Activity Monitoring Analysis

Melanie Butler

Peer Assessment 1 for Reproducible Research

Get the Data

In this section, we download the data.

This code chunk downloads the .CSV file with the data, unzips it, and appropriately changes the working directory again.

```
##Dowload data, unzip, and set working directory
fileURL<-"https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2Factivity.zip"
download.file(fileURL, "ActivityData.zip")
unzip("ActivityData.zip", exdir = "ActivityData")
```

This next chunck reads in the data and prints a summary.

```
AD<-read.csv("activity.csv")
summary(AD)
```

```
##      steps                date          interval
##  Min.   :  0.00   2012-10-01:  288   Min.   :   0.0
##  1st Qu.:  0.00   2012-10-02:  288   1st Qu.: 588.8
##  Median :  0.00   2012-10-03:  288   Median :1177.5
##  Mean   : 37.38   2012-10-04:  288   Mean   :1177.5
##  3rd Qu.: 12.00   2012-10-05:  288   3rd Qu.:1766.2
##  Max.   :806.00   2012-10-06:  288   Max.   :2355.0
##  NA's   :2304     (Other)   :15840
```

Process the Data

In this section we process the data so that we can begin the analysis. We also load libraries that we need.

In this first chunk in this section, we coerce the date column from a factor variable to a date variable.

```
AD$date<-as.Date(AD$date)
```

We load ggplot2, dplyr, plyr, and gridExtra.

```
library(ggplot2)
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
```

```
library(plyr)
```

```
## -------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr
:
## library(plyr); library(dplyr)
```

```
## -------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 3.4.4
```

```
##
## Attaching package: 'gridExtra'
```

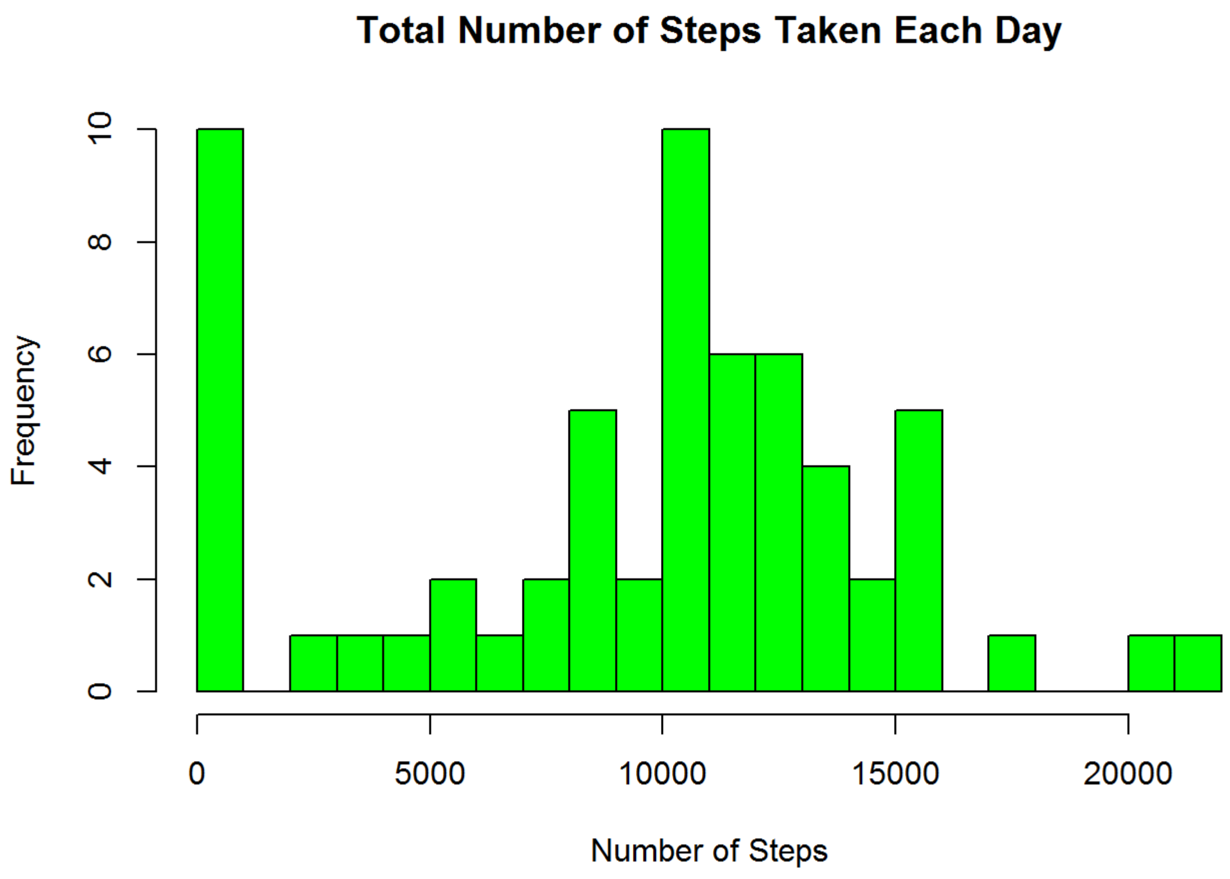
```
## The following object is masked from 'package:dplyr':
##
##     combine
```

