Reproducible Research: Peer Assessment 1

Background of The Assessment Problem

** Reference ** The details of the problem, data is available from original Prof RD Peng GitHub repo

This document presents the results of the Reproducible Research Peer Assessment#1 in a report using **a single R markdown document** that can be processed by **knitr** and be transformed into an HTML file.

Through this report you can see that activities on weekdays mostly follow a work related routine, where we find some more intensity activity in little a free time that the employ can made some sport.

An important consideration is the fact of our data presents as a t-student distribution (see both histograms), it means that the impact of imputing missing values with the mean has a good impact on our predictions without a significant distortion in the distribution of the data.

Loading and preprocessing the data

Before loading and preprocessing the data we will load the necessary libraries, like **knitr** for single file markdown output in html or latex/pdf. Other libraries are **ggplot, lattice, data.table,**, etc. Also we set the **echo=TRUE** for all code chunks to be readable and reproducible. For the sake of convenience, we set this as global option, rather at each code chunk level.

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.1.2
```

```
opts_chunk$set(echo = TRUE)
```

Load required libraries

```
library(data.table)
library(ggplot2) # we shall use ggplot2 for plotting figures
```

Loading and preprocessing the data

This assignment makes use of data from a personal activity monitoring device. This device collects data at 5 minute intervals through out the day. The data consists of two months of data from an anonymous individual collected during the months of October and November, 2012 and include the number of steps taken in 5 minute intervals each day.

This assignment instructions request to show any code that is needed to loading and preprocessing the data, like to:

1. Load the data (i.e. > read.csv())

2. Process/transform the data (if necessary) into a format suitable for your analysis

Load the required data

The following statement is used to load the data using `read.csv()`.

Note: It is assumed that the file activity.csv is in the current working directory. File can be downloaded from here

```
rdata <- read.csv(unz("activity.zip", "activity.csv"))
```

tidy the data or preprocess the data

We convert the **date** field to `Date` class and **interval** field to `Factor` class.

```
rdata$date <- as.Date(rdata$date, format = "%Y-%m-%d")
rdata$interval <- as.factor(rdata$interval)
```

Now, let us check the data using `str()` method:

```
names(rdata)
```

```
## [1] "steps"    "date"     "interval"
```

```
str(rdata)
```

```
## 'data.frame':    17568 obs. of  3 variables:
##  $ steps   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ date    : Date, format: "2012-10-01" "2012-10-01" ...
##  $ interval: Factor w/ 288 levels "0","5","10","15",..: 1 2 3 4 5 6 7 8 9 10 ...
```

What is mean total number of steps taken per day?

Lets compute the number of missing vs. complete records

```
originalValue <- complete.cases(rdata)
nMissing <- length(originalValue[originalValue==FALSE])                       # number of records with N
nComplete <- length(originalValue[originalValue==TRUE])                       # number of complete recor

