Reproducible Research: Peer Assessment 1

K. Grace Kennedy

Loading and preprocessing the data

Variable description and initial data load

```
options(scipen=999)#Gets rid of scientific notation
setwd("~/Dropbox/0Coursera_DataAnalysisR/5ReproducibleResearch/Projects/Project1ActivityMonitor/RepData_
rawdata=read.csv("../activity.csv")
head(rawdata)
```

```
##   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
```

The variables included in the original dataset are:
steps (Class: integer): The number of steps in the time interval
date (Class factor): Date of the observation
interval (Class: factor): Start of 5-minute time interval from 0 (midnight) to 2355 (11:55PM)

Tagging observations with day fo the week and day type

For future analysis, we want to be able to call observations by day of the week and by whether or not it is a weekday or weekend. The last part of the following code (last `else` section) is checking that the assignment of weekday/weekend ran well. It keeps track of any unassigned observations to be printed after the code chunk.

```
rawdata[[2]]=as.Date(rawdata[[2]])
dayOfWk=as.factor(weekdays(rawdata[[2]]))
data=data.frame(rawdata,dayOfWk)
weekdays=c("Monday","Tuesday","Wednesday","Thursday","Friday")
weekend=c("Saturday","Sunday")
k=1
x=c("All days were classified as weekday or weekend.")
check=x
for (i in 1:length(data[[1]])){
        if (data$dayOfWk[i] %in% weekdays){
                data[i,5]="weekday"
        } else if (data$dayOfWk[i] %in% weekend) {
                data[i,5]="weekend"
        } else {
                check=x
                x[k]=data[i,4]
                k=k+1
                if (check!=x){
```

```
                        check=c("There were days of the week that did
                                not get identified as weekend or weekday.
                                 Check the variable x.")
                }
        }
}
data[[5]]=as.factor(data[[5]])
names(data)[5]="dayType"
head(data)
```

```
##   steps       date interval dayOfWk dayType
## 1    NA 2012-10-01        0  Monday weekday
## 2    NA 2012-10-01        5  Monday weekday
## 3    NA 2012-10-01       10  Monday weekday
## 4    NA 2012-10-01       15  Monday weekday
## 5    NA 2012-10-01       20  Monday weekday
## 6    NA 2012-10-01       25  Monday weekday
```

All days were classified as weekday or weekend. The new variables are
date (Updated to Class: Date)
dayOfWk (Class: factor)
dayType (Class: factor)

What is mean total number of steps taken per day?

For this question we do not consider missing values.

