Reproducible Research Course Project Week 2

Executive Summary

This project analyzes a dataset from apersonal activity monitoring device. The 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.

The data was downloaded from the original source. R was used to process and visualize the data. Three additional libraries were loaded: `dplyr`, `ggplot2`, and `timeDate`. Some entries in the dataset contain `NA`. They were filled with the mean values of the specific date. Results did not change significantly after the replacement.

The results also showed differences in the steps made during the time interval, weekday and weekend.

Libraries used in this project

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)
library(timeDate)
```

1. Code for reading in the dataset and/or processing the data

```r
download.file(
   url = 'https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2Factivity.zip',
   destfile = 'activity.zip',
   method = 'auto'
   )
unzip(
   'activity.zip',
   exdir='inputdir'
   )
input.raw.data <- read.csv(
   'inputdir/activity.csv',
   sep = ',',
   header = TRUE
   )
```

2. Histogram of the total number of steps taken each day

```
grouped.data <- input.raw.data %>% group_by(date)
grouped.data <- grouped.data %>% summarise(steps=sum(steps))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

Checking the content of the data and its dimension, and determine the total number of steps in each day:

```
head(grouped.data)
```

```
## # A tibble: 6 x 2
##   date       steps
##   <fct>      <int>
## 1 2012-10-01    NA
## 2 2012-10-02   126
## 3 2012-10-03 11352
## 4 2012-10-04 12116
## 5 2012-10-05 13294
## 6 2012-10-06 15420
```

```
dim(grouped.data)
```