title="Missing vs. Complete Cases"
barplot(table(originalValue),main=title,xaxt='n', col="gray")                 # render Complete Cases ba
axis(side=1,at=c(.7,1.9),labels=c("Missing","Complete"),tick=FALSE)           # render axis
text(.7,0,labels=nMissing, pos=3)                                             # label the NA's bar
text(1.9,0,labels=nComplete, pos=3)
```

Now here we ignore the missing values(*a valid assumption*).

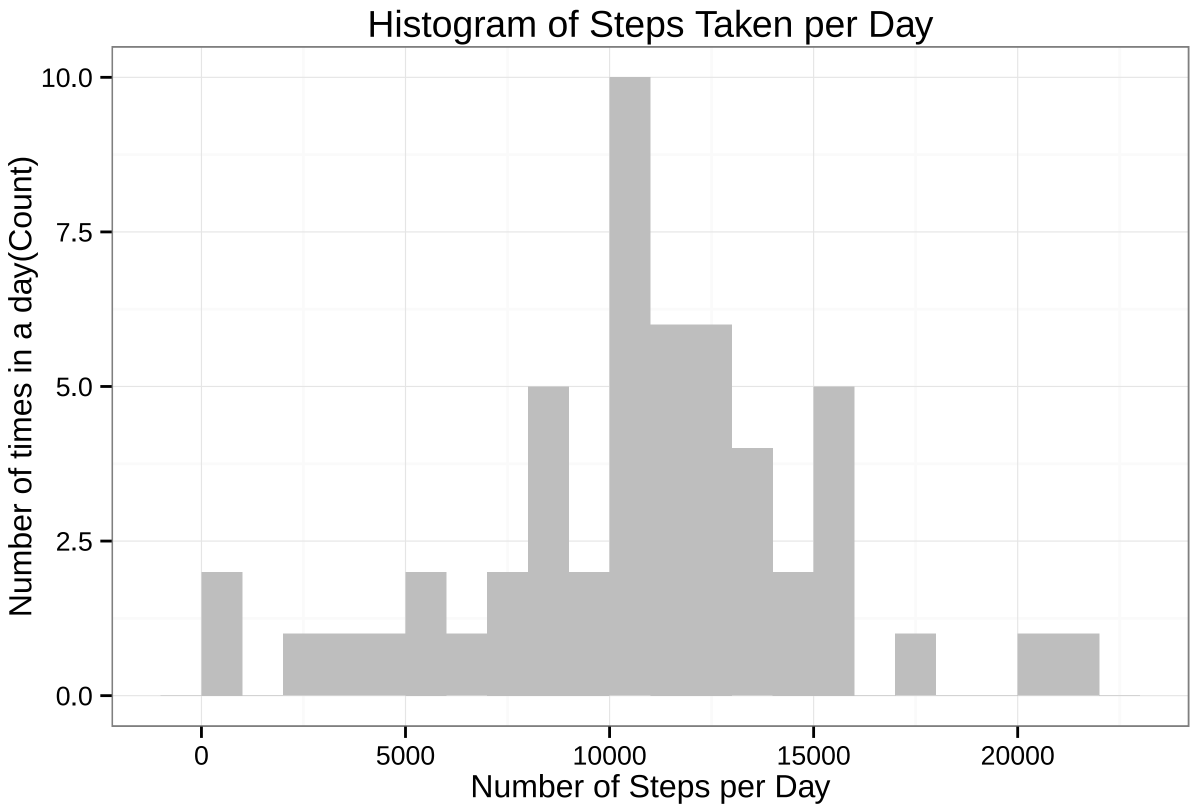
We proceed by calculating the total steps per day.

```
steps_per_day <- aggregate(steps ~ date, rdata, sum)
colnames(steps_per_day) <- c("date","steps")
head(steps_per_day)
```

```
##         date steps
## 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
```

1. Now we make a histogram of the total number of steps taken per day, plotted with appropriate bin interval.

```
ggplot(steps_per_day, aes(x = steps)) +
       geom_histogram(fill = "gray", binwidth = 1000) +
        labs(title="Histogram of Steps Taken per Day",
             x = "Number of Steps per Day", y = "Number of times in a day(Count)") + theme_bw()
```



2. Now we calculate the ***mean*** and ***median*** of the number of steps taken per day.

