Reproducible Research: Peer Assessment 1

Programming Assignment 1

PART 1

by Prateek Sarangi, Sun , Mar 15 2020

Abstract: *This document represents the work for the* ***Programming Assignment 1*** *as part of* ***Reproducible esearch*** *online class offered by* ***Coursera****.*

1.1 Loading and preprocessing the data

1.1.1 Read the raw data in R

```
activity_raw <- read.csv("~/Downloads/Data/activity.csv")
```

1.1.2 Eliminate the rows which contains NAs

This process is done in two steps: 1. Detect the rows that contain NAs
2. Eliminate the rows with NAs(detected by step 1) from the raw data

```
activity_with_na <- apply(activity_raw, 1, function(x){any(is.na(x))})
sum(activity_with_na)
```

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

```
activity <- activity_raw[!activity_with_na, ]
```

Verify that number of rows of clean data is
is equal to the difference between the number of rows of raw data and number of rows with NA.

```
    nrow(activity) == nrow(activity_raw) - sum(activity_with_na)
```

```
## [1] TRUE
```

1.1.4 Transform column "date" in a R "Date" type format

```
typeof(activity$date)
```

```
## [1] "integer"
```

```
head(activity)
```

```
##     steps       date interval
## 289     0 2012-10-02        0
## 290     0 2012-10-02        5
## 291     0 2012-10-02       10
## 292     0 2012-10-02       15
## 293     0 2012-10-02       20
## 294     0 2012-10-02       25
```

```r
activity$date <- as.character(activity$date)
activity$date <- as.Date(activity$date, format = '%Y-%m-%d')
head(activity)
```

```
##     steps       date interval
## 289     0 2012-10-02        0
## 290     0 2012-10-02        5
## 291     0 2012-10-02       10
## 292     0 2012-10-02       15
## 293     0 2012-10-02       20
## 294     0 2012-10-02       25
```

```r
tail(activity)
```

```
##       steps       date interval
## 17275     0 2012-11-29     2330
## 17276     0 2012-11-29     2335
## 17277     0 2012-11-29     2340
## 17278     0 2012-11-29     2345
## 17279     0 2012-11-29     2350
## 17280     0 2012-11-29     2355
```

```r
typeof(activity$date)
```

```
## [1] "double"
```

```r
data.class(activity$date)
```

```
## [1] "Date"
```

```r
summary(activity$date)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "2012-10-02" "2012-10-16" "2012-10-29" "2012-10-30" "2012-11-16" "2012-11-29"
```

```r
start_rec <- min(activity$date)
end_rec <- max(activity$date)
(days_span <- end_rec - start_rec)
```

```
## Time difference of 58 days
```

```r
(number_days_rec <- length(levels(factor(activity$date))))
```

```
## [1] 53
```

From a quick look into the date it can be observed that the span of recordings is over 58 days
but it does not guarantee that we have records for every single day between 2012-10-02 and 2012-11-29.
To find the actual number of days for which activity has been recorded, extract the number of levels
for table activity$date which is 53 days. Therefore, we have only 53 days with recorded actvity.

