Reproducible Research: Assigment 1

Matthias S

Wednesday, September 16, 2015

Contents

1 Introduction

This report is written to fulfill assignemnt 1 of the Reproducible Research course. The data and assignemnt was downloaded from Pr Roger D. Peng's Github repository.

2 Useful ressources

Following ressources were used to write this report:

- R Markdown v2;
- R Markdown Reference Guide;
- R Markdown Cheat Sheet;
- And lots of Googling ...

3 Loading and preprocessing the data

This report is configured in such a way that the directory containing it can be downloaded from my Gihub repository onto any folder where the code will be available along the compressed data used to produce the report.

Reading the data:

```
unzip("activity.zip")
stepsdate <- read.csv("activity.csv")
str(stepsdate)
```

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

The date in the data above is in factor format, so I will convert it to date format using the package lubridate:

```
library(lubridate)
stepsdate$date <- ymd(stepsdate$date)
```

The interval data is in format *hhmm* with no leading zeros, so I do the following:

- Pad it out to 4 integers with a semi-colon in the middle, which implies converting the field to character;
- Convert the string using the function hm from lubridate which yields an object of class period.

```
tmp <- sprintf("%02d:%02d", stepsdate$interval%/%100, stepsdate$interval%%100)
stepsdate$timeperiod <- hm(tmp)
head(stepsdate)
```

```
##   steps       date interval timeperiod
## 1    NA 2012-10-01        0         0S
## 2    NA 2012-10-01        5      5M 0S
## 3    NA 2012-10-01       10     10M 0S
## 4    NA 2012-10-01       15     15M 0S
## 5    NA 2012-10-01       20     20M 0S
## 6    NA 2012-10-01       25     25M 0S
```

4 What is the mean total number of steps taken per day?

For convenience I'll display the histogram after the caluculations. I start by calculating the sum of the steps taken each day, together with mean and median, using the plyr package for convenience:

```
library(plyr)
ddply(stepsdate, "date", summarize,
                         meanstepcount = sum(steps, na.rm = TRUE),
                         maxstep = max(steps[steps>0], na.rm = TRUE),
                         stepmean = round(mean(steps, na.rm = TRUE), 1),
                         stepmedian = median(steps[steps>0], na.rm = TRUE))
```

```
##          date meanstepcount maxstep stepmean stepmedian
## 1  2012-10-01             0    -Inf      NaN         NA
## 2  2012-10-02           126     117      0.4       63.0
## 3  2012-10-03         11352     613     39.4       61.0
## 4  2012-10-04         12116     547     42.1       56.5
## 5  2012-10-05         13294     555     46.2       66.0
```

