Reproducible Research: Peer Assessment 1

Loading and preprocessing the data

Unzip the data into "activity.csv" file and Load the data into "activity" dataFrame See the dimensions, head, structure of the activity dataFrame

```
if(!file.exists("activity.csv")) {
  unzip("reproducible_research_week2_project/RepData_PeerAssessment1/activity.zip", exdir = "reproducibl
}
activity = read.csv("activity.csv")
```

```
dim(activity)
```

```
## [1] 17568     3
```

```
head(activity)
```

```
##   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
```

```
str(activity)
```

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

Transforming the class of activity$date to Date format

```
activity$date = as.Date(as.character(activity$date), "%Y-%m-%d")
str(activity)
```

```
## '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: int  0 5 10 15 20 25 30 35 40 45 ...
```

What is mean total number of steps taken per day?

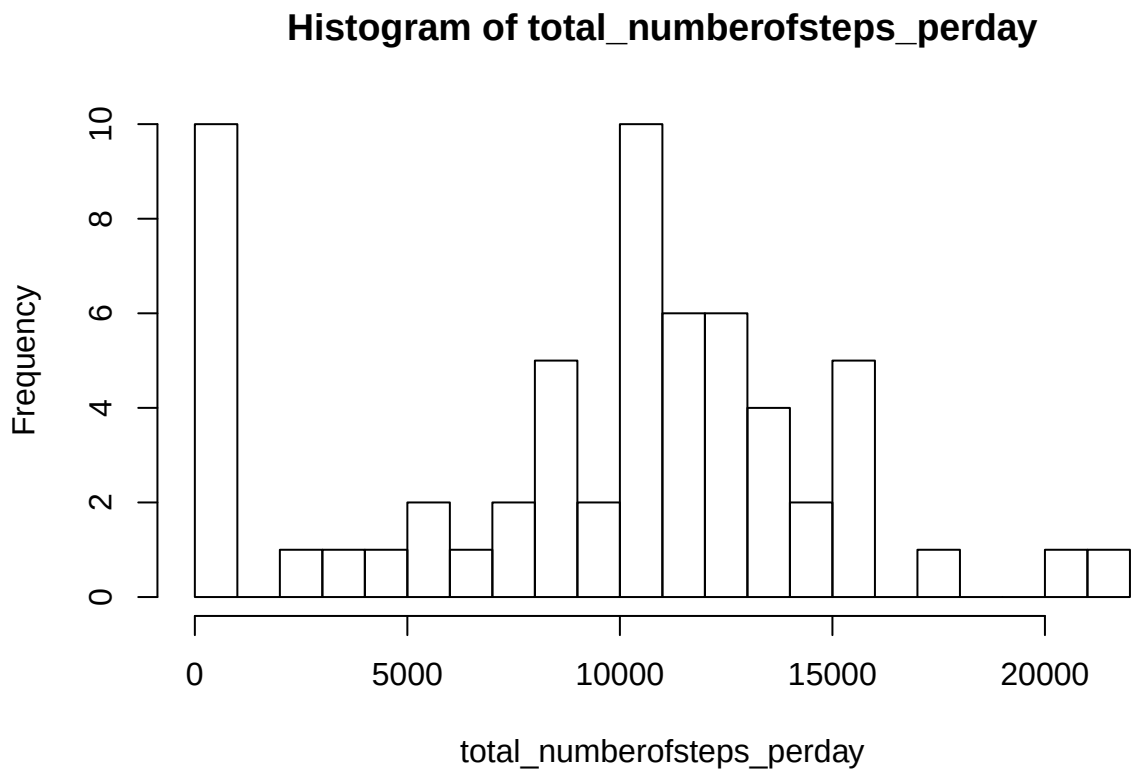
Lets calculate total number of steps taken per day using tapply excluding NAs and see some values using head

```
total_numberofsteps_perday = with(activity, tapply(steps, date, sum, na.rm = TRUE))
head(total_numberofsteps_perday)
```

```
## 2012-10-01 2012-10-02 2012-10-03 2012-10-04 2012-10-05 2012-10-06
##          0        126      11352      12116      13294      15420
```

Lets make a histogram of the total number of steps taken each day

```
hist(total_numberofsteps_perday,breaks = 20)
```



Lets calculate and report the mean and median of the total number of steps taken per day

```
mean(total_numberofsteps_perday)
```

```
## [1] 9354.23
```

```
median(total_numberofsteps_perday)
```

```
## [1] 10395
```

What is the average daily activity pattern?