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

1.2.1 Make a histogram of total number of stes taken each day

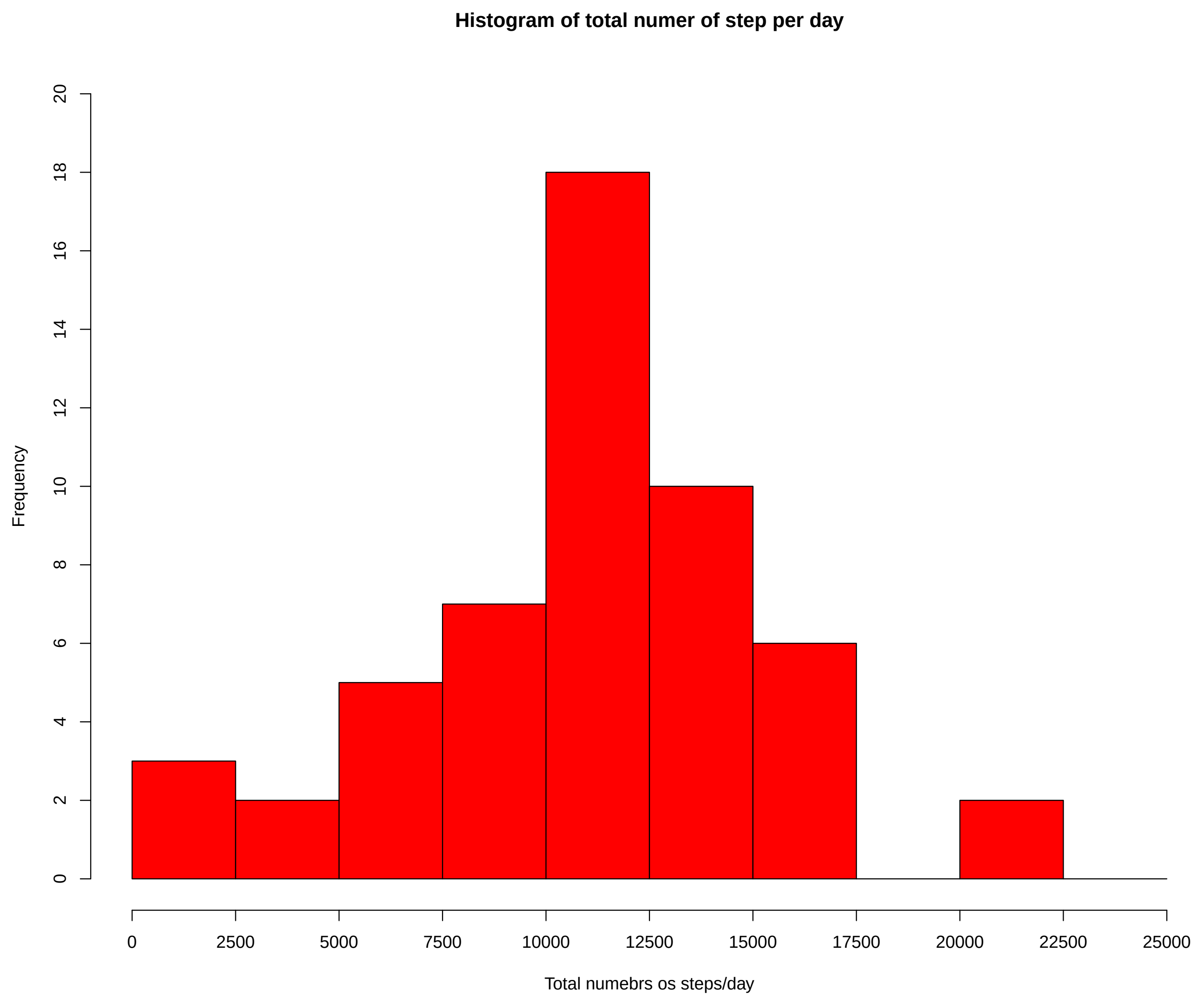
- A new data frame has been build. It has two columns [dates, steps] and 53 rows corresponding
  to the number of days that were recorded.

```
daily_steps <- data.frame(steps=numeric(0),
                date=as.Date(character()),
                stringsAsFactors=FALSE)
for (i in levels(factor(activity$date))) {
 new_vect <- c(sum(activity$steps[activity$date == i]), i)
# print(new_vect)
daily_steps[i, 1] <- sum(activity$steps[activity$date == i])
daily_steps[i, 2] <- i
}
rownames(daily_steps) <- c(1:nrow(daily_steps))
head(daily_steps)
```

```
##   steps       date
## 1   126 2012-10-02
## 2 11352 2012-10-03
## 3 12116 2012-10-04
## 4 13294 2012-10-05
## 5 15420 2012-10-06
## 6 11015 2012-10-07
```

- Bellow is the histogram showing the frequency of number of steps per day.
  It is a quick overview of the distribution centered around 11000steps/day.

```
hist(daily_steps$steps,
     col = "red",
     xlab = "Total numebrs os steps/day",
     main = "Histogram of total numer of step per day",
     breaks = seq(0, 25000, by = 2500),
     xlim = c(0, 25000),
     xaxp = c(0, 25000, 10),
     ylim = c(0, 20),
     yaxp = c(0, 20, 10)
)
```



```
# dev.copy(png,'/Users/ashwini/RepData_PeerAssessment1/figures/histogram_of_steps_from_activity_without
# dev.off()
```

1.2.2 Calculate and report the mean and median total number of steps taken per day

```
average_number_of_steps <- mean(daily_steps$steps)
median_of_daily_steps <- median(daily_steps$steps)
cat("Average number of steps per day is : ", average_number_of_steps)
```

```
## Average number of steps per day is :  10766.19
```

```
cat("The median of number of steps per day is : ", median_of_daily_steps)
```

```
## The median of number of steps per day is :  10765
```