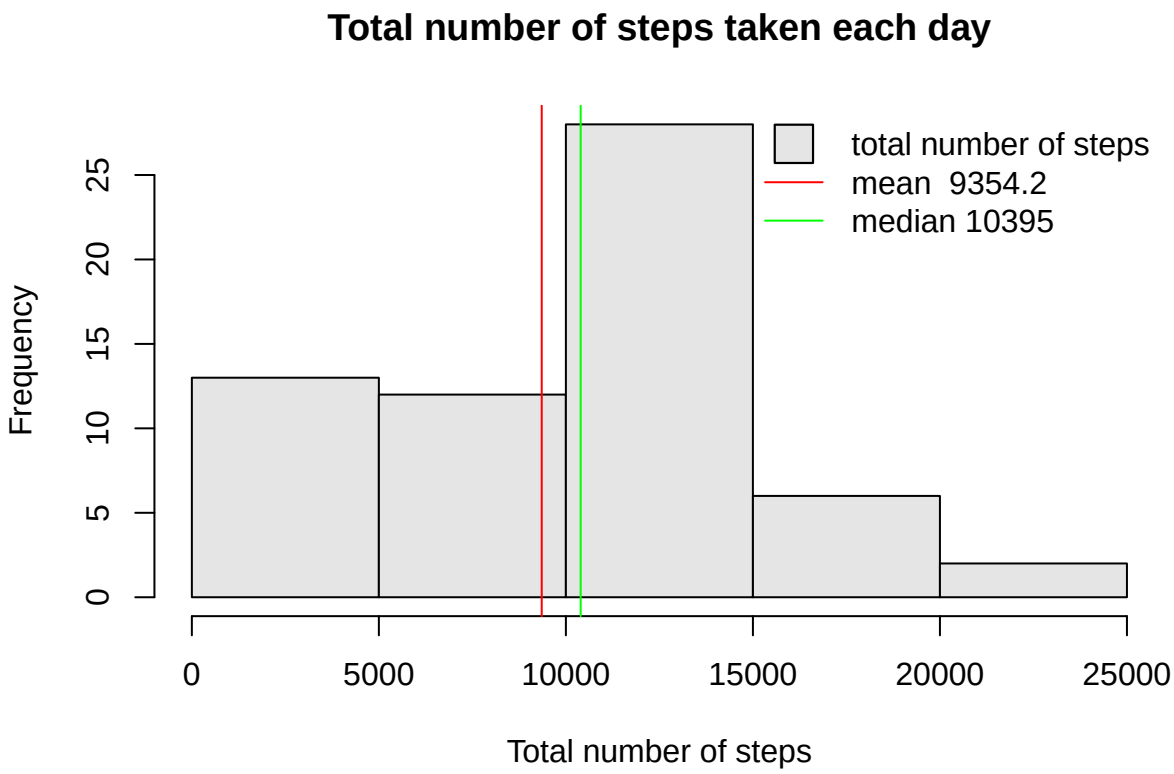
```
## 6  2012-10-06        15420     526     53.5       67.0
## 7  2012-10-07        11015     523     38.2       52.5
## 8  2012-10-08            0    -Inf      NaN         NA
## 9  2012-10-09        12811     748     44.5       48.0
## 10 2012-10-10         9900     413     34.4       56.5
## 11 2012-10-11        10304     748     35.8       35.0
## 12 2012-10-12        17382     802     60.4       46.0
## 13 2012-10-13        12426     542     43.1       45.5
## 14 2012-10-14        15098     540     52.4       60.5
## 15 2012-10-15        10139     786     35.2       54.0
## 16 2012-10-16        15084     758     52.4       64.0
## 17 2012-10-17        13452     744     46.7       61.5
## 18 2012-10-18        10056     759     34.9       52.5
## 19 2012-10-19        11829     512     41.1       74.0
## 20 2012-10-20        10395     532     36.1       49.0
## 21 2012-10-21         8821     501     30.6       48.0
## 22 2012-10-22        13460     783     46.7       52.0
## 23 2012-10-23         8918     499     31.0       56.0
## 24 2012-10-24         8355     533     29.0       51.5
## 25 2012-10-25         2492     443      8.7       35.0
## 26 2012-10-26         6778     440     23.5       36.5
## 27 2012-10-27        10119     555     35.1       72.0
## 28 2012-10-28        11458     533     39.8       61.0
## 29 2012-10-29         5018     591     17.4       54.5
## 30 2012-10-30         9819     523     34.1       40.0
## 31 2012-10-31        15414     757     53.5       83.5
## 32 2012-11-01            0    -Inf      NaN         NA
## 33 2012-11-02        10600     753     36.8       55.5
## 34 2012-11-03        10571     533     36.7       59.0
## 35 2012-11-04            0    -Inf      NaN         NA
## 36 2012-11-05        10439     785     36.2       66.0
## 37 2012-11-06         8334     630     28.9       52.0
## 38 2012-11-07        12883     766     44.7       58.0
## 39 2012-11-08         3219     359     11.2       42.5
## 40 2012-11-09            0    -Inf      NaN         NA
## 41 2012-11-10            0    -Inf      NaN         NA
## 42 2012-11-11        12608     540     43.8       55.0
## 43 2012-11-12        10765     542     37.4       42.0
## 44 2012-11-13         7336     444     25.5       57.0
## 45 2012-11-14            0    -Inf      NaN         NA
## 46 2012-11-15           41      33      0.1       20.5
## 47 2012-11-16         5441     475     18.9       43.0
## 48 2012-11-17        14339     753     49.8       65.5
## 49 2012-11-18        15110     785     52.5       80.0
## 50 2012-11-19         8841     789     30.7       34.0
## 51 2012-11-20         4472     500     15.5       58.0
## 52 2012-11-21        12787     758     44.4       55.0
## 53 2012-11-22        20427     567     70.9       65.0
## 54 2012-11-23        21194     760     73.6      113.0
## 55 2012-11-24        14478     785     50.3       65.5
## 56 2012-11-25        11834     551     41.1       84.0
## 57 2012-11-26        11162     709     38.8       53.0
## 58 2012-11-27        13646     806     47.4       57.0
## 59 2012-11-28        10183     733     35.4       70.0
```

```
## 60 2012-11-29          7047     568     24.5       44.5
## 61 2012-11-30             0    -Inf      NaN         NA
```

In the above output, the option message=FALSE has been set to suppress the message that loading plyer masks object here (whathever that means ...) and the warning=FALSE to suppress the warnings for max values when no step data is available (in which case the output is -Inf).

