Reproducible Research: Peer Assessment 1

Introduction

It is now possible to collect a large amount of data about personal movement using activity monitoring devices such as a Fitbit, Nike Fuelband, or Jawbone Up. These type of devices are part of the "quantified self" movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. But these data remain under-utilized both because the raw data are hard to obtain and there is a lack of statistical methods and software for processing and interpreting 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.

Data

The data for this assignment can be downloaded from the course web site:

Dataset: [Activity monitoring data] (https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2Factivity.zip)

The variables included in this dataset are:

- **steps:** Number of steps taking in a 5-minute interval (missing values are coded as NA)
- **date:** The date on which the measurement was taken in YYYY-MM-DD format
- **interval:** Identifier for the 5-minute interval in which measurement was taken

The dataset is stored in a comma-separated-value (CSV) file and there are a total of 17,568 observations in this dataset.

Solution

Setting global options and loading required libraries

```r
library(knitr)
library(ggplot2)
library(data.table)
opts_chunk$set(echo = TRUE, results = 'hold')
```

Loading the data

```
activity <- unzip("activity.zip")
act_data <- read.csv("activity.csv", header=TRUE, sep=",")
str(act_data)
```

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

Process the data

Convert some of the vectors to appropriate forms

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

Post converting the column classes print the structure of the data

```
str(act_data)
```

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

Print the header of the dataset

```
head(act_data, 10)
```

```
##    steps       date interval
## 1     NA 2012-10-01        0
## 2     NA 2012-10-01        5
## 3     NA 2012-10-01       10
## 4     NA 2012-10-01       15
## 5     NA 2012-10-01       20
## 6     NA 2012-10-01       25
## 7     NA 2012-10-01       30
## 8     NA 2012-10-01       35
## 9     NA 2012-10-01       40
## 10    NA 2012-10-01       45
```

What is mean total number of steps taken per day?

1. Calculate the total number of steps taken per day

```
steps_per_day <- aggregate(steps ~ date, data=act_data, FUN=sum)
colnames(steps_per_day) <- c("date", "steps")
```

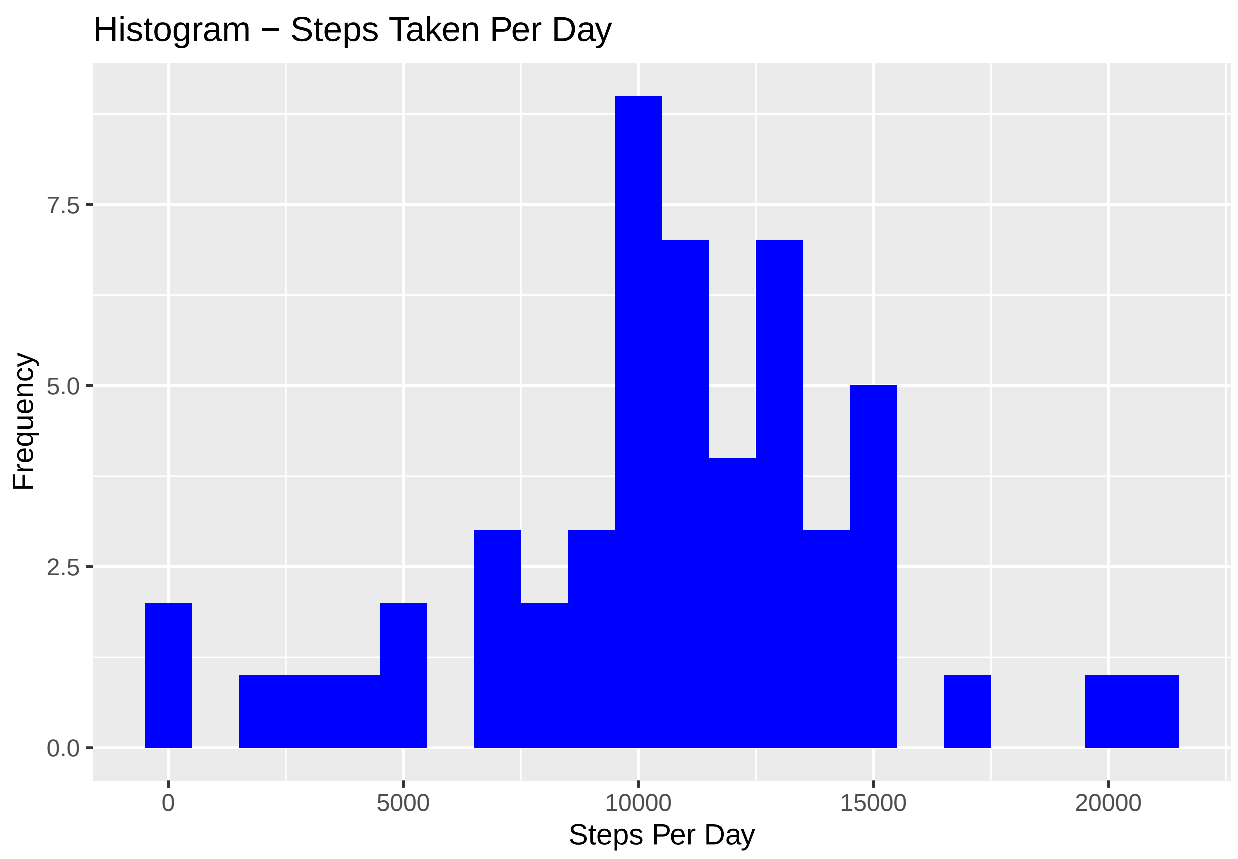
Print the header of the data frame with steps aggregated per day

```
head(steps_per_day, 10)
```

```
##          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
## 7  2012-10-09 12811
## 8  2012-10-10  9900
## 9  2012-10-11 10304
## 10 2012-10-12 17382
```