1.3 What is the average daily activity pattern?

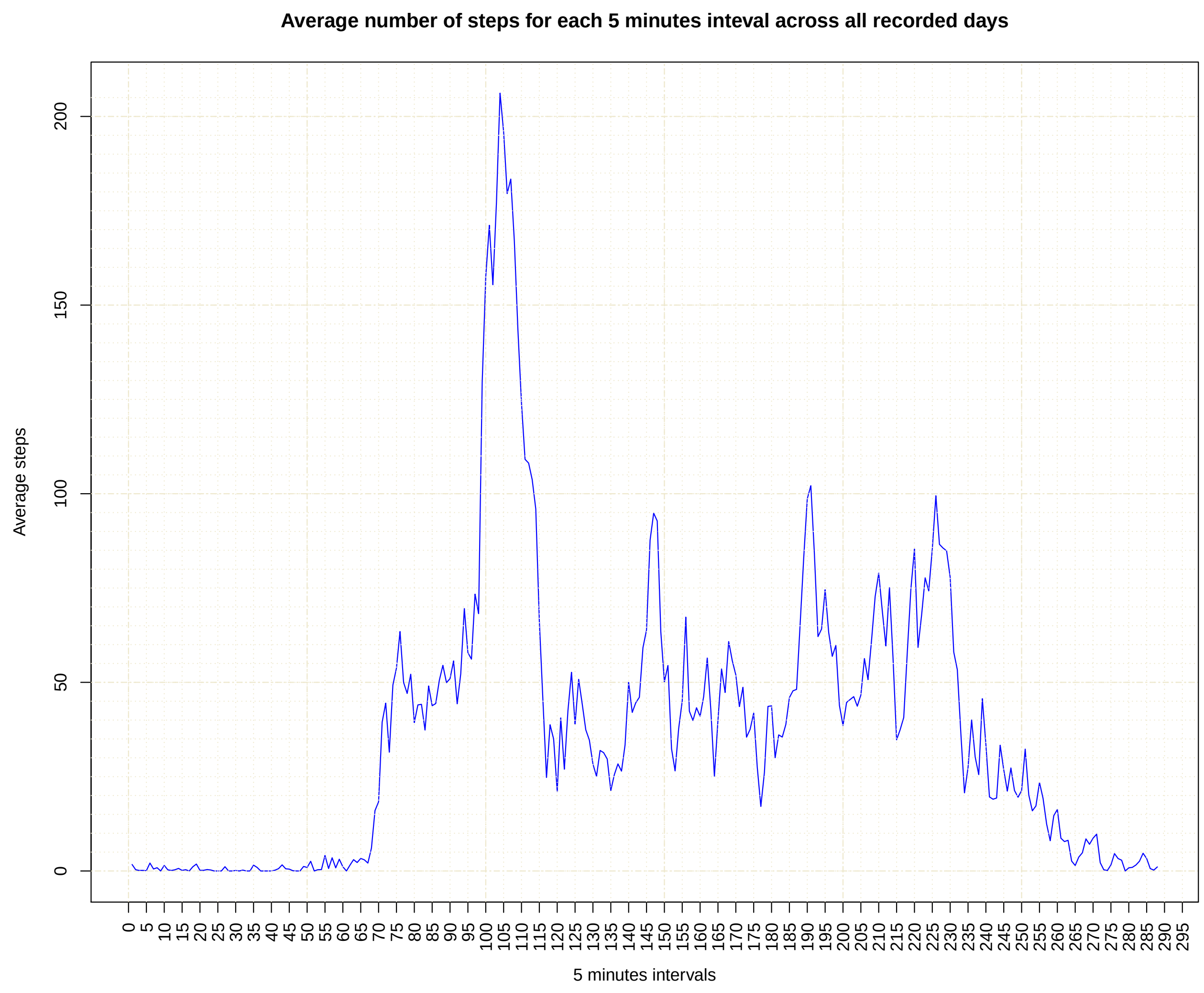
1.3.1 Build a new table and calculate the average number of steps for each 5 minute interval across all recorded days

```
average_per_interval <- data.frame(interval = numeric(length(levels(factor(activity$interval)))),
                                   average_steps_per_interval = numeric(length(levels(factor(activity$in
average_per_interval$interval = as.numeric(levels(factor(activity$interval)))
for(i in average_per_interval$interval){
average_per_interval[average_per_interval$interval == i, 2] <- mean(activity[activity$interval == i, ]$s
}
head(average_per_interval)
```

```
##   interval average_steps_per_interval
## 1        0                  1.7169811
## 2        5                  0.3396226
## 3       10                  0.1320755
## 4       15                  0.1509434
## 5       20                  0.0754717
## 6       25                  2.0943396
```

1.3.2 Plot the average of steps for each 5 minutes interval across all recorded days

```
plot(average_per_interval$average_steps_per_interval,
     xaxt = "n",
     type = "l",
     col = "blue",
     ylab = "Average steps",
     xlab = "5 minutes intervals",
     main = "Average number of steps for each 5 minutes inteval across all recorded days",
     panel.first = grid(nx = NULL, ny = NULL, lty = 6, col = "cornsilk2")
     )
axis(1,
     at=seq(min(average_per_interval$interval),
            max(average_per_interval$interval),
            5),
     las=2)
abline(v = seq(min(average_per_interval$interval),
            max(average_per_interval$interval),
            5),
       h = seq(min(average_per_interval$average_steps_per_interval),
            max(average_per_interval$average_steps_per_interval),
            5),
       col = "cornsilk2",
       lty = 3)
```



1.3.3 Which 5-minute interval, on average across all the days in the dataset, contains the maximum number of steps?

```
max_step_index <- which(average_per_interval$average_steps_per_interval == max(c(average_per_interval$av
interval_num <- average_per_interval$interval[max_step_index]
cat("The 5-minutes interval containing the maximum number of steps, ", max(c(average_per_interval$avera
```

```
## The 5-minutes interval containing the maximum number of steps, 206.1698, is 835.
```

NOTE: This corresponds to 8:35AM.