Total Number of Steps Each Day

In this section we want to get a general feel for the data. We ignore missing values, which are reported in the data as NA.

In this first code chunk, we make a histogram of the total number of steps taken each day.

```
dailytotal<-aggregate(AD$steps, FUN=sum, by=list(Group.date=AD$date), na.rm=TRUE)
hist(dailytotal$x, col="green", breaks = 25, xlab = "Number of Steps",
     ylab ="Frequency", main = "Total Number of Steps Taken Each Day" )
```



In this second code chunk we calulate the mean and median number of steps taken each day.

```
M<-mean(dailytotal$x)
Med<-median(dailytotal$x)
```

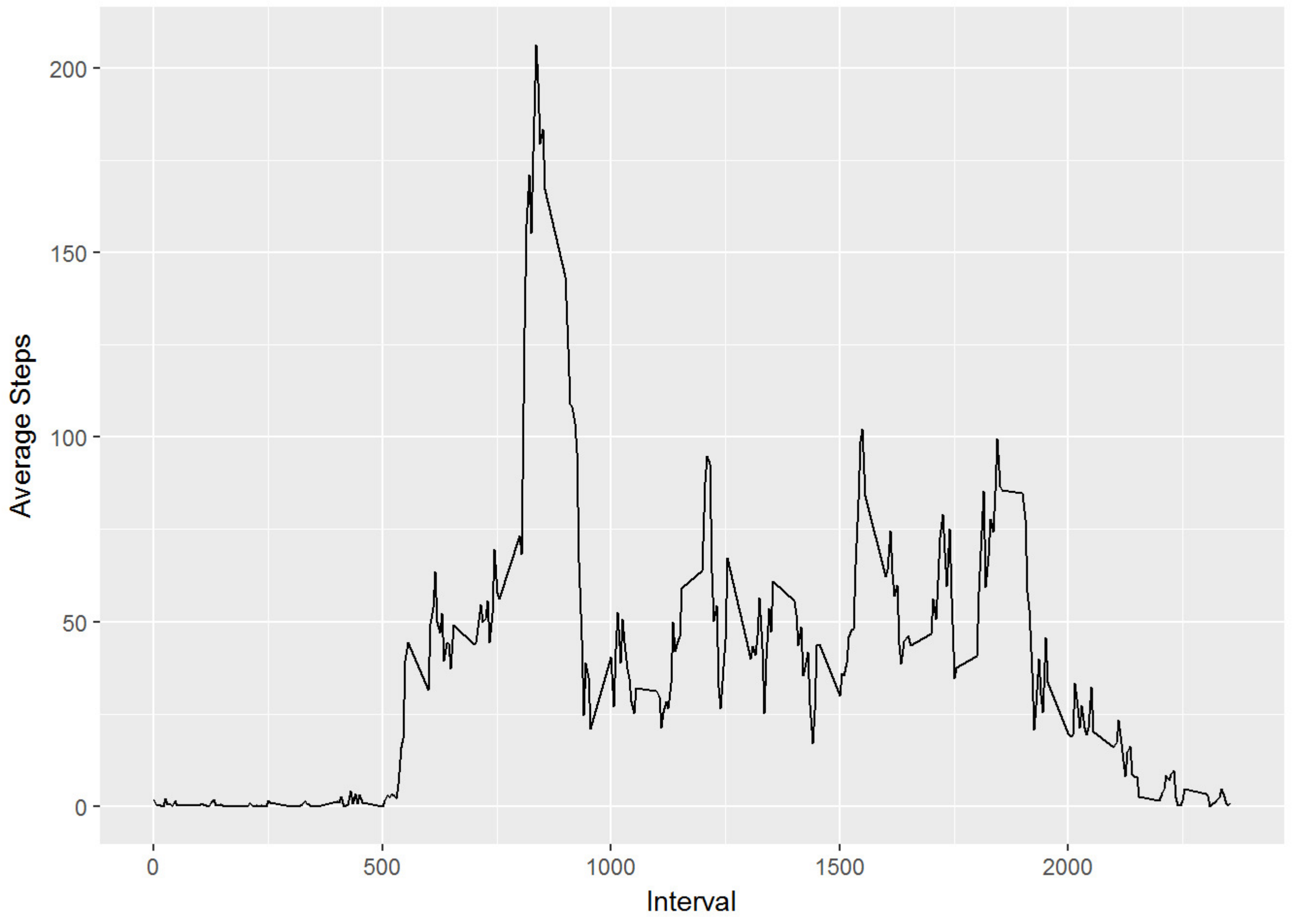
The mean is 9354.2295082. The median is 10395.