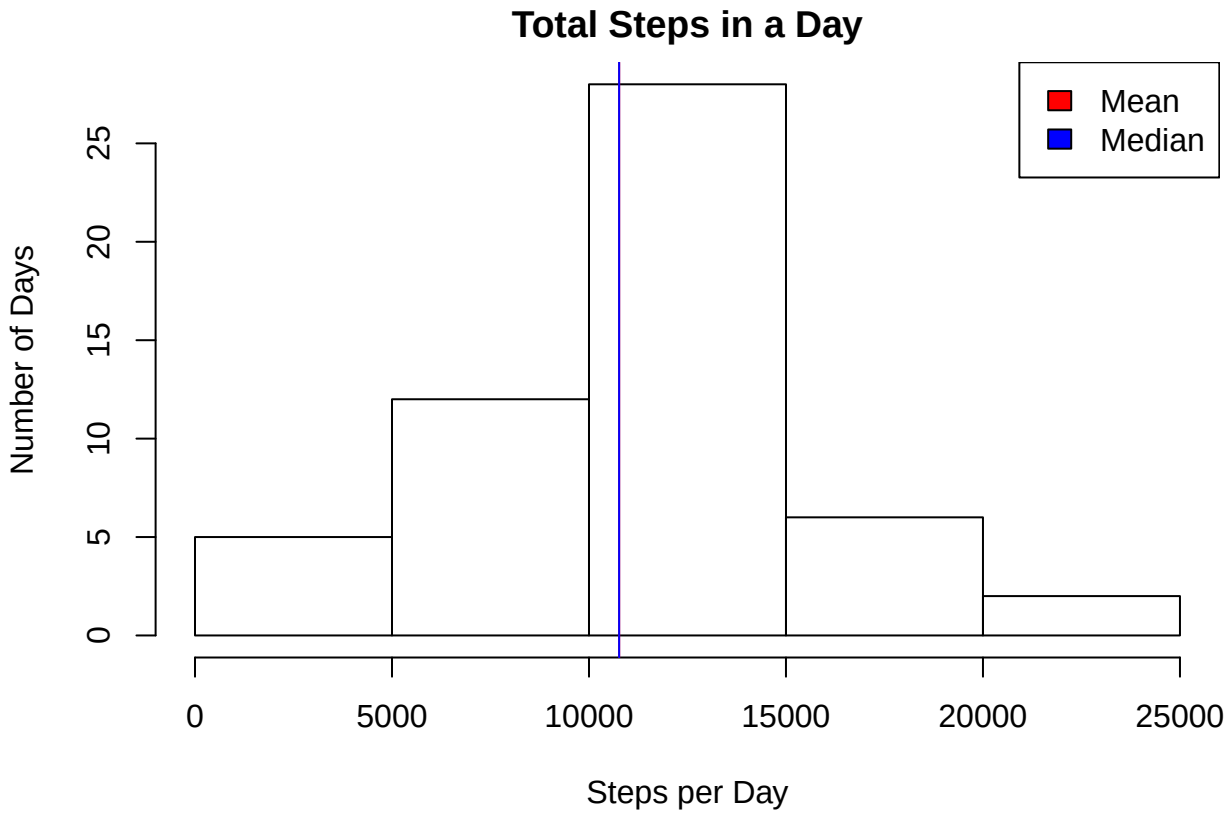
```
steppingDays=data[!is.na(data$steps),]
stepsByDay=aggregate(steppingDays$steps,by=list(Date=steppingDays$date),sum)
```

Statistic	Value	Code
Mean	10766.19	`round(mean(stepsByDay[[2]]),digits=2)`
Median	10765	`median(stepsByDay[[2]])`

The mean and the median are so close relative to the range of steps in a day that we cannot distinguish them on the histogram below.

```
par(mfrow=c(1, 1), mar=c(5, 4, 2, .8))
hist(stepsByDay[[2]],
     main="Total Steps in a Day",
     xlab="Steps per Day",
     ylab="Number of Days",
     )
legend("topright",
       legend=c("Mean","Median"),
       fill=c("red","blue"))
abline(
        v=c(mean(stepsByDay[[2]]),median(stepsByDay[[2]])),
```

```
        col=c("red","blue")
)
```



A barplot would show us the number of steps for each individual day rather than a histogram of the total of number of steps per day, which gives us a sense for the distribution for the total number of steps per day. The individual bars are ordered by rows, not necessarily chronologically. So we need to check that the observations are actually in chronological order, which they are. However there are definitely days for which no data was recorded.

```
!is.unsorted(stepsByDay[[1]])
```