PART 2

2.1 Imputing missing values

2.1.1 Calculate and report the total number of missing values in the dataset (i.e. the total number of rows with NAs)

```
activity_with_na <- apply(activity_raw, 1, function(x){any(is.na(x))})
cat("Total number of rows with NAs is :", sum(activity_with_na))
```

```
## Total number of rows with NAs is : 2304
```

NOTE: This has already been calculated in the first part. ### 2.1.2 Devise a strategy for filling in all of the missing values in the dataset. The strategy does not need to be sophisticated. For example, you could use the mean/median for that day, or the mean for that 5-minute interval, etc. >>> I will replace the NA from original table with the mean of the number of steps per 5 minutes interval calculated in PART 1. >>> To do this, a table containg only NAs values from original data has been build. See table bellow. >>> The updates will be performed on this new table called activity_na

```
activity_na <- activity_raw[which(activity_with_na == TRUE), ]
activity_na$steps <- round(average_per_interval$average_steps_per_interval)
head(activity_na)
```

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

```
tail(activity_na)
```

```
##       steps       date interval
## 17563     3 2012-11-30     2330
## 17564     5 2012-11-30     2335
## 17565     3 2012-11-30     2340
## 17566     1 2012-11-30     2345
## 17567     0 2012-11-30     2350
## 17568     1 2012-11-30     2355
```

2.2 Update the original set

2.2.1 Create a new dataset that is equal to the original dataset but with the missing data filled in.

> The new table, activity_new, is built by concatenation between table without NA (activity) and table with updated NAs (activity_na). To restore the order of rows, I use builtin function sort() on the newly created table.

```
activity_new <- rbind(activity, activity_na)
activity_new <- activity_new[sort(as.integer(rownames(activity_new))), ]
head(activity_new)
```

```
##     steps       date interval
## 289     0 2012-10-02        0
## 290     0 2012-10-02        5
## 291     0 2012-10-02       10
## 292     0 2012-10-02       15
## 293     0 2012-10-02       20
## 294     0 2012-10-02       25
```

```
tail(activity_new)
```

```
##       steps       date interval
## 17563     3 2012-11-30     2330
## 17564     5 2012-11-30     2335
## 17565     3 2012-11-30     2340
## 17566     1 2012-11-30     2345
## 17567     0 2012-11-30     2350
## 17568     1 2012-11-30     2355
```

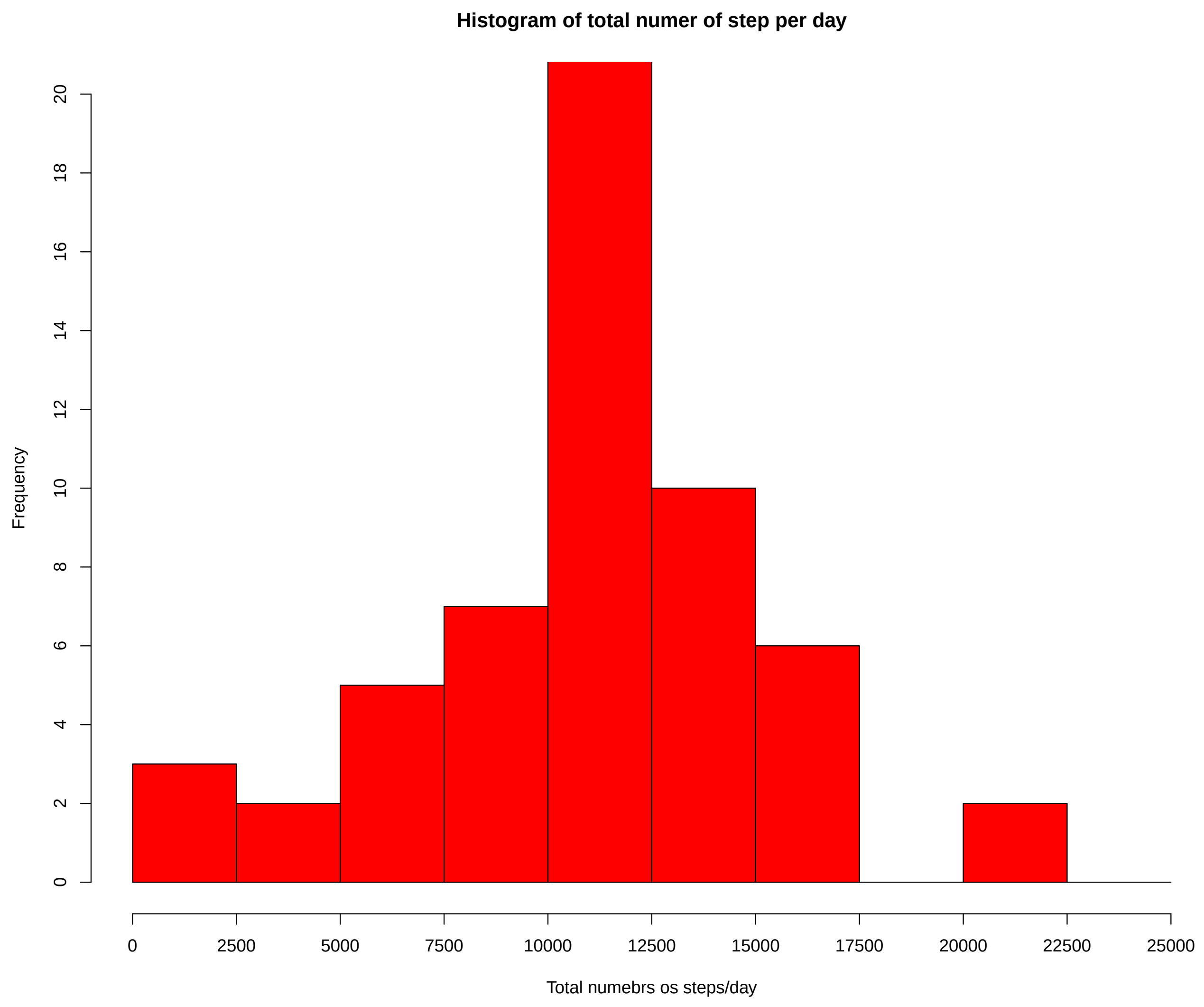
2.2.2 Make a histogram of the total number of steps taken each day and Calculate and report the mean and median total number of steps taken per day. Do these values differ from the estimates from the first part of the assignment? What is the impact of imputing missing data on the estimates of the total daily number of steps?

