs values.

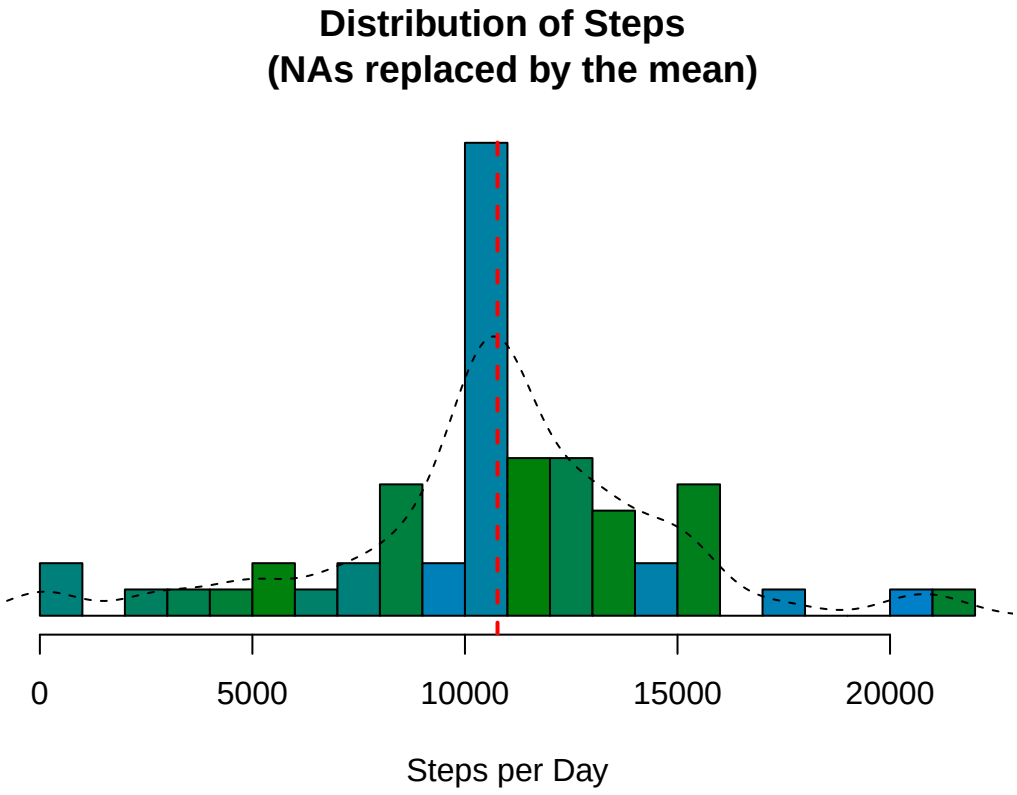
```r
# merge mydata and the mean of steps by interval
tmp <- aggregate(steps ~ interval, mydata, mean)
mydata.merged <- merge(mydata, tmp, by = "interval", sort = FALSE)

#change odd variable names after the merged because was duplicated
names(mydata.merged) <- c("interval", "steps",  "date", "mean.steps")

# sort it
mydata.merged <- mydata.merged[with(mydata.merged, order(date, interval)), ]

# replace the NAs values in steps variable in mydata with the new mean values
mydata.merged$steps[is.na(mydata.merged$steps)] <- mydata.merged$mean.steps[is.na(mydata.merged$steps)]

# remove the column that is no longer needed
mydata.merged$mean.steps <- NULL
```

The New Dataset It is stated to create a new data set that is equal to the original data set but with the missing data filled in, as we have done above and we are going to copy it as *mydata.new*.

```r
# copy it as mydata.new and change the variables order to match the one in mydata
mydata.new <- mydata.merged[, c(2, 3, 1)]
```

Histogram of total number of steps Now, we make a new 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.

```r
# get the x to be ploted. As the first histogram but from mydata.new
x <- aggregate(steps ~ date, mydata.new, sum)
x <- x$steps

# calculate psychedelic colors to avoid a boring histogram with only one color
relative.steps <- rank(x) / length(x)
mypsycos <- rgb(1 - relative.steps, red=0, green=0.5)

# plot it
hist(x, col=mypsycos,
     main = paste("Distribution of Steps",
                  "\n",
                  "(NAs replaced by the mean)"),
     xlab = "Steps per Day",
     breaks=21,
     prob=T,
     yaxt="n",
     ylab="")

# add a density line
lines(density(x,
              na.rm=TRUE),
      col="black",
      lty=2)

# add a vertical line at the mean position
abline(v = mean(x,na.rm=TRUE),
       lty = 2,
       lwd=2,
       col = "red" )
```



One more time, we calculate the mean and median of the total number of steps taken per day in order to compare them with the previous we got.