2. Make a histogram of the total number of steps taken each day

```
ggplot(steps_per_day, aes(x = steps)) +
  geom_histogram(fill = "blue", binwidth = 1000) +
  labs(title = "Histogram - Steps Taken Per Day", x = "Steps Per Day", y = "Frequency")
```



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

```
mean_steps_per_day <- mean(steps_per_day$steps)
mean_steps_per_day
median_steps_per_day <- median(steps_per_day$steps)
median_steps_per_day
```

```
## [1] 10766.19
## [1] 10765
```

What is the average daily activity pattern?

1. Make 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)

```
steps_per_interval <- aggregate(steps ~ interval, data = act_data, FUN = mean, na.rm = TRUE)
steps_per_interval$interval <- as.integer(levels(steps_per_interval$interval)[steps_per_interval$interva
colnames(steps_per_interval) <- c("interval", "steps")
```

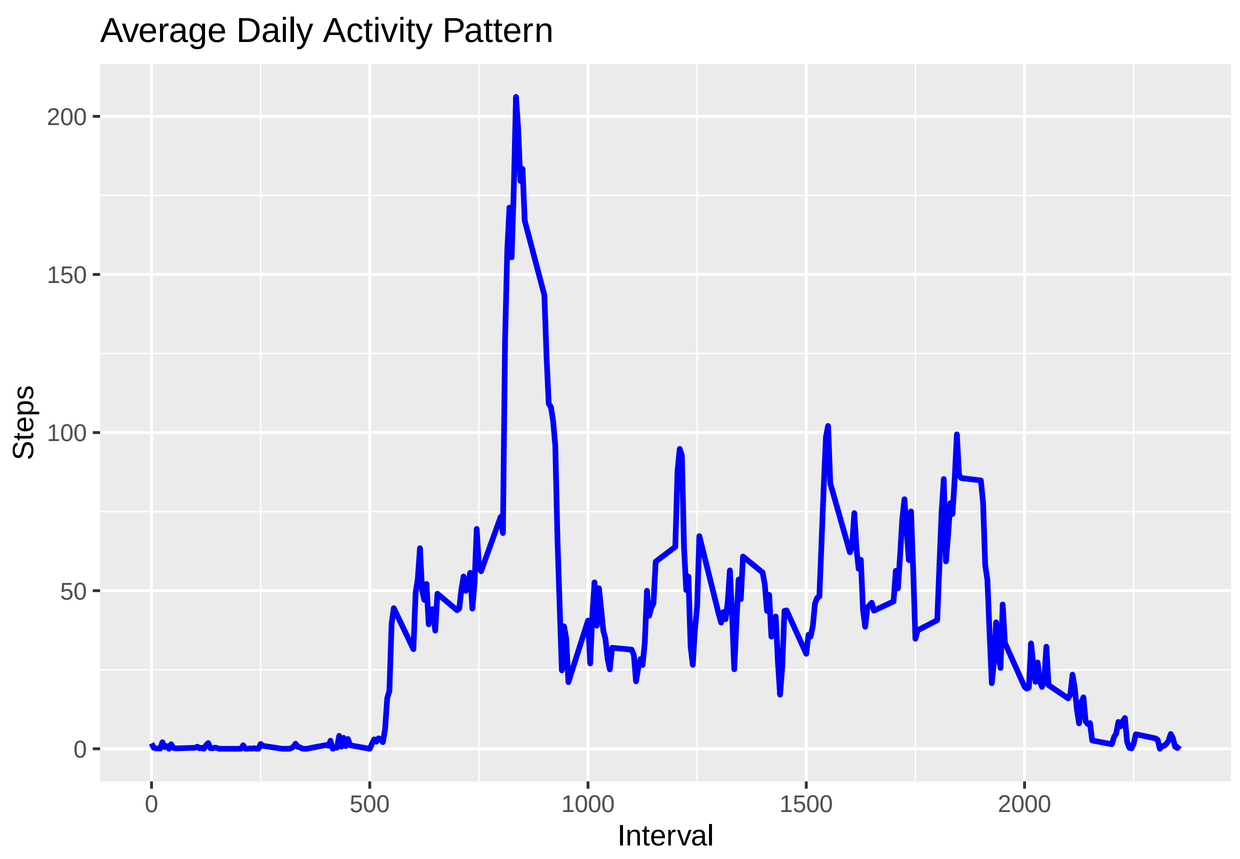
Print the header of the data frame with steps aggregated per interval

```
head(steps_per_interval, 10)
```

```
##    interval     steps
## 1         0 1.7169811
## 2         5 0.3396226
## 3        10 0.1320755
## 4        15 0.1509434
## 5        20 0.0754717
## 6        25 2.0943396
## 7        30 0.5283019
## 8        35 0.8679245
## 9        40 0.0000000
## 10       45 1.4716981
```

Plot the timeseries graph

```
ggplot(steps_per_interval, aes(x = interval, y = steps)) +
  geom_line(col = "blue", size = 1) +
  labs(title = "Average Daily Activity Pattern", x = "Interval", y = "Steps")
```



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

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

```
##     interval    steps
## 104      835 206.1698
```

Imputing missing values

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

```
missing_values <- sum(is.na(act_data$steps))
missing_values
```

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

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.

To populate missing values, we choose to replace them with the mean value at the same interval across days.

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

```r
new_act_data <- act_data
index_of_na <- which(is.na(new_act_data$steps))
for (i in index_of_na) {
  new_act_data$steps[i] <- with(steps_per_interval, steps[interval = new_act_data$interval[i]])
}
```

Print the top rows of newly created dataset

```r
head(new_act_data, 10)
```

```
##        steps       date interval
## 1  1.7169811 2012-10-01        0
## 2  0.3396226 2012-10-01        5
## 3  0.1320755 2012-10-01       10
## 4  0.1509434 2012-10-01       15
## 5  0.0754717 2012-10-01       20
## 6  2.0943396 2012-10-01       25
## 7  0.5283019 2012-10-01       30
## 8  0.8679245 2012-10-01       35
## 9  0.0000000 2012-10-01       40
## 10 1.4716981 2012-10-01       45
```