5-minute intervals

In this section we explore the data grouped by the five-minute intervals, still ignoring the missing (NA) values.

In this first code chunk, we make a time series plot of the average number of steps taken in each five-minute interval.

```
intervals<-aggregate(AD$steps, FUN=mean,
                     by=list(Group.Interval=AD$interval), na.rm=TRUE)
ggplot(intervals, aes(Group.Interval, x))+
    geom_line() +
    xlab("Interval") + ylab("Average Steps")
```



In this next code chunk, we find the five-minute interval that on average contains the maximum number of steps.

```
maximum<-max(intervals$x)
maxstepinterval<-filter(intervals, x>206)
maxstepinterval<-maxstepinterval[1,1]
```

The time interval that on average contains the maximum number of steps is 835.

Imputing Missing Data

In this section, we impute the missing data. Without knowing more about how the data was collected, we do not know why the data is missing. For this reason, it makes the most sense to use the mean number of steps for a given day to replace any of the NA values for that day.

In this first code chunk, we calculate the total number of missing (NA) data values.

```
missing<-sum(is.na(AD$steps))
```

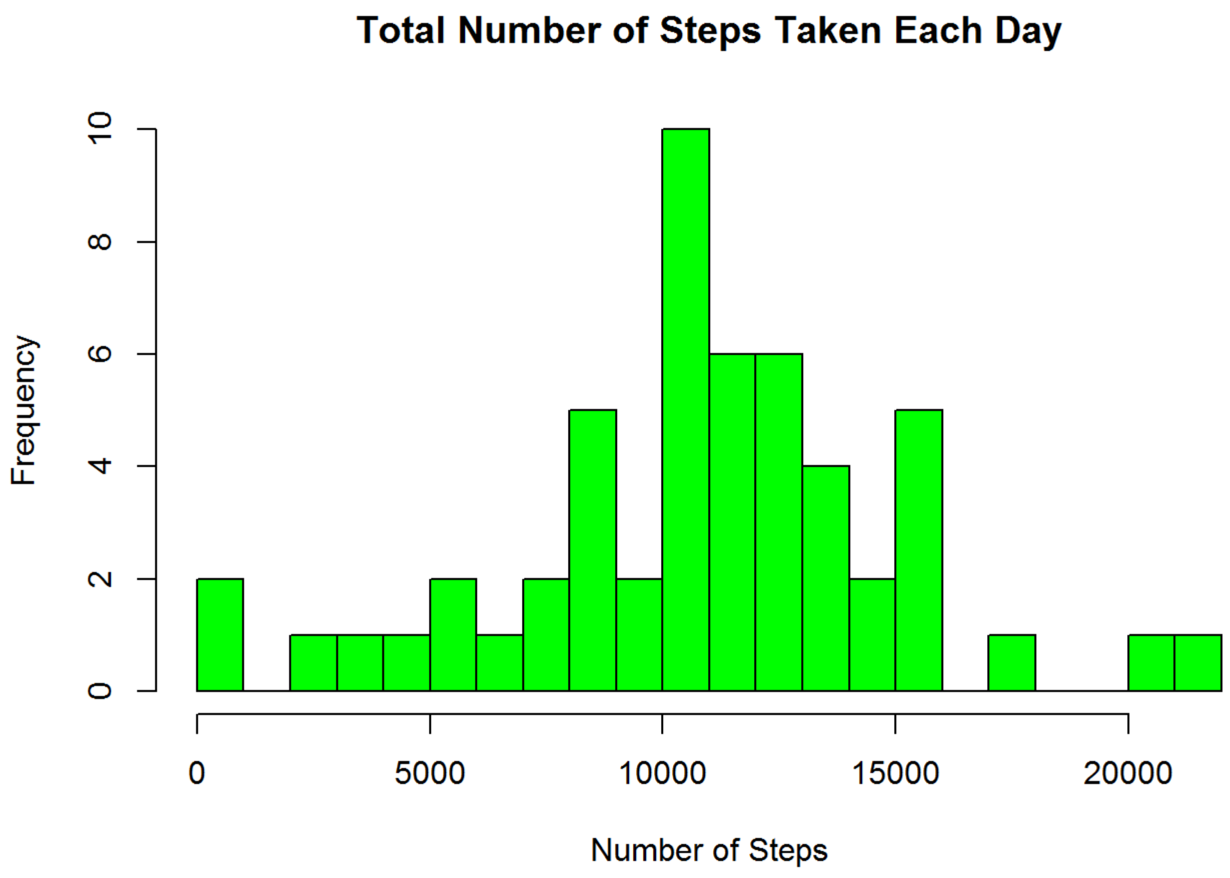
The total number of rows with missing data values is 2304.

In this next code chunk we impute the missing data values by replacing missing data values with the mean number of steps for that day. In the case of missing data over every interval in a day, we remove that day from the analysis.

```
AD2<-read.csv("activity.csv")
AD2$date<-as.character(AD2$date)