Lets 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)

Lets store x-axis data in fivemin_interval variable using unique function on activity$interval

```
fivemin_interval = unique(activity$interval)
str(fivemin_interval)
```

```
##  int [1:288] 0 5 10 15 20 25 30 35 40 45 ...
```

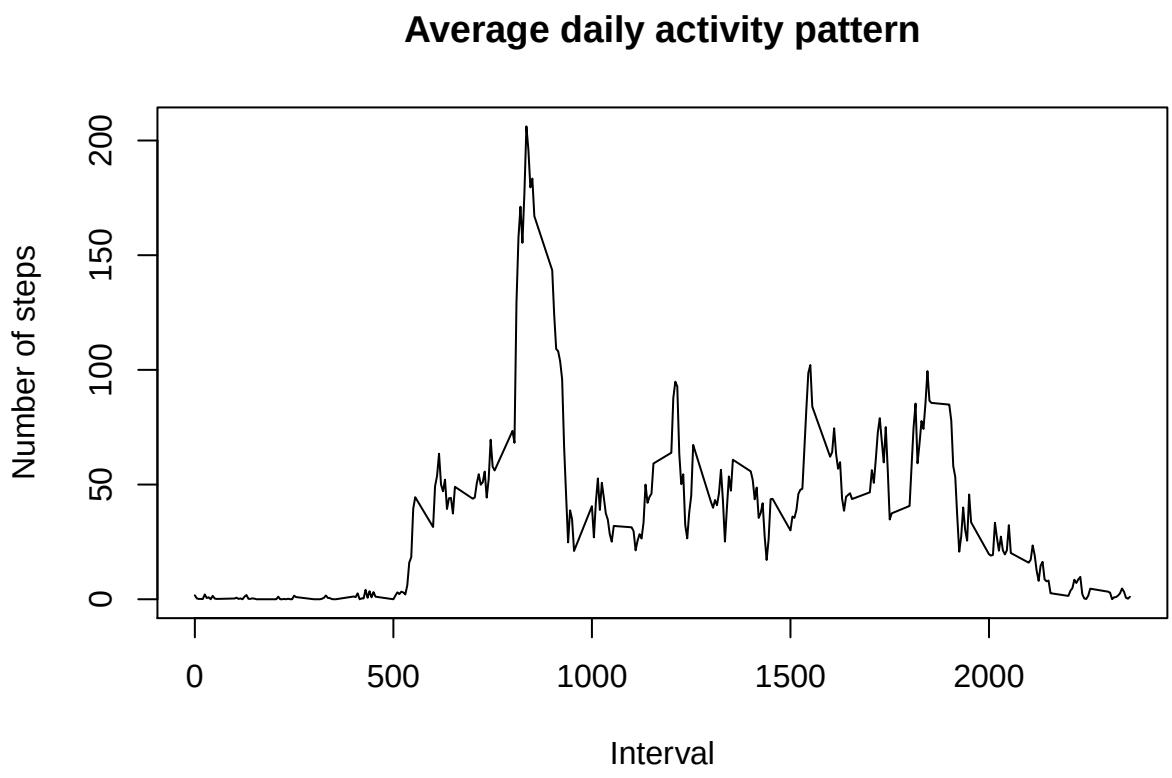
And lets store y-axis data as below

```
averagenumberofstepstaken_averagedacrossalldays = with(activity,tapply(steps, interval, mean, na.rm=TRU
str(averagenumberofstepstaken_averagedacrossalldays)
```

```
##  num [1:288(1d)] 1.717 0.3396 0.1321 0.1509 0.0755 ...
##  - attr(*, "dimnames")=List of 1
##   ..$ : chr [1:288] "0" "5" "10" "15" ...
```

And here comes the plot

```
#png(filename = "instructions_fig/plot1.png")
plot(fivemin_interval,averagenumberofstepstaken_averagedacrossalldays,
     type = "l", xlab = "Interval" , ylab = "Number of steps", main = "Average daily activity pattern" )
```



```
#dev.off()
```