```
steps_mean   <- mean(steps_per_day$steps, na.rm=TRUE)
steps_median <- median(steps_per_day$steps, na.rm=TRUE)
```

The mean is **10766.189** and median is **10765**.

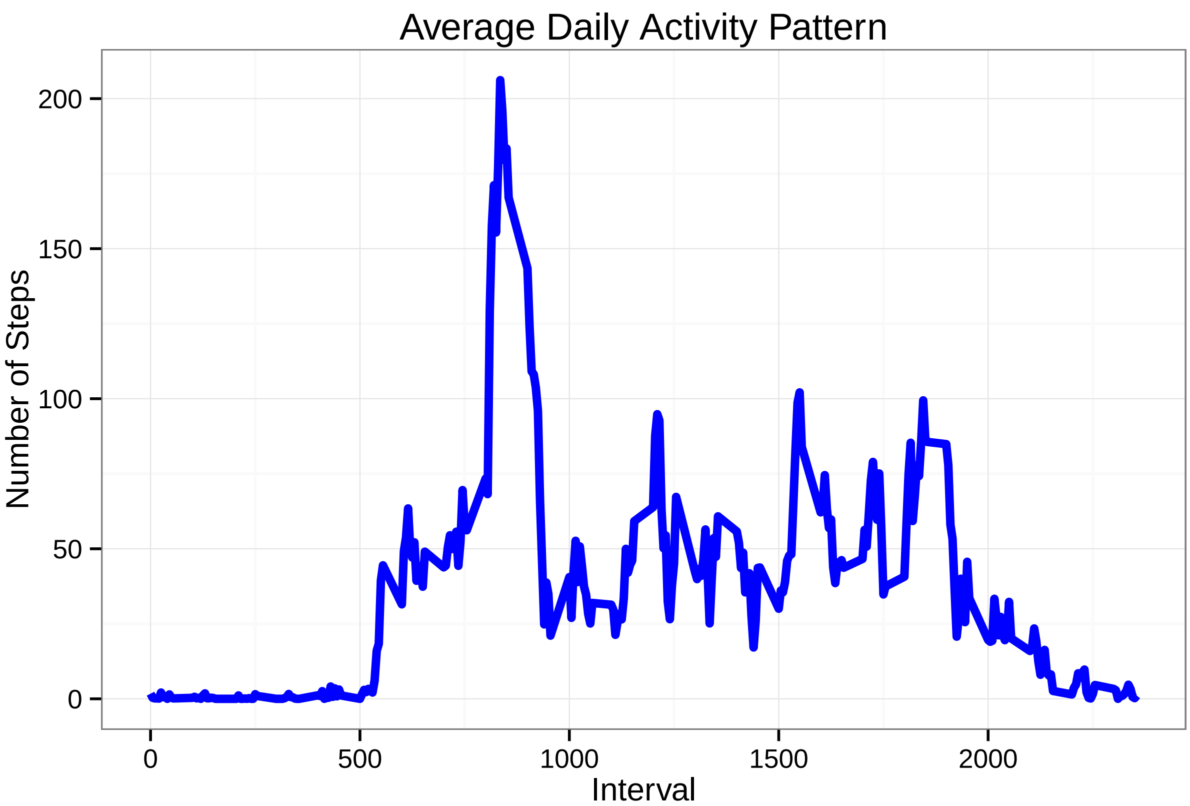
What is the average daily activity pattern?

We calculate the aggregation of steps by intervals of 5-minutes and convert the intervals as integers and save them in a data frame called `steps_per_interval`.

```
steps_per_interval <- aggregate(rdata$steps,
                                by = list(interval = rdata$interval),
                                FUN=mean, na.rm=TRUE)
#convert to integers
##this helps in plotting
steps_per_interval$interval <-
        as.integer(levels(steps_per_interval$interval)[steps_per_interval$interval])
colnames(steps_per_interval) <- c("interval", "steps")
```

1. We make the plot with the time series of the average number of steps taken (averaged across all days) versus the 5-minute intervals:

```r
ggplot(steps_per_interval, aes(x=interval, y=steps)) +
        geom_line(color="blue", size=1.5) +
        labs(title="Average Daily Activity Pattern", x="Interval", y="Number of Steps") +
        theme_bw()
```



2. Now, we find the 5-minute interval with the containing the maximum number of steps:

```r
max_interval <- steps_per_interval[which.max(
        steps_per_interval$steps),]
```

The **835th** interval has maximum **206** steps.

Imputing missing values:

1. Total number of missing values:

The total number of missing values in steps can be calculated using `is.na()` method to check whether the value is mising or not and then summing the logical vector.