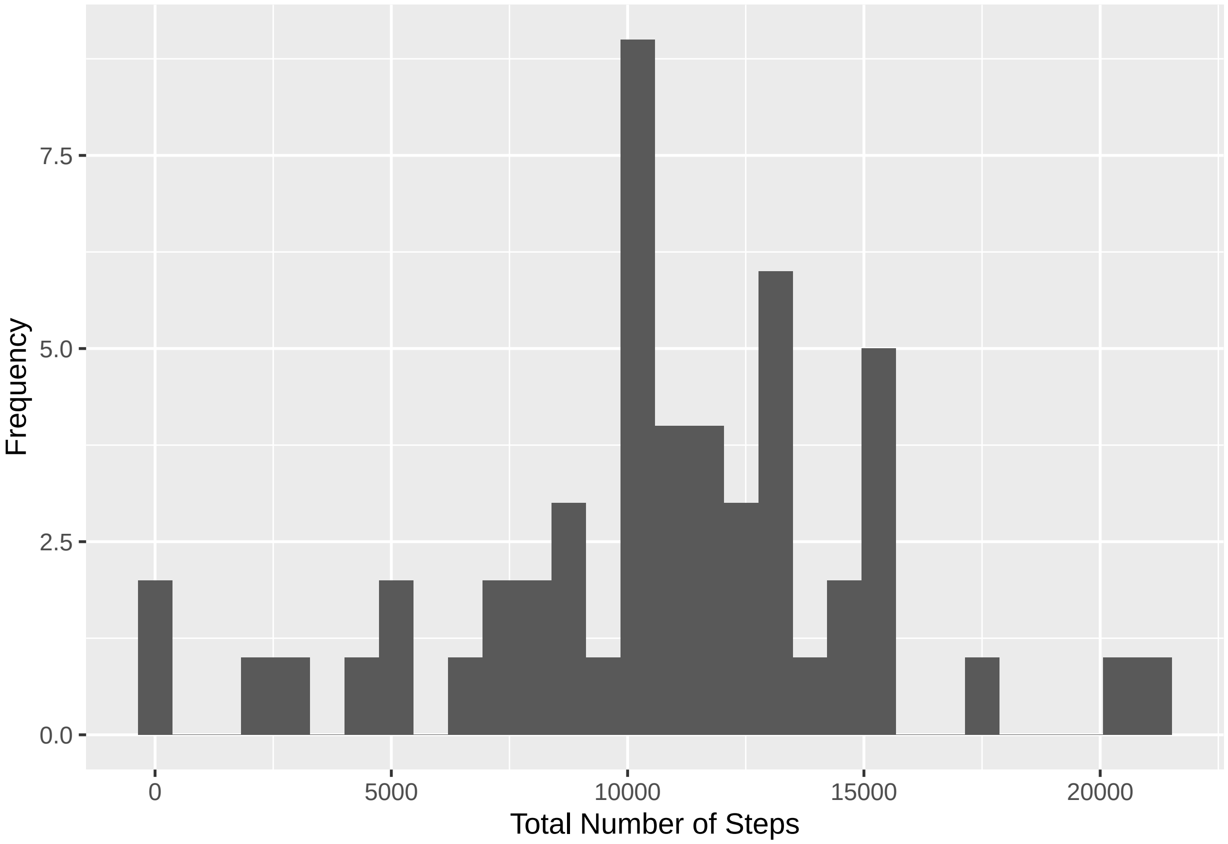
```
## [1] 61  2
```

Plotting the histogram:

```
processed.data <- data.frame(
   date = grouped.data$date,
   total.steps = grouped.data$steps
   )
ggplot(processed.data,
   aes(total.steps)
   ) +
   geom_histogram() +
   xlab('Total Number of Steps') +
   ylab('Frequency')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 8 rows containing non-finite values (stat_bin).
```



3. Mean and median number of steps taken each day

From the object processed.data:

```
mean.value <-  mean(processed.data$total.steps, na.rm=TRUE)
median.value <-  median(processed.data$total.steps, na.rm=TRUE)

print(mean.value)
```

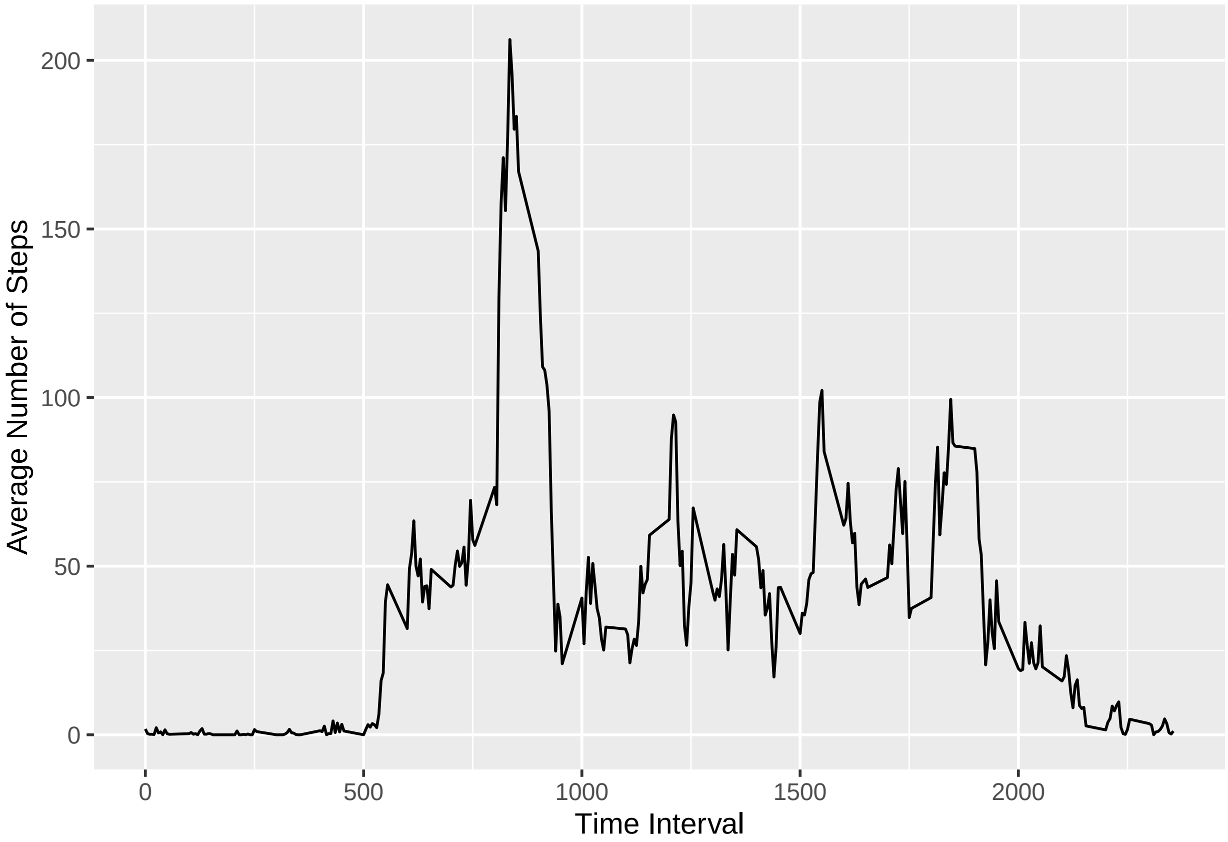
```
## [1] 10766.19
```

```
print(median.value)
```

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

4. Time series plot of the average number of steps taken

```
interval.data <- aggregate(steps ~ interval, input.raw.data, mean)
ggplot(
   interval.data, aes(x=interval, y=steps)
   ) + geom_line() +
   xlab('Time Interval') + ylab('Average Number of Steps')
```