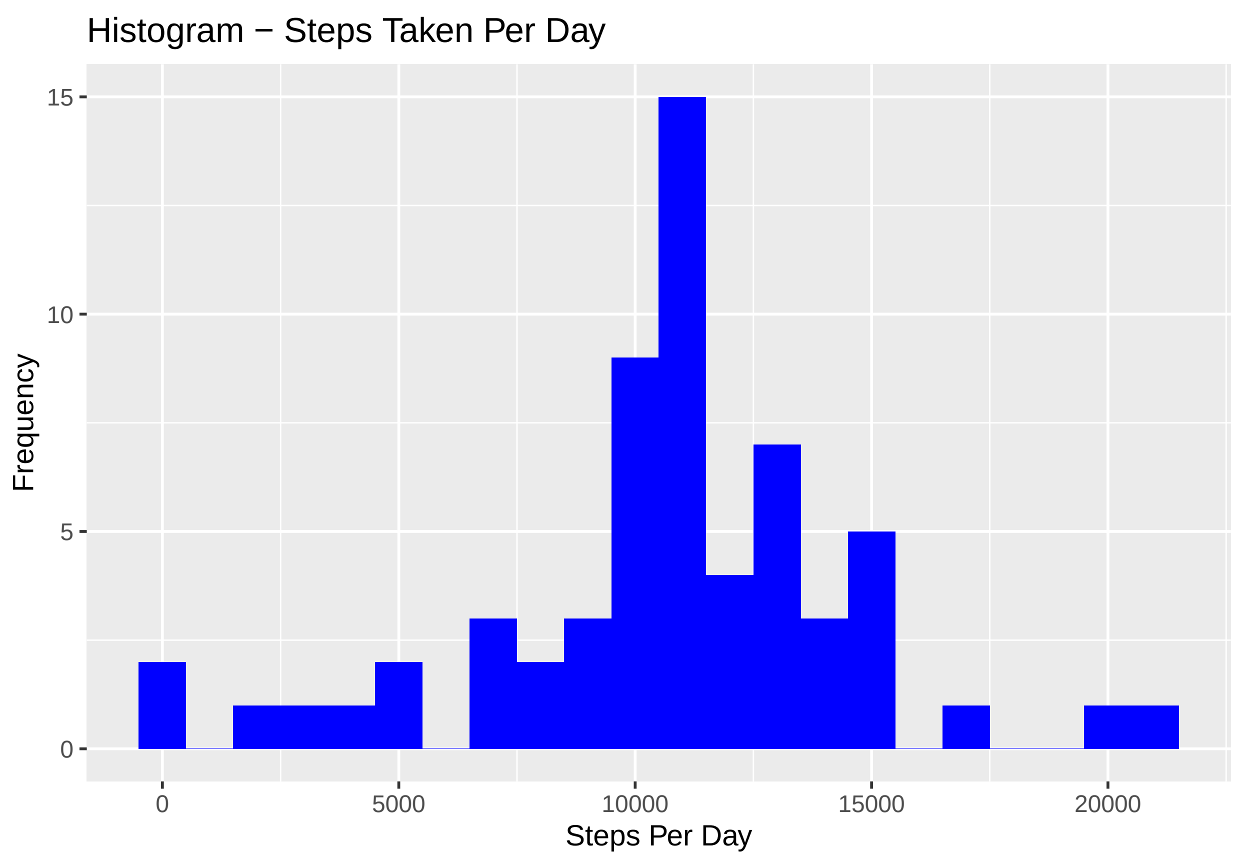
Given we have handled the missing values, let us check if the above strategy really worked out.

```r
new_missing_values <- sum(is.na(new_act_data$steps))
new_missing_values
```

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

4. 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?

```r
new_steps_per_day <- aggregate(steps ~ date, data = new_act_data, FUN=sum)
colnames(new_steps_per_day) <- c("date", "steps")
ggplot(new_steps_per_day, aes(x = steps)) +
  geom_histogram(fill = "blue", binwidth = 1000) +
  labs(title = "Histogram - Steps Taken Per Day", x = "Steps Per Day", y = "Frequency")
```



In order to find the impact of imputing the missing values, let us compute the mean and median of steps taken per day

```
new_mean_steps_per_day <- mean(new_steps_per_day$steps)
new_mean_steps_per_day
new_median_steps_per_day <- median(new_steps_per_day$steps)
new_median_steps_per_day
```

```
## [1] 10766.19
## [1] 10766.19
```

Are there differences in activity patterns between weekdays and weekends?

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.