The histogram of the total number of steps taken each day discarding the missing values is below:

```
stepsday <- ddply(stepsdate, "date", summarize,
                        meanstepcount = sum(steps, na.rm = TRUE))
meansteps = round(mean(stepsday$meanstepcount), 1)
mediansteps = median(stepsday$meanstepcount)
hist(stepsday$meanstepcount,
     main = "Total number of steps taken each day",
     xlab = "Total number of steps",
     ylab = "Frequency",
     col = "grey90")
abline(v=meansteps,col="red")
abline(v=mediansteps,col="green")
legend("topright",
       c("total number of steps",
         paste("mean ",meansteps),
         paste("median",mediansteps)),
       col = c("black", "red", "green"),
       lty = c(0, 1, 1),
       lwd = c(0, 1, 1),
       pch = c(22, NA, NA),
       pt.bg = c("grey90", NA, NA),
       pt.cex = c(3, NA, NA),
       pt.lw = c(1.1, NA, NA),
       bty="n")
```



The mean and median function return 9354.2 and 10395 repectively.

5 What is the average daily activity pattern?

Below is a time series plot (i.e. type = "l") of the 5-minute interval (x-axis) and the average number of steps taken, averaged across all days (y-axis). This turned out to be trickier than first thought:

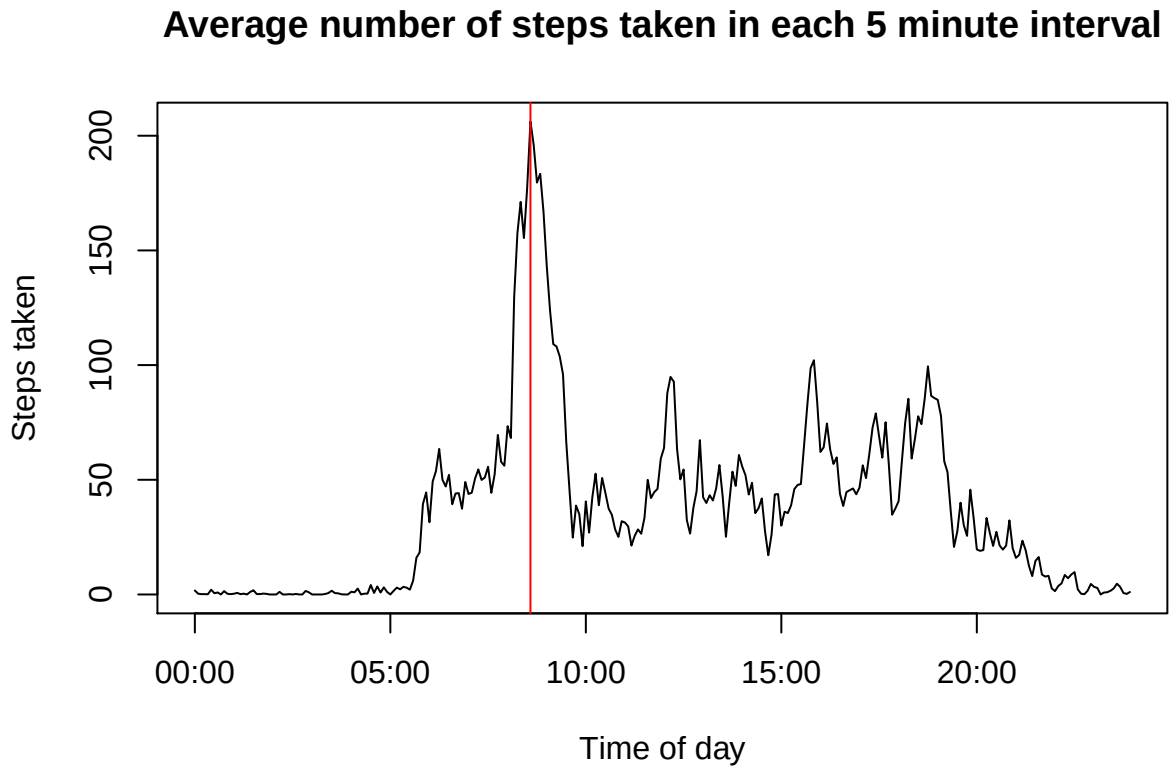
- The converted intervals *timeperiod* of class period can not be directly used with plot. Instead, a perhour column was added whereby the date remains the same and the intervals are added to create the time intervals over the same day, which I have arbitrarely set to the first avaiable date in the *date* variable.
- The default addition uses the timezone GMT but plot does not check for timezone and simply uses the machine's default, which means that I need to insure that the dates and intervals are added using the machine's timezone as hte time scale will be out of sync if the two timezones are different.