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

Lets print averagenumberofstepstaken_averagedacrossalldays in ascending order and get the index of its maximum.

```
order(averagenumberofstepstaken_averagedacrossalldays)
```

```
##   [1]   9  17  24  25  26  28  29  31  33  34  37  38  39  40  47  48  52
##  [18]  61 279   5  46 274   3  12  30   4  15  20  21  41  32 287  23  11
##  [35]  13 273   2  16  53  54  22  45   7  44  42 286  56  14  58 280   8
##  [52]  36  50 281 288  18  60  27  49 265  10  35  62 282 275  43   1  19
##  [69]   6  67 272  64  51 283 264 278  66  63  59 277 285  65  57 266  55
##  [86] 276 284 267  68 269 262 258 263 268 261 270 271 257 259 253  69 260
## [103] 177 254  70 242 256 243 249 241 252 234 120 246 135 250 248 255 117
## [120] 131 164 136 238 178 138 153 245 122 247 235 176 130 137 134 181 237
## [137] 133  73 132 251 152 244 139 240 129 215 119 173 183 182 233  83 128
## [154] 216 174 154 200 118 184 125  80  71 158 165 236 121 217 160 175 141
## [171] 157 123 163 159 171 179 204 180  85 199  81  82 127  92  86  72 142
## [188] 201 155 116 202 239 185 143 203 161 205  78 167 186 187 172  84  74
## [205]  89  77 140 150  87 207 126  90 170  79  93 124 232 166  75 151  88
## [222]  91 169  96 206 162 214 197  95 218 231 144 221 212 198 168 208 193
## [239] 196 149  76 145 194 188 115 156 222  98 211  94 209  97 224 195 219
## [256] 213 223 230 210 189 192 229 220 225 228 227 146 148 147 114 190 226
## [273] 191 113 112 111 110  99 109 102 100 108 101 103 106 107 105 104
```

Its largest value is in index 104. Lets see the interval and maximum average steps

```
averagenumberofstepstaken_averagedacrossalldays[104]
```

```
##      835
## 206.1698
```

Imputing missing values

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

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

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

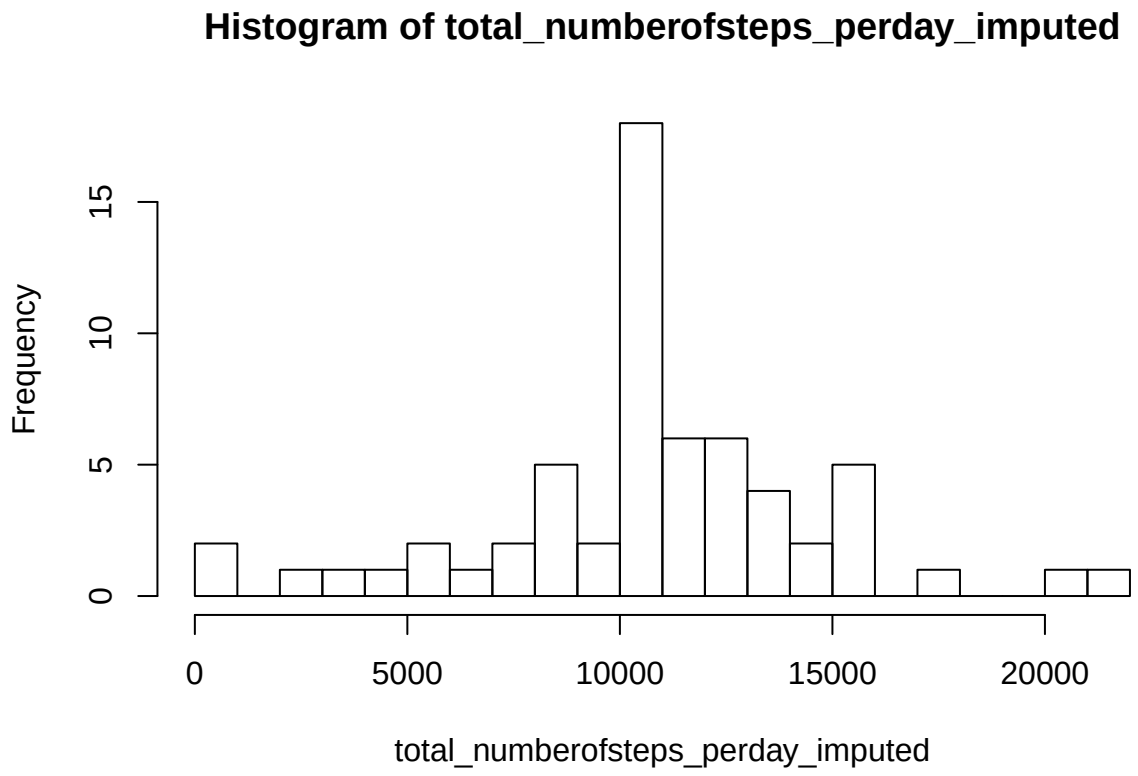
Lets devise a strategy for filling in all of the missing values in the dataset. Here lets impute missing value with mean of that interval and lets create a new dataset (activity_impute) that is equal to the original dataset but with the missing data filled in.

```
activity_imputed= activity
missingData = is.na(activity$steps)
meanValuesByInterval = tapply(activity$steps, activity$interval, mean, na.rm = TRUE)
activity_imputed$steps[missingData] =  meanValuesByInterval[as.character(activity_imputed$interval[miss
sum(is.na(activity_imputed))
```