5. The 5-minute interval that, on average, contains the maximum number of steps

By looking the previous plot, the highest peak is around the 800 range.

```
output <- interval.data[which.max(interval.data$steps),]
print(paste('Interval ', output[1,1], sep = ' '))
```

```
## [1] "Interval  835"
```

```
print(paste('Average Number of Steps ', output[1,2], sep = ' '))
```

```
## [1] "Average Number of Steps  206.169811320755"
```

6. Code to describe and show a strategy for imputing missing data

One possibility is to remove all the NA from the dataset:

```
clean.raw.data <- input.raw.data[complete.cases(input.raw.data),]
head(clean.raw.data)
```

```
##     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
```

Another possibility is to fill them in with a value such as the mean or median. Choosing the mean:

```
clean.raw.data <- transform(input.raw.data,
   steps = ifelse(is.na(steps),
      ave(steps, interval, FUN = function(x) mean(x, na.rm = TRUE)),
      steps)
   )
```

7. Histogram of the total number of steps taken each day after missing values are imputed

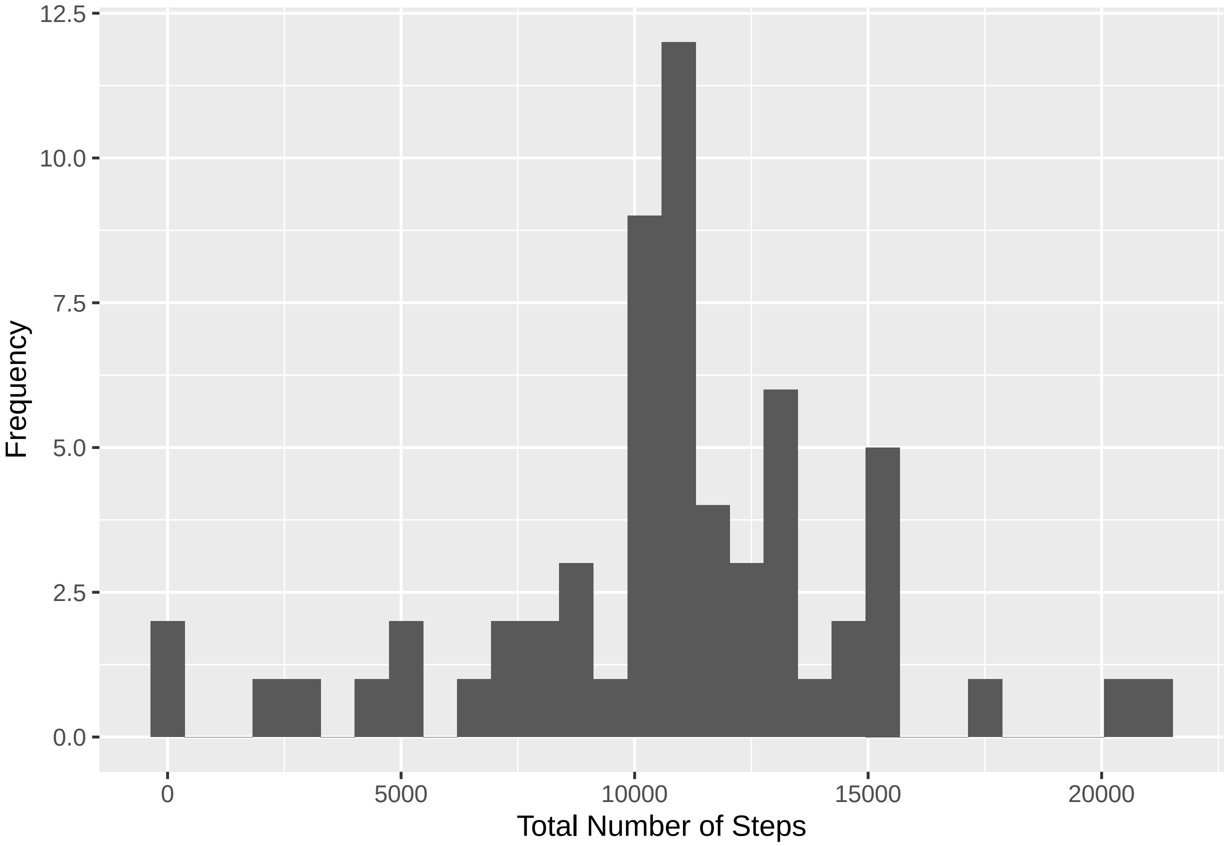
Plotting the histogram after cleaning the data:

```
grouped.data <- clean.raw.data %>% group_by(date)
grouped.data <- grouped.data %>% summarise(steps=sum(steps))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
processed.data <- data.frame(
   date = grouped.data$date,
   total.steps = grouped.data$steps
   )
ggplot(processed.data,
   aes(total.steps)
   ) +
   geom_histogram() +
   xlab('Total Number of Steps') +
   ylab('Frequency')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



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?

No change with respect to the mean but it slightly changed the value of the median.