- A new data frame has been build. It has two columns [dates, steps] and 53
  rows corresponding
  to the total number of days from the updated activity table.

```r
daily_steps_new<- data.frame(steps=numeric(0),
                date=as.Date(character()),
                stringsAsFactors=FALSE)
for (i in levels(factor(activity_new$date))) {
 new_vect <- c(sum(activity_new$steps[activity_new$date == i]), i)
# print(new_vect)
daily_steps_new[i, 1] <- sum(activity_new$steps[activity_new$date == i])
daily_steps_new[i, 2] <- i
}
rownames(daily_steps_new) <- c(1:nrow(daily_steps_new))
head(daily_steps_new)
```

```
##   steps       date
## 1 10762 2012-10-01
## 2   126 2012-10-02
## 3 11352 2012-10-03
## 4 12116 2012-10-04
## 5 13294 2012-10-05
## 6 15420 2012-10-06
```

```r
hist(daily_steps_new$steps,
     col = "red",
     xlab = "Total numebrs os steps/day",
     main = "Histogram of total numer of step per day",
     breaks = seq(0, 25000, by = 2500),
     xlim = c(0, 25000),
     xaxp = c(0, 25000, 10),
     ylim = c(0, 20),
     yaxp = c(0, 20, 10)
)
```



```
new_average_number_of_steps <- mean(daily_steps_new$steps)
new_median_of_daily_steps <- median(daily_steps_new$steps)
cat("Average number of steps per day is : ", new_average_number_of_steps)
```

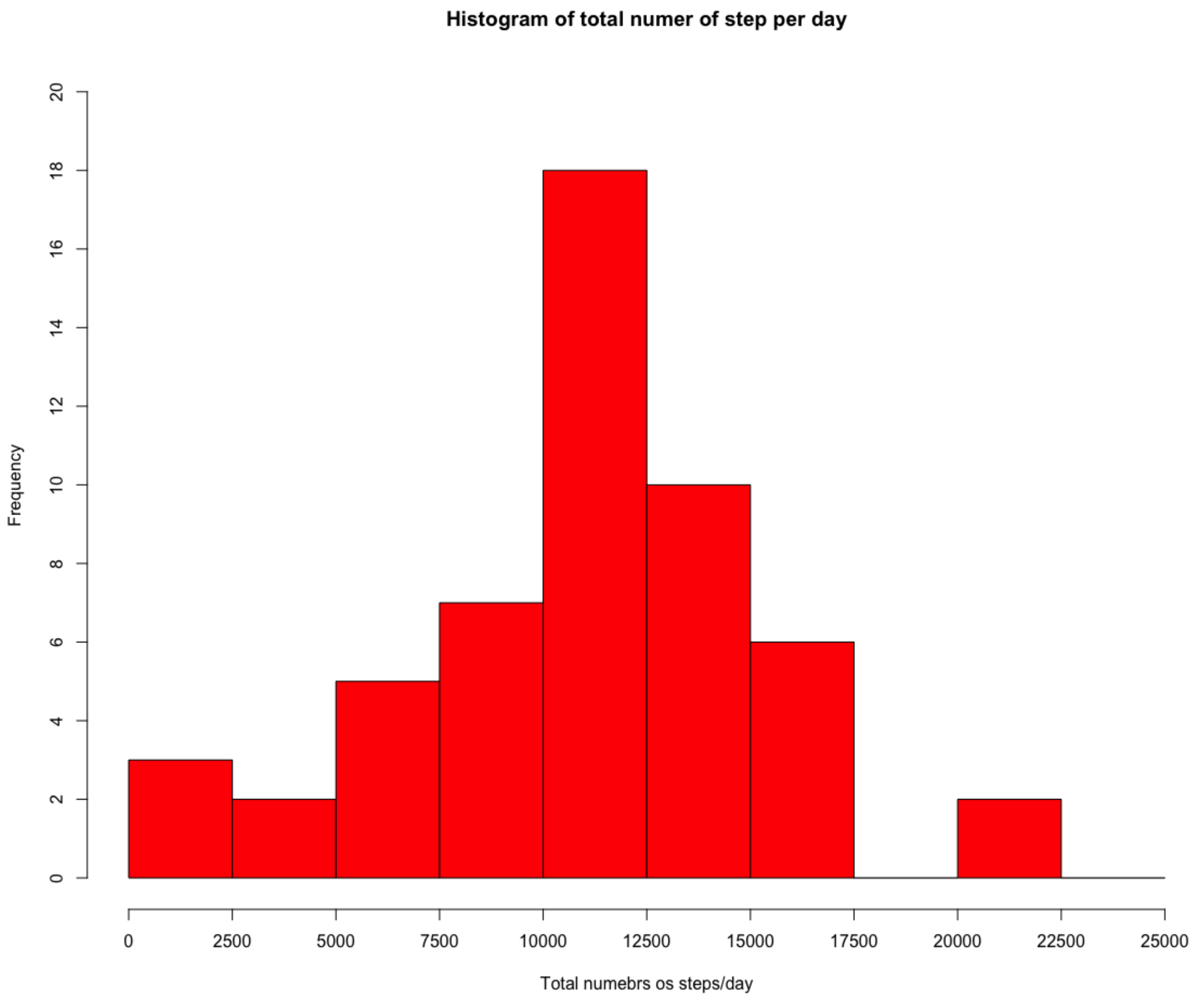
```
## Average number of steps per day is :  10765.64
```