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

Lets 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.

```
total_numberofsteps_perday_imputed = with(activity_imputed, tapply(steps, date, sum))
hist(total_numberofsteps_perday_imputed, breaks = 20)
```




```
mean(total_numberofsteps_perday_imputed)
```

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

```
median(total_numberofsteps_perday_imputed)
```

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

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?

```
mean(total_numberofsteps_perday)
```

```
## [1] 9354.23
```

```
mean(total_numberofsteps_perday_imputed)
```

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

Yes, Mean differs

```
median(total_numberofsteps_perday)
```

```
## [1] 10395
```

```
median(total_numberofsteps_perday_imputed)
```

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

Yes, Median differs

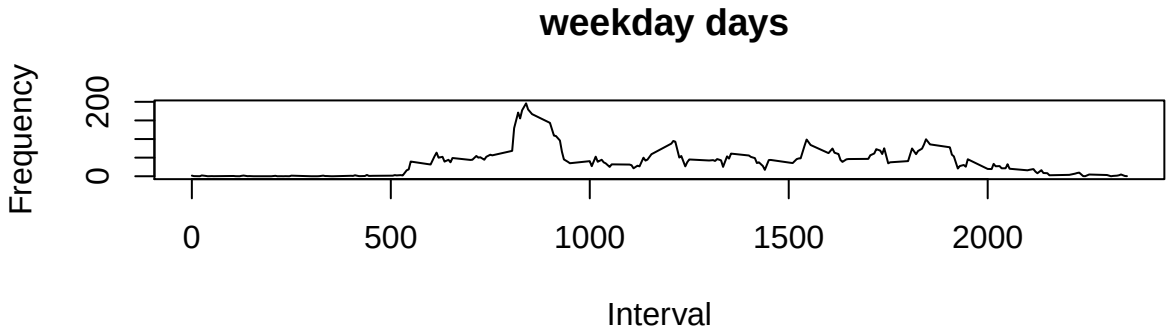
Are there differences in activity patterns between weekdays and weekends? Lets use dataset (activity_imputed) with filled-in missing values

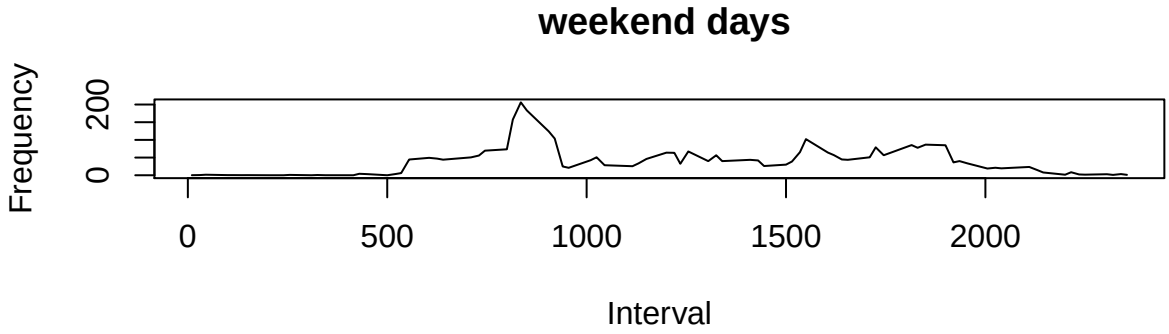
Lets create a new factor variable in the dataset with two levels - "weekday" and "weekend" indicating whether a given date is a weekday or weekend day.

```
library(chron)
week_factor = is.weekend(activity_imputed$interval)
activity_imputed_weekend = activity_imputed[week_factor,]
activity_imputed_weekday = activity_imputed[!week_factor,]
average_numberofsteps_perday_imputed_weekday = with(activity_imputed_weekday, tapply(steps, interval, m
average_numberofsteps_perday_imputed_weekend = with(activity_imputed_weekend, tapply(steps, interval, m
```

_ Lets 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(unique(activity_imputed_weekday$interval) , average_numberofsteps_perday_imputed_weekday,
     type="l" , xlab = "Interval",ylab = "Frequency", main = " weekday days")
plot(unique(activity_imputed_weekend$interval) , average_numberofsteps_perday_imputed_weekend,
     type="l", xlab = "Interval",ylab = "Frequency", main = " weekend days")
```





From the plots we can assumer that there is the hike in steps beginning from the start of the day in weekdays and in weekends the average is shared equally among the intervals compared to the weekdays