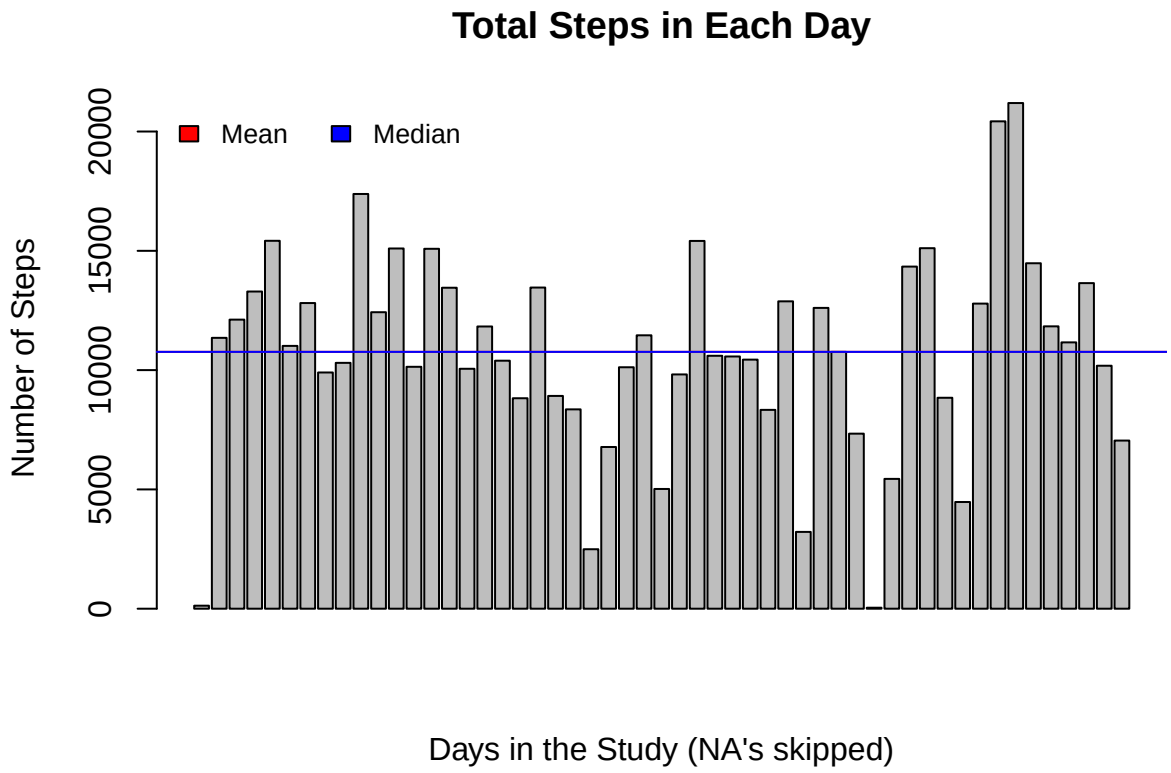
```
## [1] TRUE
```

```
length(unique(steppingDays[[2]]))==length(unique(data[[2]]))
```

```
## [1] FALSE
```

```
barplot(stepsByDay[[2]],
     main="Total Steps in Each Day",
     xlab="Days in the Study (NA's skipped)",
     ylab="Number of Steps",
)
legend("topleft",
       horiz=TRUE, bty='n',
       cex=0.8,
       legend=c("Mean","Median"),
```

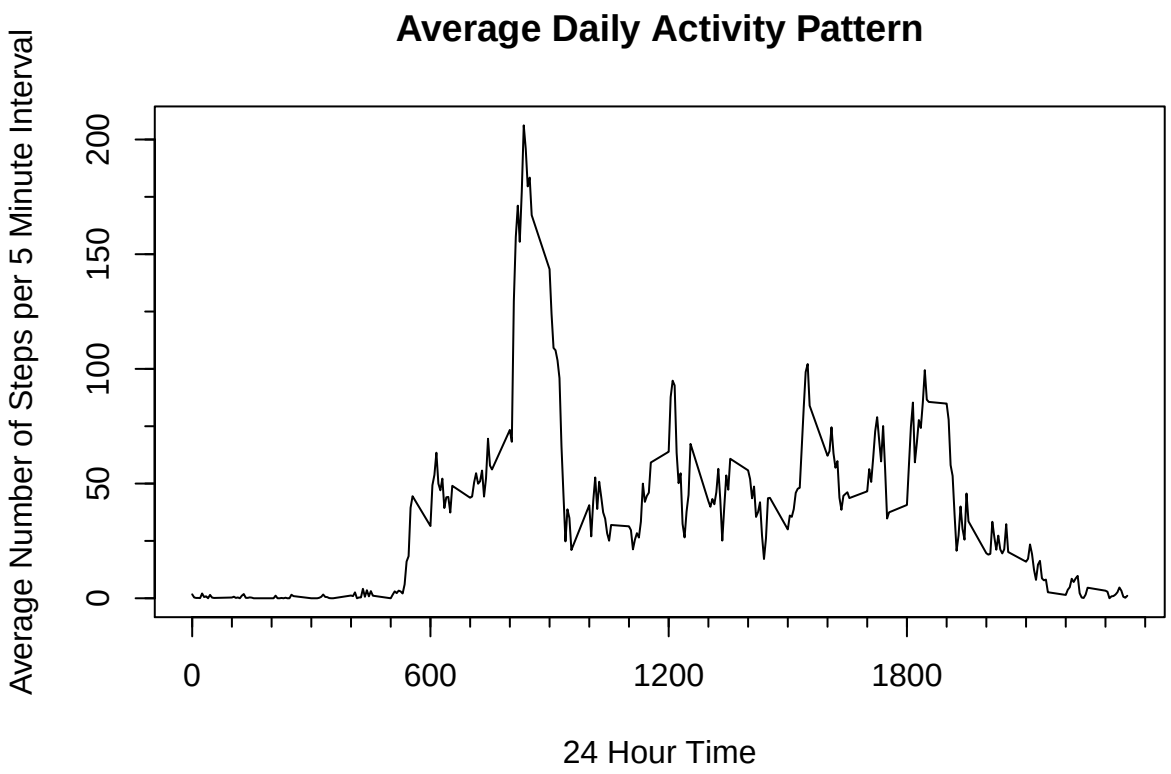
```
       fill=c("red","blue"))
abline(
        h=c(mean(stepsByDay[[2]]),median(stepsByDay[[2]])),
        col=c("red","blue")
)
```