```
mean.value <-  mean(processed.data$total.steps, na.rm=TRUE)
median.value <-  median(processed.data$total.steps, na.rm=TRUE)

print(mean.value)
```

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

```
print(median.value)
```

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

8. Panel plot comparing the average number of steps taken per 5-minute interval across weekdays and weekends

Start from input raw data:

```
clean.raw.data <- transform(input.raw.data,
   steps = ifelse(is.na(steps),
      ave(steps, interval, FUN = function(x) mean(x, na.rm = TRUE)),
      steps
      )
   )
head(clean.raw.data)
```

```
##       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
```

Determine which row is a weekday (TRUE or FALSE):

```
weekday.list <- mutate(
   clean.raw.data,
   weekday.checker = isWeekday(clean.raw.data$date, wday=1:5)
   )
aggregated.data <- aggregate(
   steps ~ interval + weekday.checker,
   weekday.list,
   mean
   )

head(aggregated.data)
```

```
##   interval weekday.checker       steps
## 1        0           FALSE 0.214622642
## 2        5           FALSE 0.042452830
## 3       10           FALSE 0.016509434
## 4       15           FALSE 0.018867925
## 5       20           FALSE 0.009433962
## 6       25           FALSE 3.511792453
```

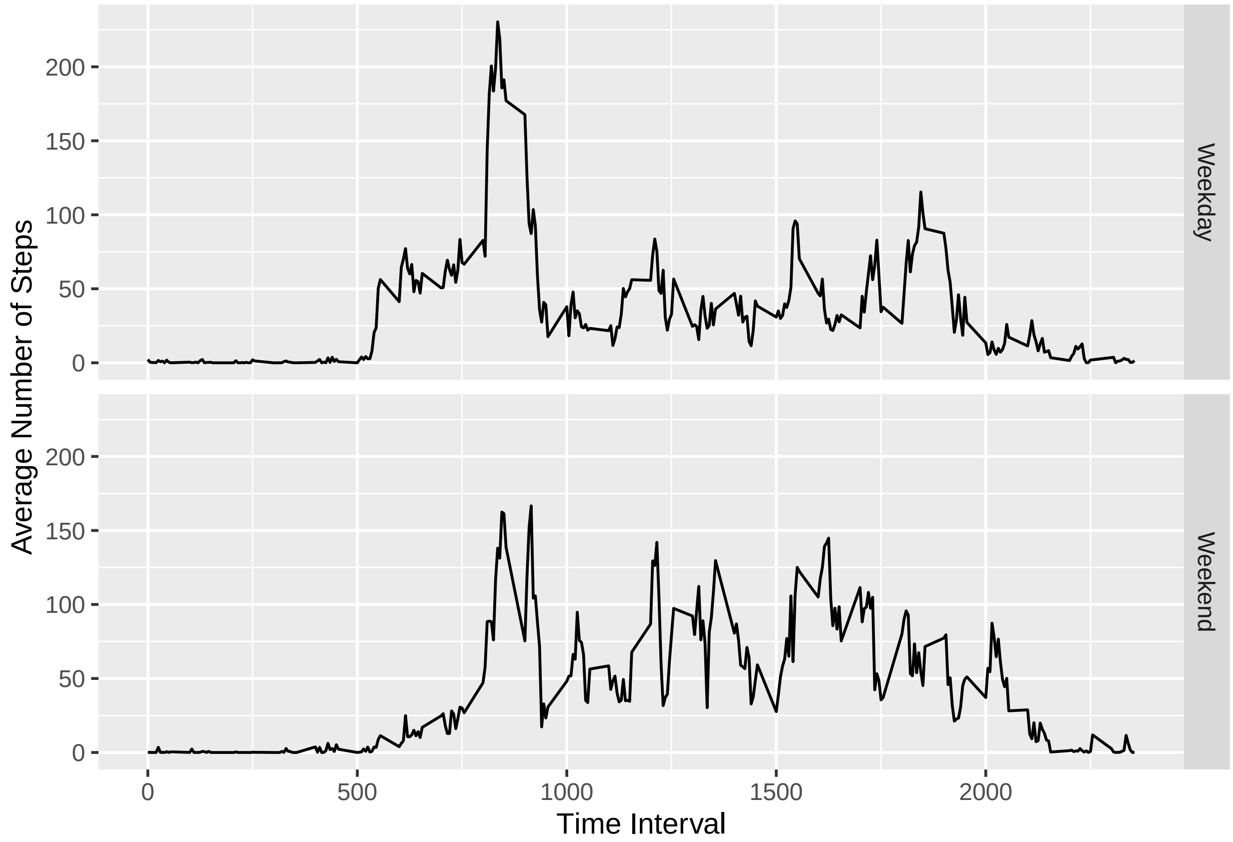
```
transformed.data <- transform(
   aggregated.data,
   weekend.checker = ifelse(
      aggregated.data$weekday.checker == 'FALSE',
      'Weekend',
      'Weekday'
      )
   )