Mean number of steps per day:

```
# calculate the mean
mean(x, na.rm = TRUE)
```

```
## [1] 10766
```

Median number of steps per day:

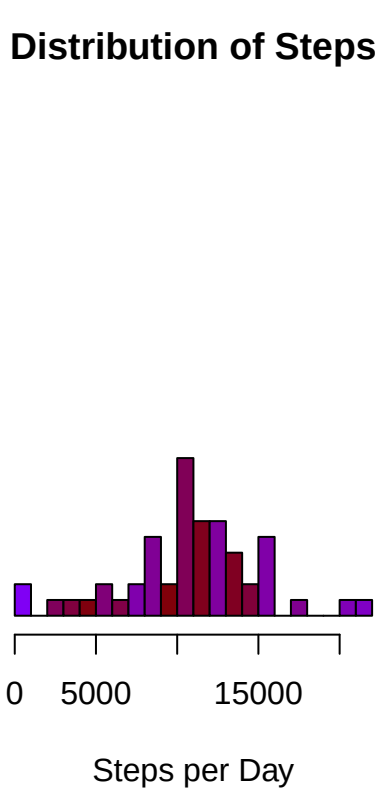
```
# and the meadian
median(x, na.rm = TRUE)
```

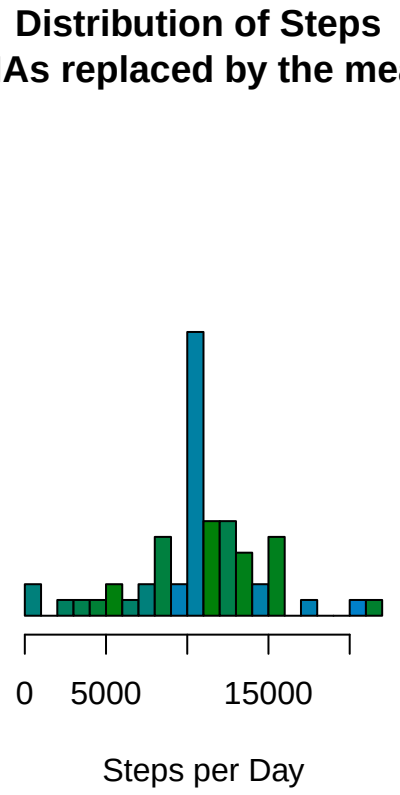
```
## [1] 10766
```

The Mean is equal to the estimates from the first part of the assignment. The Median is roughtly the same to the one in the first part of the assignment. Really, the difference between the before and the after is just one unit in the median. Basically the are roughly the same. Therefore there is not impact when imputing missing data in this particular case.

We can give a look to both histograms, side by side:

```r
# set the output device as an array with 1 column and two rows
par(mfrow = c(1, 2))

# make the first plot with base package as the first histogram showed.
# get the x to be ploted
x <- aggregate(steps ~ date, mydata, sum)
x <- x$steps

# calculate psychedelic colors to avoid a boring histogram with only one color
relative.steps <- rank(x) / length(x)
mypsycos <- rgb(1 - relative.steps, red=.5, green=0)

# plot it
hist(x, col=mypsycos,
     main = "Distribution of Steps",
     xlab = "Steps per Day",
     breaks=21,
     prob=F,
     yaxt="n",
     ylab="",
     ylim=c(0,30))


# add the second
# get the x to be ploted. As the first histogram but from mydata.new
x <- aggregate(steps ~ date, mydata.new, sum)
x <- x$steps

# calculate psychedelic colors to avoid a boring histogram with only one color
relative.steps <- rank(x) / length(x)
mypsycos <- rgb(1 - relative.steps, red=0, green=0.5)

# plot it
hist(x, col=mypsycos,
     main = paste("Distribution of Steps",
                  "\n",
                  "(NAs replaced by the mean)"),
     xlab = "Steps per Day",
     breaks=21,
     prob=F,
     yaxt="n",
     ylab="",
     ylim=c(0,30))
```





5. Weekdays and Weekend

Now it's time to create a new factor variable in the data set with two levels - "weekday" and "weekend" indicating whether a given date is a weekday or weekend day. Then, we'll make a panel plot containing a time series plot of the 5-minute interval vs. the average number of steps taken, averaged across all weekday days or weekend days. It's mandatory to alert that we have done all plots until now by using the base system but the instruction gave us an examples that we are going to follow as close as we can.

```
# make a backup to work with to make this panel
mydatapanel <- mydata.new

# create a factor with the names of the days for all days
mydatapanel$weekdays <- factor(format(mydatapanel$date, "%A"))

# the names (...)
levels(mydatapanel$weekdays)
```