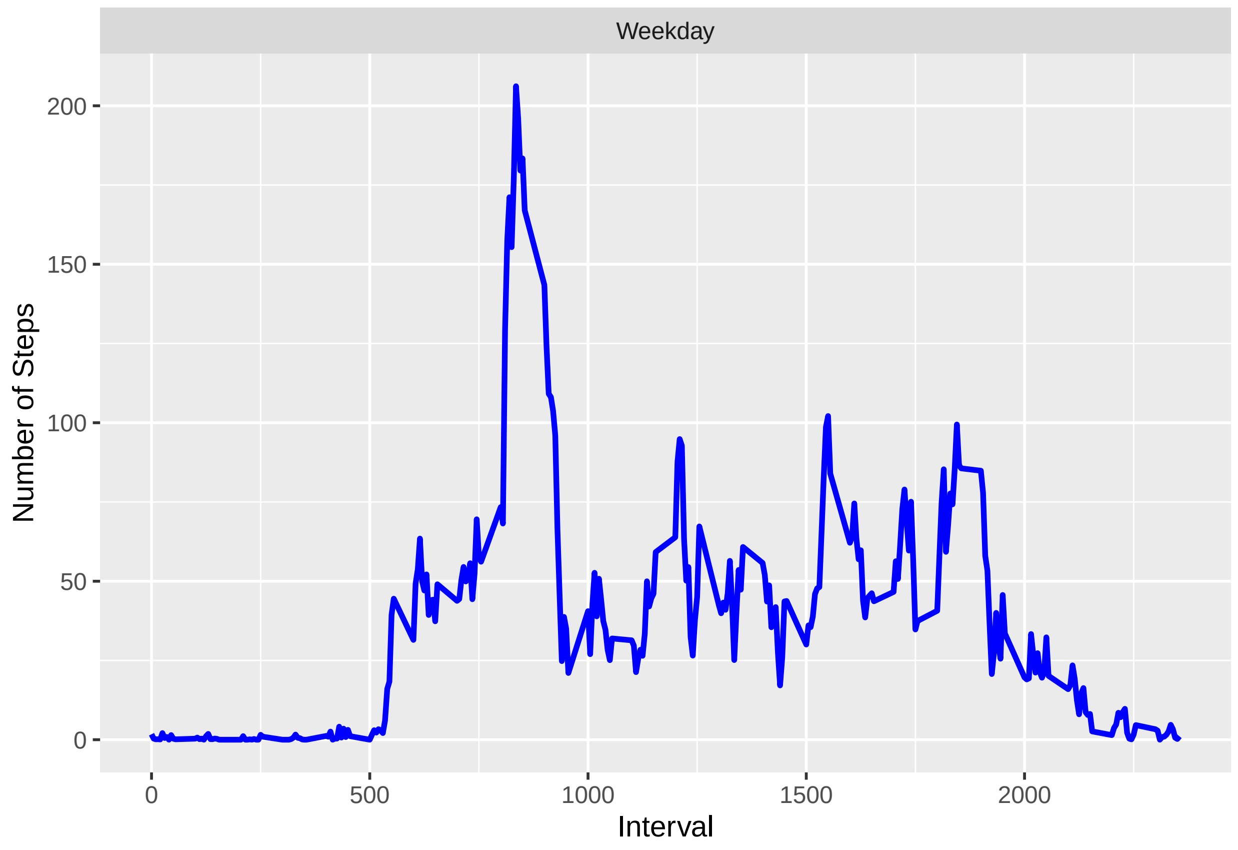
Let us first add a factor variable to identify the given date as Weekday or Weekend

```
dt <- data.table(new_act_data)
dt[, weekday := ifelse(weekdays(date) %in% c("Saturday", "Sunday"), "Weekend", "Weekday")]
dt$weekday <- as.factor(dt$weekday)
dt$interval <- as.integer(levels(dt$interval)[dt$interval])
head(dt, 10)
```

```
##         steps       date interval weekday
##  1: 1.7169811 2012-10-01        0 Weekday
##  2: 0.3396226 2012-10-01        5 Weekday
##  3: 0.1320755 2012-10-01       10 Weekday
##  4: 0.1509434 2012-10-01       15 Weekday
##  5: 0.0754717 2012-10-01       20 Weekday
##  6: 2.0943396 2012-10-01       25 Weekday
##  7: 0.5283019 2012-10-01       30 Weekday
##  8: 0.8679245 2012-10-01       35 Weekday
##  9: 0.0000000 2012-10-01       40 Weekday
## 10: 1.4716981 2012-10-01       45 Weekday
```

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)

```
steps_per_weekday <- aggregate(steps ~ interval+weekday, data = dt, FUN = mean)
ggplot(steps_per_weekday, aes(x = interval, y = steps)) +
  geom_line(col = "blue", size = 1) +
  facet_wrap(~ weekday, nrow=2, ncol=1) +
  labs(x = "Interval", y = "Number of Steps")
```


Weekday
Number of Steps
Interval