```r
missing_vals <- sum(is.na(rdata$steps))
```

The total number of ***missing values*** are **2304**.

2. Strategy for filling in all of the missing values in the dataset

To populate missing values, we choose to replace them with the mean value at the same interval across days. In most of the cases the median is a better centrality measure than mean, but in our case the total median is not much far away from total mean, and probably we can make the mean and median meets.

We create a function `na_fill(data, pervalue)` which the `data` arguement is the `rdata` data frame and `pervalue` arguement is the `steps_per_interval` data frame.

```
na_fill <- function(data, pervalue) {
        na_index <- which(is.na(data$steps))
        na_replace <- unlist(lapply(na_index, FUN=function(idx){
                interval = data[idx,]$interval
                pervalue[pervalue$interval == interval,]$steps
        }))
        fill_steps <- data$steps
        fill_steps[na_index] <- na_replace
        fill_steps
}

rdata_fill <- data.frame(
        steps = na_fill(rdata, steps_per_interval),
        date = rdata$date,
        interval = rdata$interval)
str(rdata_fill)
```

```
## 'data.frame':    17568 obs. of  3 variables:
##  $ steps   : num  1.717 0.3396 0.1321 0.1509 0.0755 ...
##  $ date    : Date, format: "2012-10-01" "2012-10-01" ...
##  $ interval: Factor w/ 288 levels "0","5","10","15",..: 1 2 3 4 5 6 7 8 9 10 ...
```

We check that are there any missing values remaining or not

```
sum(is.na(rdata_fill$steps))
```

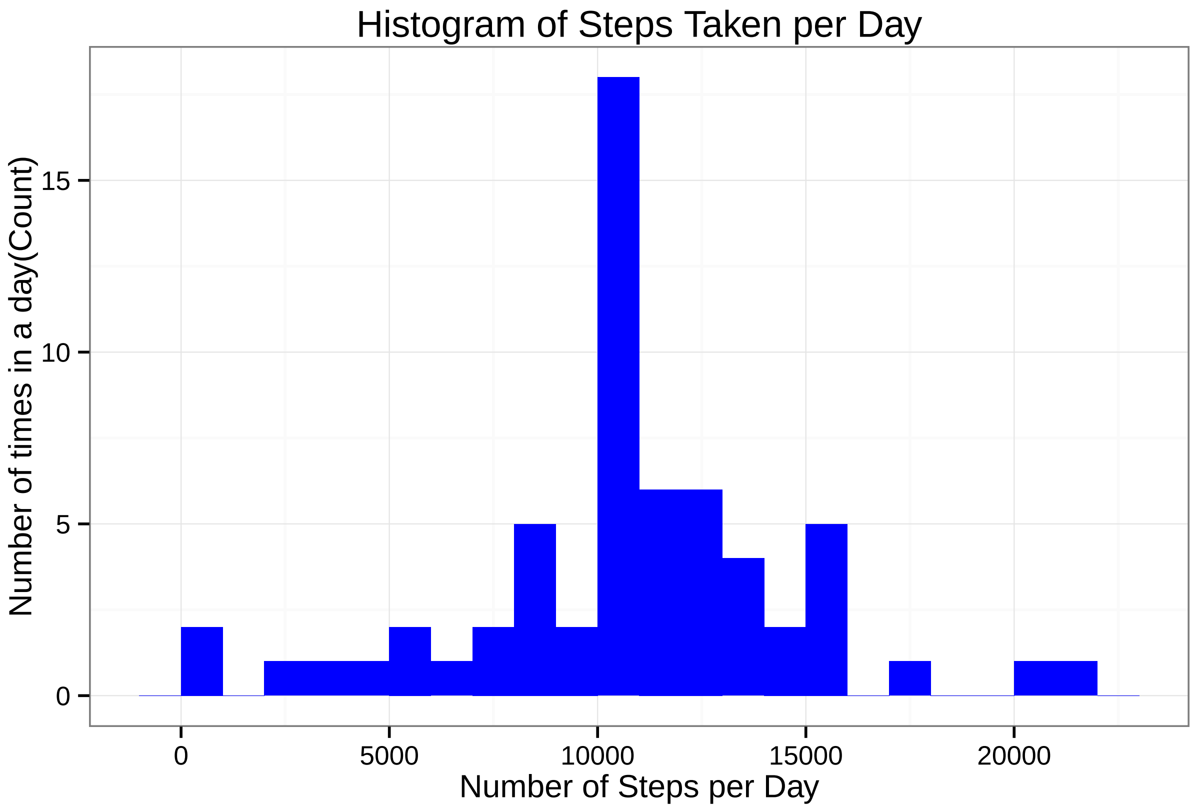
```
## [1] 0
```

Zero output shows that there are ***NO MISSING VALUES***.

3. A histogram of the total number of steps taken each day

Now let us plot a histogram of the daily total number of steps taken, plotted with a bin interval of 1000 steps, after filling missing values.