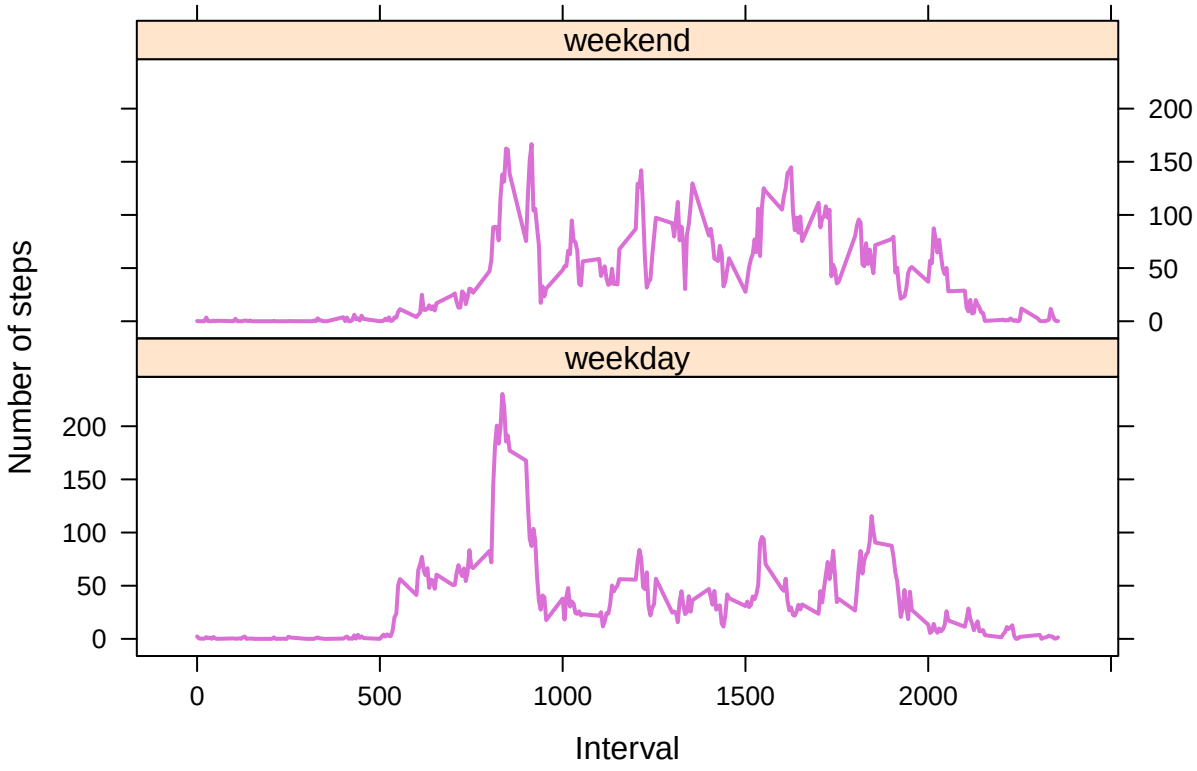
```
## [1] "Friday"    "Monday"    "Saturday"  "Sunday"    "Thursday"  "Tuesday"
## [7] "Wednesday"
```

```
# replace the levels with what we want
levels(mydatapanel$weekdays) <- list(weekday = c("Monday",
                                                 "Tuesday",
                                                 "Wednesday",
```

```
                                                  "Thursday",
                                                  "Friday"),
                                        weekend = c("Saturday",
                                                  "Sunday"))
```

Now, we make a panel plot containing a time series plot of the 5-minute interval at x-axis and the average number of steps taken, averaged across all weekday days or weekend days at y-axis. The plot is done by using the lattice library to get a look like the one in the instructions:

```
# get the x's and y's value to be plotted
tmp <- aggregate(mydatapanel$steps,
                 by = list(mydatapanel$weekdays,
                           mydatapanel$interval),
                 mean)

# remove groupes names and set nice one's
names(tmp) <- c("weekday","interval","steps")

# and get them pretty
y <- tmp$steps
x <- tmp$interval
cond <- tmp$weekday

# plot it with lattice to match the example given in the instructions
library(lattice)

# make the panel
xyplot(y ~ x | cond ,
       layout = c(1, 2),
       type = "l",
       lwd=2,
       col= "orchid",
       xlab = "Interval",
       ylab = "Number of steps")
```




```
# To end, we restore the working directory
setwd(curdir)

# end.
```