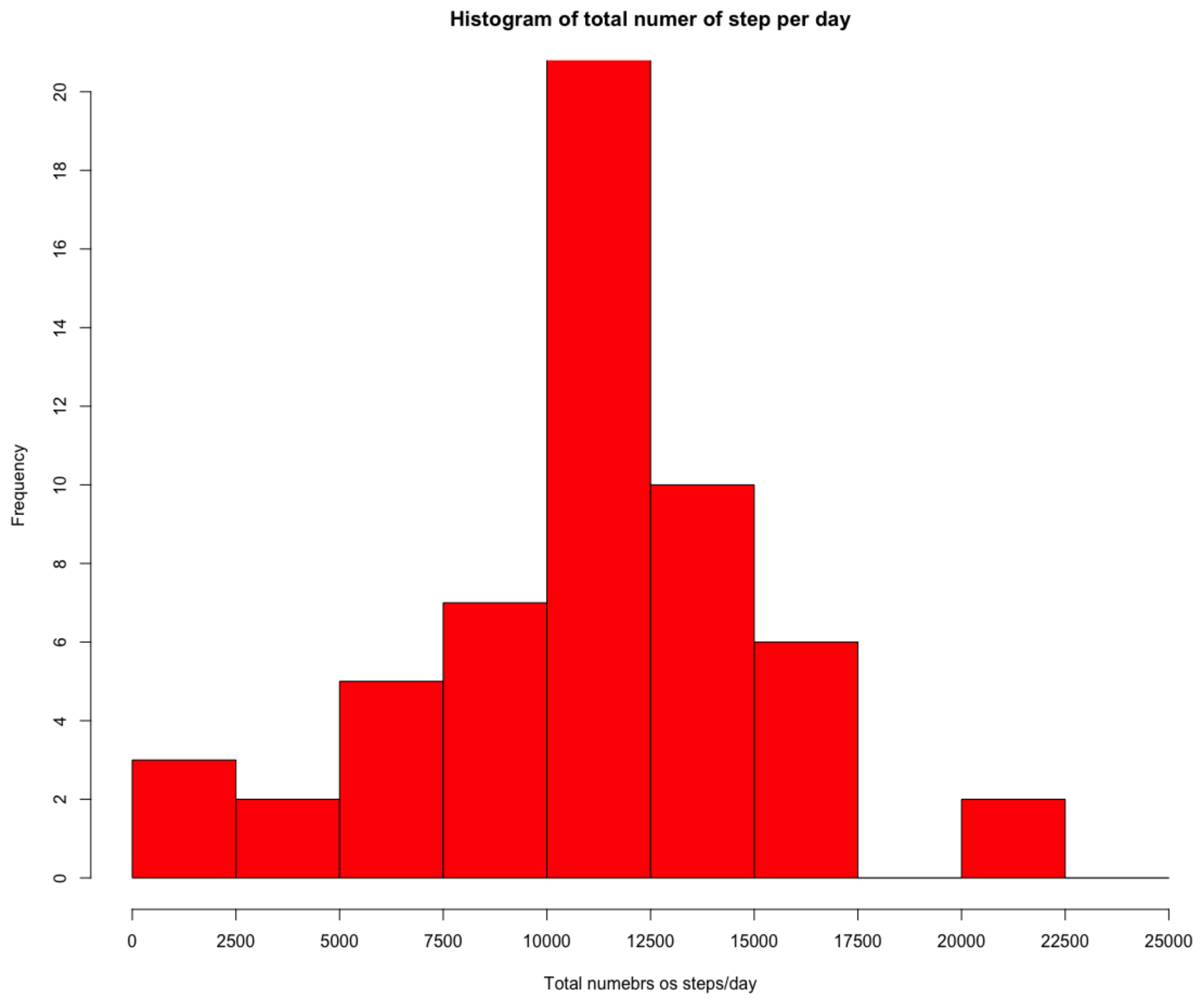
```
cat("The median of number of steps per day is : ", new_median_of_daily_steps)
```

```
## The median of number of steps per day is :  10762
```

2.2.3 Summary:

Compare the two cases from PART 1 and PART 2 – cleaned data(NA were removed) and updated data (NA were replaced with the mean of 5-minutes inteval)
1. The mean remains the same (it makes sense as NA were replaced by the means of 5-minutes interval) 2. The median is "almost" the same. The median for the "Activity Updated" slightly moves away from the mean 3. The impact of updating the original data with the mean of 5-minutes interval is that we will see an increase in the frequencies 4. This can be seen in the two histograms reproduced here for convenience | Activity no NA | Activity Updated — | — | — median |10766 |10765 mean |10765 |10762 **Reproduce the two histograms side-by-side** | Activity without NA | Activity Updated — | — | — |