What is the average daily activity pattern?

We need to get a new data set that has the average time for each time interval, and plot the results. You will need to install the Hmisc package to run the following chunk of code with `install.packages("Hmisc")` if you haven't already. Then it must be loaded each time with `library(Hmisc)` (hidden here).

```
dayPatterns=aggregate(steppingDays[1],by=list(interval=steppingDays$interval),mean)
plot(dayPatterns,
     type="l",
     xaxt="n",
     xlab="24 Hour Time",
     ylab="Average Number of Steps per 5 Minute Interval",
     main="Average Daily Activity Pattern"
     )
axis(side=1,
     at=seq(0,2300,by=600)
     )
minor.tick(nx=5, tick.ratio=.5)
```



The time interval with the maximum number of average steps is 835, calculated with `dayPatterns[dayPatterns$steps==max(d`

Inputing missing values

There are 2304 missing data points. I decided to fill in the missing values using the average of the days of the week that are present to avoid bias such as the individual running on certain days of the week or walking to work. The number of each type of day that is missing is below:

```
naData=data[is.na(data[[1]]),]
summary(naData$dayOfWk)/(24*60/5)
```

```
##    Friday    Monday  Saturday    Sunday  Thursday   Tuesday Wednesday
##         2         2         1         1         1         0         1
```

```
summary(naData$dayType)/(24*60/5)
```

```
## weekday weekend
##       6       2
```

```
dataByDays=aggregate(steppingDays$steps,by=list(dayOfWk=steppingDays$dayOfWk,interval=steppingDays$inte
newData=data
for (i in 1:length(newData[[1]])){
        if (is.na(newData$steps[i])){
                newData$steps[i]=
                        dataByDays[dataByDays$dayOfWk==newData$dayOfWk[i]
                                & dataByDays$interval==newData$interval[i],3]
        }
```