head(transformed.data)
```

```
##   interval weekday.checker       steps weekend.checker
## 1        0           FALSE 0.214622642         Weekend
## 2        5           FALSE 0.042452830         Weekend
## 3       10           FALSE 0.016509434         Weekend
## 4       15           FALSE 0.018867925         Weekend
## 5       20           FALSE 0.009433962         Weekend
## 6       25           FALSE 3.511792453         Weekend
```

Making a panel plot:

```
ggplot(transformed.data, aes(x=interval, y=steps)) +
   geom_line() +
   xlab('Time Interval') + ylab('Average Number of Steps') +
   facet_grid(weekend.checker ~ .)
```



9. All of the R code needed to reproduce the results (numbers, plots, etc.) in the report

```
# Loading Library:
library(dplyr)
library(ggplot2)
library(timeDate)

# Dowloading the dataset and loading to the console:
download.file(
   url = 'https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2Factivity.zip',
   destfile = 'activity.zip',
   method = 'auto'
   )
unzip(
   'activity.zip',
   exdir='inputdir'
   )
input.raw.data <- read.csv(
   'inputdir/activity.csv',
   sep = ',',
   header = TRUE
   )
```

```
# First Histogram:
grouped.data <- input.raw.data %>% group_by(date)
grouped.data <- grouped.data %>% summarise(steps=sum(steps))
processed.data <- data.frame(
   date = grouped.data$date,
   total.steps = grouped.data$steps
   )
ggplot(processed.data,
   aes(total.steps)
   ) +
   geom_histogram() +
   xlab('Total Number of Steps') +
   ylab('Frequency')

mean.value <-  mean(processed.data$total.steps, na.rm=TRUE)
median.value <-  median(processed.data$total.steps, na.rm=TRUE)
print(mean.value)
print(median.value)


interval.data <- aggregate(steps ~ interval, input.raw.data, mean)
ggplot(
   interval.data, aes(x=interval, y=steps)
   ) + geom_line() +
   xlab('Time Interval') + ylab('Average Number of Steps')


# Processing the NA:
clean.raw.data <- transform(input.raw.data,
   steps = ifelse(is.na(steps),
      ave(steps, interval, FUN = function(x) mean(x, na.rm = TRUE)),
      steps)
   )

grouped.data <- clean.raw.data %>% group_by(date)
grouped.data <- grouped.data %>% summarise(steps=sum(steps))
processed.data <- data.frame(
   date = grouped.data$date,
   total.steps = grouped.data$steps
   )
ggplot(processed.data,
   aes(total.steps)
   ) +
   geom_histogram() +
   xlab('Total Number of Steps') +
   ylab('Frequency')

mean.value <-  mean(processed.data$total.steps, na.rm=TRUE)
median.value <-  median(processed.data$total.steps, na.rm=TRUE)

# Making the Panel Plot:

clean.raw.data <- transform(input.raw.data,
   steps = ifelse(is.na(steps),
      ave(steps, interval, FUN = function(x) mean(x, na.rm = TRUE)),
      steps
      )
   )

# Determine which row is a weekday (TRUE or FALSE):
weekday.list <- mutate(
   clean.raw.data,
   weekday.checker = isWeekday(clean.raw.data$date, wday=1:5)
   )
aggregated.data <- aggregate(
   steps ~ interval + weekday.checker,
   weekday.list,
   mean
   )

transformed.data <- transform(
   aggregated.data,
   weekend.checker = ifelse(
      aggregated.data$weekday.checker == 'FALSE',
      'Weekend',
      'Weekday'
      )
   )

ggplot(transformed.data, aes(x=interval, y=steps)) +
   geom_line() +
   xlab('Time Interval') + ylab('Average Number of Steps') +
   facet_grid(weekend.checker ~ .)
```