Histogram of total numer of step per day
Frequency
0
2
4
6
8
10
12
14
16
18
20
0
2500
5000
7500
10000
12500
15000
17500
20000
22500
25000
Total numebrs os steps/day



|
PART 3 —— ## 3.1 Are there differences in activity patterns between weekdays and weekends? ### 3.1.1 Create a new factor variable in the dataset with two levels – "weekday" and "weekend" indicating whether a given date is a weekday or weekend day.

```
activity_days <- cbind(activity_new, days = (activity_new$date))
head(activity_days)
```

```
##     steps       date interval       days
## 289     0 2012-10-02        0 2012-10-02
## 290     0 2012-10-02        5 2012-10-02
## 291     0 2012-10-02       10 2012-10-02
## 292     0 2012-10-02       15 2012-10-02
## 293     0 2012-10-02       20 2012-10-02
## 294     0 2012-10-02       25 2012-10-02
```

```
activity_days$days <- weekdays(activity_days$days)
head(activity_days)
```

```
##     steps       date interval    days
## 289     0 2012-10-02        0 Tuesday
## 290     0 2012-10-02        5 Tuesday
## 291     0 2012-10-02       10 Tuesday
## 292     0 2012-10-02       15 Tuesday
## 293     0 2012-10-02       20 Tuesday
## 294     0 2012-10-02       25 Tuesday
```

```
activity_days$days[which(activity_days$days == c("Saturday", "Sunday"))] <- "weekend"
activity_days$days[which(activity_days$days != "weekend")] <- "weekday"
#factor(activity_days$days)
activity_weekdays <- activity_days[activity_days$days == "weekday", ]
activity_weekends <- activity_days[activity_days$days == "weekend", ]
head(activity_weekdays)
```

```
##     steps       date interval    days
## 289     0 2012-10-02        0 weekday
## 290     0 2012-10-02        5 weekday
## 291     0 2012-10-02       10 weekday
## 292     0 2012-10-02       15 weekday
## 293     0 2012-10-02       20 weekday
## 294     0 2012-10-02       25 weekday
```

```
head(activity_weekends)
```

```
##      steps       date interval    days
## 1441     0 2012-10-06        0 weekend
## 1443     0 2012-10-06       10 weekend
## 1445     0 2012-10-06       20 weekend
## 1447     0 2012-10-06       30 weekend
## 1449     0 2012-10-06       40 weekend
## 1451     0 2012-10-06       50 weekend
```

```
average_weekdays_interval <- data.frame(interval = numeric(length(levels(factor(activity_weekdays$inter
                              average_steps_per_interval = numeric(length(levels(factor(activity_weekda
average_weekdays_interval$interval <- as.numeric(levels(factor(activity_weekdays$interval)))
for(i in average_weekdays_interval$interval){
  average_weekdays_interval[average_weekdays_interval$interval == i, 2] <- mean(activity_weekdays[activi
}
head(average_weekdays_interval)
```

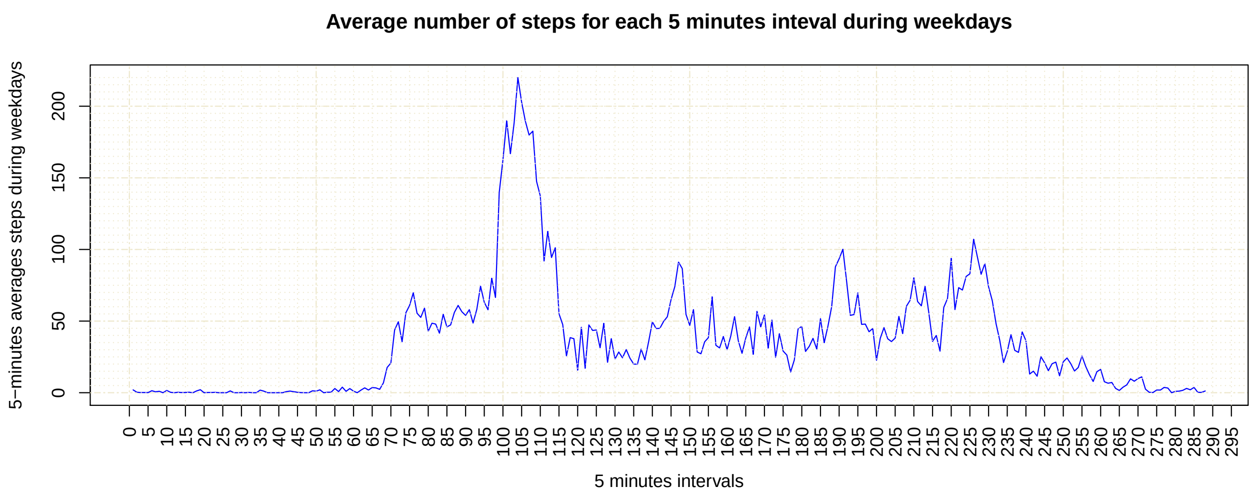
```
##   interval average_steps_per_interval
## 1        0                  1.9811321
## 2        5                  0.3396226
## 3       10                  0.1320755
## 4       15                  0.1509434
## 5       20                  0.0754717
## 6       25                  1.3773585
```