```
fill_steps_per_day <- aggregate(steps ~ date, rdata_fill, sum)
colnames(fill_steps_per_day) <- c("date","steps")

##plotting the histogram
ggplot(fill_steps_per_day, aes(x = steps)) +
       geom_histogram(fill = "blue", binwidth = 1000) +
        labs(title="Histogram of Steps Taken per Day",
             x = "Number of Steps per Day", y = "Number of times in a day(Count)") + theme_bw()
```



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

```
steps_mean_fill   <- mean(fill_steps_per_day$steps, na.rm=TRUE)
steps_median_fill <- median(fill_steps_per_day$steps, na.rm=TRUE)
```

The mean is **10766.189** and median is **10766.189**.

Do these values differ from the estimates from the first part of the assignment?

Yes, these values do differ slightly.

- **Before filling the data**
    1. Mean : **10766.189**
    2. Median: **10765**
- **After filling the data**
    1. Mean : **10766.189**
    2. Median: **10766.189**

We see that the values after filling the data mean and median are equal.

What is the impact of imputing missing data on the estimates of the total daily number of steps?

As you can see, comparing with the calculations done in the first section of this document, we observe that while the mean value remains unchanged, the median value has shifted and virtual matches to the mean.

Since our data has shown a t-student distribution (see both histograms), it seems that the impact of imputing missing values has increase our peak, but it's not affect negatively our predictions.

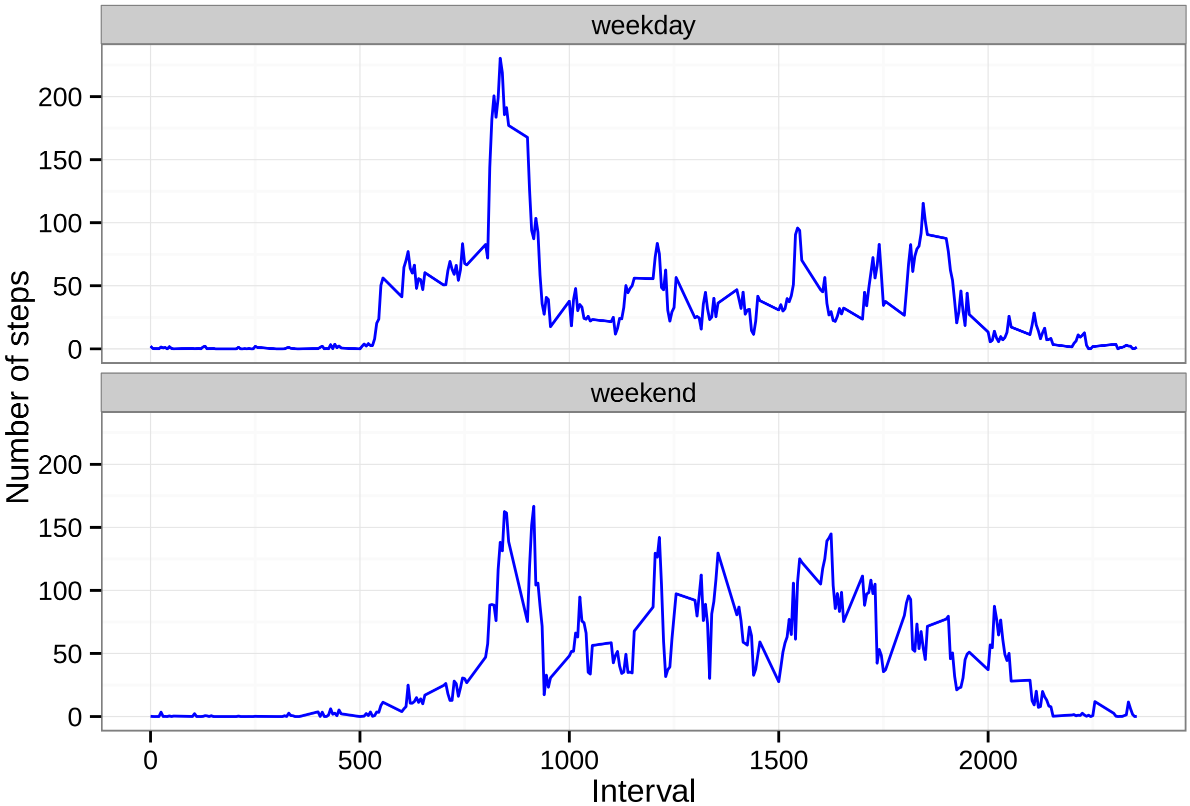
Are there differences in activity patterns between weekdays and weekends?

We do this comparison with the table with filled-in missing values.
1. Augment the table with a column that indicates the day of the week
2. Subset the table into two parts - weekends (Saturday and Sunday) and weekdays (Monday through Friday).
3. Tabulate the average steps per interval for each data set.
4. Plot the two data sets side by side for comparison.