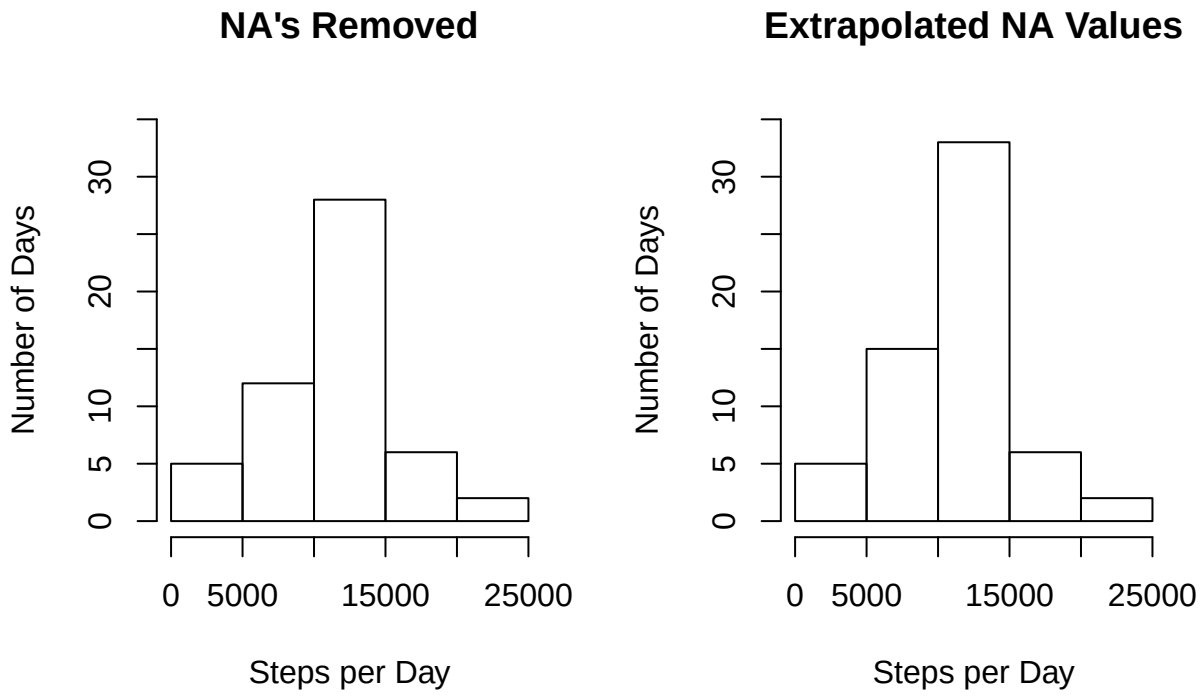
```
}
head(newData)
```

```
##      steps       date interval dayOfWk dayType
## 1 1.428571 2012-10-01        0  Monday weekday
## 2 0.000000 2012-10-01        5  Monday weekday
## 3 0.000000 2012-10-01       10  Monday weekday
## 4 0.000000 2012-10-01       15  Monday weekday
## 5 0.000000 2012-10-01       20  Monday weekday
## 6 5.000000 2012-10-01       25  Monday weekday
```

Now we aggregate our steps by day using our dataset with NA's filled in to create a new histogram of our data. We see from the table below that the mean and median are slightly higher when we extrapolate data for the missing values.

```
stepsByDayNEW=aggregate(newData$steps,by=list(Date=newData$date),sum)
```

```
par(mfrow=c(1,2),oma=c(0,0,2,0))
hist(stepsByDay[[2]],
     main="NA's Removed",
     xlab="Steps per Day",
     ylab="Number of Days",
     ylim=c(0,35)
)
hist(stepsByDayNEW[[2]],
     main="Extrapolated NA Values",
     xlab="Steps per Day",
     ylab="Number of Days",
     ylim=c(0,35)
)
title(main="Total Steps in a Day", outer=T)
```



Statistic	Value	Code
Mean (NA's removed)	10766.19	`mean(stepsByDay[[2]])`
Median (NA's removed)	10765	`median(stepsByDay[[2]])`
Mean (NA's Extrapolated)	10821.21	`mean(stepsByDayNEW[[2]])`
Median (NA's Extrapolated)	11015	`median(stepsByDayNEW[[2]])`

The mean and median are both higher for the dataset with the missing values filled in. The reason for this is because many of the days with missing values are days that have a higher average step count (Wendesday, Friday, and weekends).

```
dataByDays=aggregate(steppingDays$steps,
                     by=list(dayOfWk=steppingDays$dayOfWk,
                             interval=steppingDays$interval),
                     mean)
names(dataByDays)[3]="steps"
dataByDaysAGGR=aggregate(dataByDays$steps,
                     by=list(dayOfWk=dataByDays$dayOfWk),
                     sum)
names(dataByDaysAGGR)[2]="avgSteps"
dataByDaysAGGR[c(2,6,7,5,1,3,4),]
```

```
##     dayOfWk  avgSteps
## 2    Monday  9974.857
## 6   Tuesday  8949.556
```

```
## 7 Wednesday 11790.750
## 5  Thursday  8212.750
## 1    Friday 12359.714
## 3  Saturday 12535.429
## 4    Sunday 12277.714
```