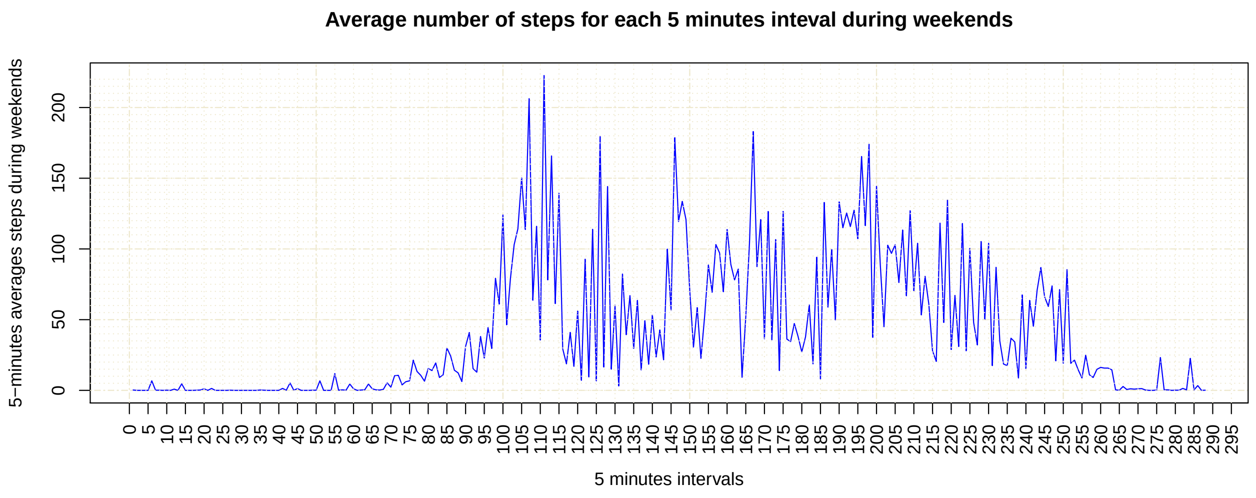
```
average_weekends_interval <- data.frame(interval = numeric(length(levels(factor(activity_weekends$inter
                              average_steps_per_interval = numeric(length(levels(factor(activity_weekend
average_weekends_interval$interval = as.numeric(levels(factor(activity_weekends$interval)))
for(i in average_weekends_interval$interval){
  average_weekends_interval[average_weekends_interval$interval == i, 2] <- mean(activity_weekends[activi
}
head(average_weekends_interval)
```

```
##   interval average_steps_per_interval
## 1        0                       0.25
## 2        5                       0.00
## 3       10                       0.00
## 4       15                       0.00
## 5       20                        0.00
## 6       25                        6.75
```

3.1.2 Make a panel plot containing 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 weekday days or weekend days (y-axis).

```
par(mfrow = c(2,1))
plot(average_weekdays_interval$average_steps_per_interval,
     xaxt = "n",
     type = "l",
     col = "blue",
     ylab = "5-minutes averages steps during weekdays",
     xlab = "5 minutes intervals",
     main = "Average number of steps for each 5 minutes inteval during weekdays",
     panel.first = grid(nx = NULL, ny = NULL, lty = 6, col = "cornsilk2")
     )
axis(1,
     at=seq(min(average_weekdays_interval$interval),
            max(average_weekdays_interval$interval),
            5),
     las=2)
abline(v = seq(min(average_weekdays_interval$interval),
            max(average_weekdays_interval$interval),
            5),
       h = seq(min(average_weekdays_interval$average_steps_per_interval),
            max(average_weekdays_interval$average_steps_per_interval),
            5),
       col = "cornsilk2",
       lty = 3)
plot(average_weekends_interval$average_steps_per_interval,
     xaxt = "n",
     type = "l",
     col = "blue",
     ylab = "5-minutes averages steps during weekends",
     xlab = "5 minutes intervals",
     main = "Average number of steps for each 5 minutes inteval during weekends",
     panel.first = grid(nx = NULL, ny = NULL, lty = 6, col = "cornsilk2")
     )
axis(1,
     at=seq(min(average_weekends_interval$interval),
            max(average_weekends_interval$interval),
            5),
     las=2)
abline(v = seq(min(average_weekends_interval$interval),
            max(average_weekends_interval$interval),
            5),
       h = seq(min(average_weekends_interval$average_steps_per_interval),
            max(average_weekends_interval$average_steps_per_interval),
            5),
       col = "cornsilk2",
       lty = 3)
```


Average number of steps for each 5 minutes inteval during weekdays
5−minutes averages steps during weekdays
0
50
100
150
200
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120 125 130 135 140 145 150 155 160 165 170 175 180 185 190 195 200 205 210 215 220 225 230 235 240 245 250 255 260 265 270 275 280 285 290 295
5 minutes intervals


Average number of steps for each 5 minutes inteval during weekends
5−minutes averages steps during weekends
0
50
100
150
200
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120 125 130 135 140 145 150 155 160 165 170 175 180 185 190 195 200 205 210 215 220 225 230 235 240 245 250 255 260 265 270 275 280 285 290 295
5 minutes intervals