```
weekdays_steps <- function(data) {
    weekdays_steps <- aggregate(data$steps, by=list(interval = data$interval),
                          FUN=mean, na.rm=T)
    # convert to integers for plotting
    weekdays_steps$interval <-
            as.integer(levels(weekdays_steps$interval)[weekdays_steps$interval])
    colnames(weekdays_steps) <- c("interval", "steps")
    weekdays_steps
}

data_by_weekdays <- function(data) {
    data$weekday <-
            as.factor(weekdays(data$date)) # weekdays
    weekend_data <- subset(data, weekday %in% c("Saturday","Sunday"))
    weekday_data <- subset(data, !weekday %in% c("Saturday","Sunday"))

    weekend_steps <- weekdays_steps(weekend_data)
    weekday_steps <- weekdays_steps(weekday_data)

    weekend_steps$dayofweek <- rep("weekend", nrow(weekend_steps))
    weekday_steps$dayofweek <- rep("weekday", nrow(weekday_steps))

    data_by_weekdays <- rbind(weekend_steps, weekday_steps)
    data_by_weekdays$dayofweek <- as.factor(data_by_weekdays$dayofweek)
    data_by_weekdays
}

data_weekdays <- data_by_weekdays(rdata_fill)
```

Below you can see the panel plot comparing the average number of steps taken per 5-minute interval across weekdays and weekends:

```
ggplot(data_weekdays, aes(x=interval, y=steps)) +
        geom_line(color="blue") +
        facet_wrap(~ dayofweek, nrow=2, ncol=1) +
        labs(x="Interval", y="Number of steps") +
        theme_bw()
```



It looks like this person may have day job and does most of his or her walking on the weekends!

References

"Exploratory Data Analysis", by Roger D. Peng, PhD, Jeff Leek, PhD, Brian Caffo, PhD, Coursera. July 11, 2014. https://www.coursera.org/course/exdata "Reproducible Research", by Roger D. Peng, PhD, Jeff Leek, PhD, Brian Caffo, PhD, Coursera. Oct 22, 2014. https://www.coursera.org/course/repdata "Data Science Specialization", by Roger D. Peng, PhD, Jeff Leek, PhD, Brian Caffo, PhD, Coursera. July 11, 2014. https://www.coursera.org/specialization/jhudatascience/1 "Imputation in R - Stack Overflow", : http://stackoverflow.com/questions/13114812/imputation-in-r