The advantage is that the x axis is now on a time scale.

```
stepsdate$dailyinterval <- ymd(stepsdate$date[1], tz = Sys.timezone()) + stepsdate$timeperiod
head(stepsdate$dailyinterval)
```

```
## [1] "2012-10-01 00:00:00 BST" "2012-10-01 00:05:00 BST"
## [3] "2012-10-01 00:10:00 BST" "2012-10-01 00:15:00 BST"
## [5] "2012-10-01 00:20:00 BST" "2012-10-01 00:25:00 BST"
```

```
stepsday <- ddply(stepsdate, "dailyinterval", summarize,
                        stepaverage = mean(steps, na.rm = TRUE))
#Finding the interval with most steps taken
maxsteps <- max(stepsday$stepaverage, na.rm = TRUE)
maxsteptime <- stepsday$dailyinterval[maxsteps == stepsday$stepaverage]
plot(stepsday$dailyinterval,
     stepsday$stepaverage,
     type="l",
     main = "Average number of steps taken in each 5 minute interval",
     xlab = "Time of day",
     ylab = "Steps taken")
abline(v=maxsteptime,col="red")
```




```
maxsteptime <- strftime(maxsteptime, format="%H:%M")
```

The 5-minute interval starting at 08:35 contains the maximum number of steps of 206.

Overwriting interval with a minute based integer would have been easier, for example:

```
stepsdate$interval <- stepsdate$interval%/%100 * 60 + stepsdate$interval%%100.
```

6 Imputing missing values

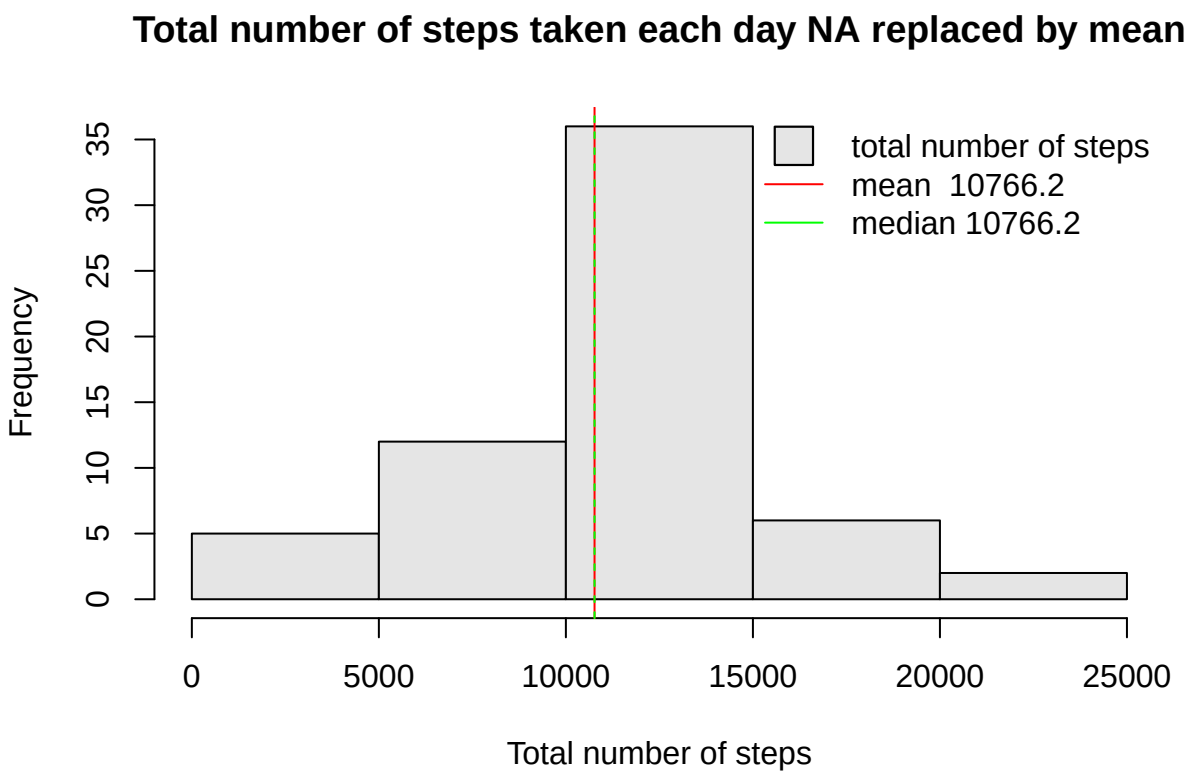
There are 2304 entries with missing values in the dataset out of a total of 17568 entries, or 13.1%.