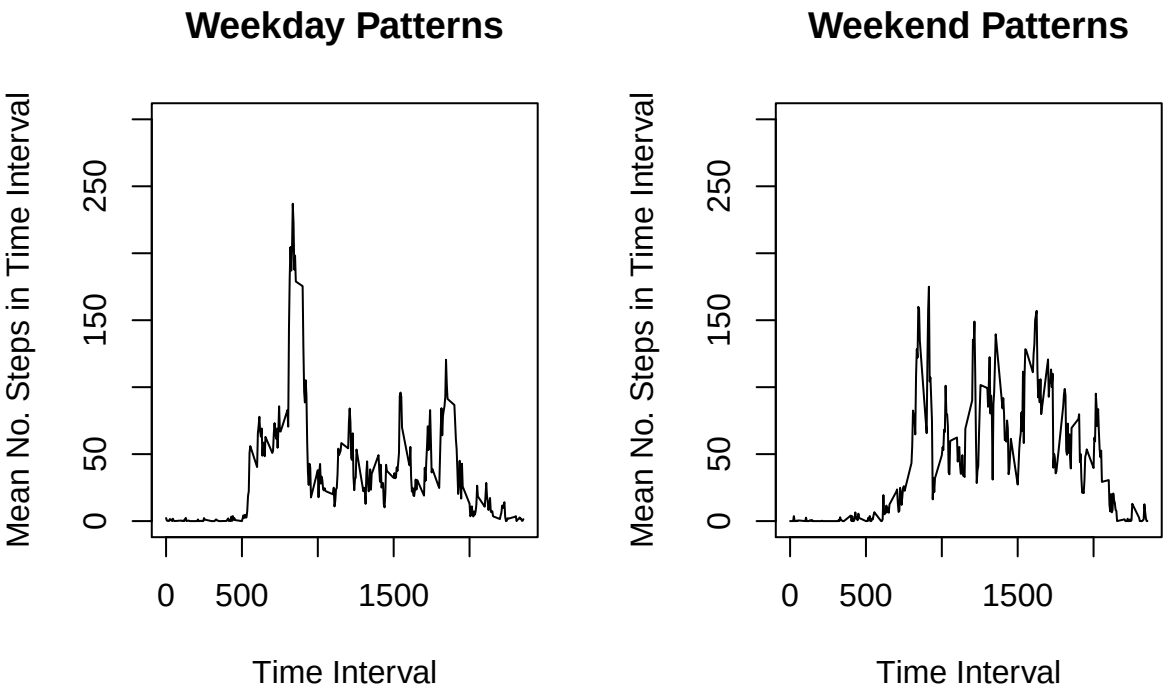
Are there differences in activity patterns between weekdays and weekends?

We can see some pattern difference between the weekdays and the weekends. On the weekdays, there appear to be certain times of the day with more activity. Weekend activity seems to be more spread out.

```
dayTypeAggr=aggregate(newData$steps,
                      by=list(dayType=newData$dayType,interval=newData$interval),
                      mean
)

par(mfrow=c(1,2),oma=c(0,0,2,0))
plot(dayTypeAggr[dayTypeAggr$dayType=="weekday",2:3],
     type="l",
     ylim=c(0,300),
     ylab="Mean No. Steps in Time Interval",
     xlab="Time Interval",
     main="Weekday Patterns"
     )
plot(dayTypeAggr[dayTypeAggr$dayType=="weekend",2:3],
     type="l",
     ylim=c(0,300),
     ylab="Mean No. Steps in Time Interval",
     xlab="Time Interval",
     main="Weekend Patterns"
     )
title(main="Weekends vs. Weekdays",outer=T)
```

Weekends vs. Weekdays



To compare weekend and weekday activity, we look at the average number of total steps for each of the day types, and see that the average number of steps on a weekend day is significantly higher than average, but is on par with Wednesday and Friday activity.

```
dayTypeComp=aggregate(newData$steps,
                      by=list(date=newData$date,
                              dayType=newData$dayType
                              ),
                      sum)
aggrDayType=aggregate(dayTypeComp$x,
                      by=list(dayType=dayTypeComp$dayType),
                      mean
                      )
names(aggrDayType)[2]="avgSteps"
head(aggrDayType)
```

```
##   dayType avgSteps
## 1 weekday 10257.53
## 2 weekend 12406.57
```

```
dataByDaysAGGR[c(2,6,7,5,1,3,4),]
```

```
##     dayOfWk  avgSteps
## 2    Monday  9974.857
## 6   Tuesday  8949.556
## 7 Wednesday 11790.750
## 5  Thursday  8212.750
```

```
## 1    Friday 12359.714
## 3  Saturday 12535.429
## 4    Sunday 12277.714
```