The assignemnt asks for a new dataset to be created with the missing data filled in. So I copy the variable *steps* to the variable *stepscompleted* in the data frame *stepsdate* and replace the missing values in *stepscompleted* with the average for the relevant period as stored in the data frame *stepsday*.

```
stepsdate$stepscompleted <- stepsdate$steps
repairindex <- is.na(stepsdate$stepscompleted)
repairinterval <- match(stepsdate$dailyinterval[repairindex], stepsday$dailyinterval)
stepsdate$stepscompleted[repairindex] <- stepsday$stepaverage[repairinterval]
```

Now using the data with NA replaced by the mean in the correponding time interval, I draw the histogram of the total number of steps taken each day and calculate the mean and median:

```
stepsdaycompleted <- ddply(stepsdate, "date", summarize,
                           stepscompletedcount = sum(stepscompleted, na.rm = TRUE))
meanstepscompleted = round(mean(stepsdaycompleted$stepscompletedcount), 1)
medianstepscompleted = round(median(stepsdaycompleted$stepscompletedcount), 1)
hist(stepsdaycompleted$stepscompletedcount,
     main = "Total number of steps taken each day NA replaced by mean",
     xlab = "Total number of steps",
     ylab = "Frequency",
     col = "grey90")
abline(v=meanstepscompleted,col="red")
abline(v=medianstepscompleted,col="green", lty=2)
legend("topright",
       c("total number of steps",
         paste("mean ",meanstepscompleted),
         paste("median",medianstepscompleted)),
       col = c("black", "red", "green"),
       lty = c(0, 1, 1),
       lwd = c(0, 1, 1),
       pch = c(22, NA, NA),
       pt.bg = c("grey90", NA, NA),
       pt.cex = c(3, NA, NA),
       pt.lw = c(1.1, NA, NA),
       bty="n")
```



The mean function changed from 9354.2 to 10766.2 and the median function from 10395 to 10766.2. Both mean and median are now the same. The shape of the distribution changed with the bucket containing fewest steps decreasing the most.

Note: why I needed to add as.character to meanstepscompleted and medianstepscompleted in order to display those two numbers correctly in the above paragraph is a mistery as without it the numbers are diplayed unrounded in scientific notation even though they are correctly formatted in the graph amd as mediansteps is displayed properly.

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

I will use the weekdays() function on the dataset with the filled-in missing values and create a new factor variable *daytype* in the dataset with two levels - "weekday" and "weekend" - indicating whether a given date is a weekday or weekend day.

```
stepsdate$daytype <- weekdays(stepsdate$date)
stepsdate$daytype[which(!(stepsdate$daytype %in% c("Saturday", "Sunday")))] <- "weekday"
stepsdate$daytype[which(stepsdate$daytype %in% c("Saturday", "Sunday"))] <- "weekend"
```

And now the 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). I wanted to use ggplot2, so had to load scales as well to allow for a formatting of the x axis which allowed to remove the date (which shows up in ggplot2 but not in plot).

```r
library(ggplot2)
library(scales) # Daily interval as time without date

stepsdaytype <- ddply(stepsdate, c("dailyinterval", "daytype"), summarize,
                          meanstepcount = sum(steps, na.rm = TRUE))

ggplot(stepsdaytype, aes(x = dailyinterval, y = meanstepcount, group = daytype)) +
        scale_x_datetime(labels = date_format("%H:%M")) +
        geom_line(colour = "blue") +
        facet_wrap( ~ daytype, ncol = 1) +
        ggtitle("Weekday and weekend number of steps taken per 5 minute interval")
```