AD2$date<-factor(AD2$date, labels=c(1:61))
imputedmean<-NULL
for (i in 1:61){
    imputedmean<-c(imputedmean,
        ifelse(is.na(AD2[which(AD2$date==i),]$steps),
               mean(AD2[which(AD2$date==i),]$steps, na.rm= TRUE),
               AD2[which(AD2$date==i),]$steps))
}
AD3<-cbind(AD2[,2:3], imputedmean)
AD3<-na.omit(AD3)
```

In this code chunk, we make a histogram of the total number of steps taken each day after we have imputed the missing data values.

```
dailytotal2<-aggregate(AD3$imputedmean, FUN=sum, by=list(Group.date=AD3$date), na.rm=
TRUE)
hist(dailytotal2$x, col="green", breaks = 25, xlab = "Number of Steps",
     ylab ="Frequency", main = "Total Number of Steps Taken Each Day" )
```



Note that there does not appear to be a difference in the two histograms (total number of steps taken each day before and after imputing), except for the number of days with 0 steps.

Weekdays vs Weekends

In this section we compare the number of steps taken per 5-minute interval in weekdays vs weekends.

In this code chunk, we separate the data into two dataframes - one for weekdays and one for Weekends.

```
AD4<-cbind(AD, weekdays(AD$date))
names(AD4)<-c("steps", "date", "interval", "dayofweek")
Monday<-AD4[which(AD4$dayofweek=="Monday"),]
Tuesday<-AD4[which(AD4$dayofweek=="Tuesday"),]
Wednesday<-AD4[which(AD4$dayofweek=="Wednesday"),]
Thursday<-AD4[which(AD4$dayofweek=="Thursday"),]
Friday<-AD4[which(AD4$dayofweek=="Friday"),]
Saturday<-AD4[which(AD4$dayofweek=="Saturday"),]
Sunday<-AD4[which(AD4$dayofweek=="Sunday"),]
MTWHF<-join(join(join(join(Monday, Tuesday), Wednesday), Thursday), Friday)
```

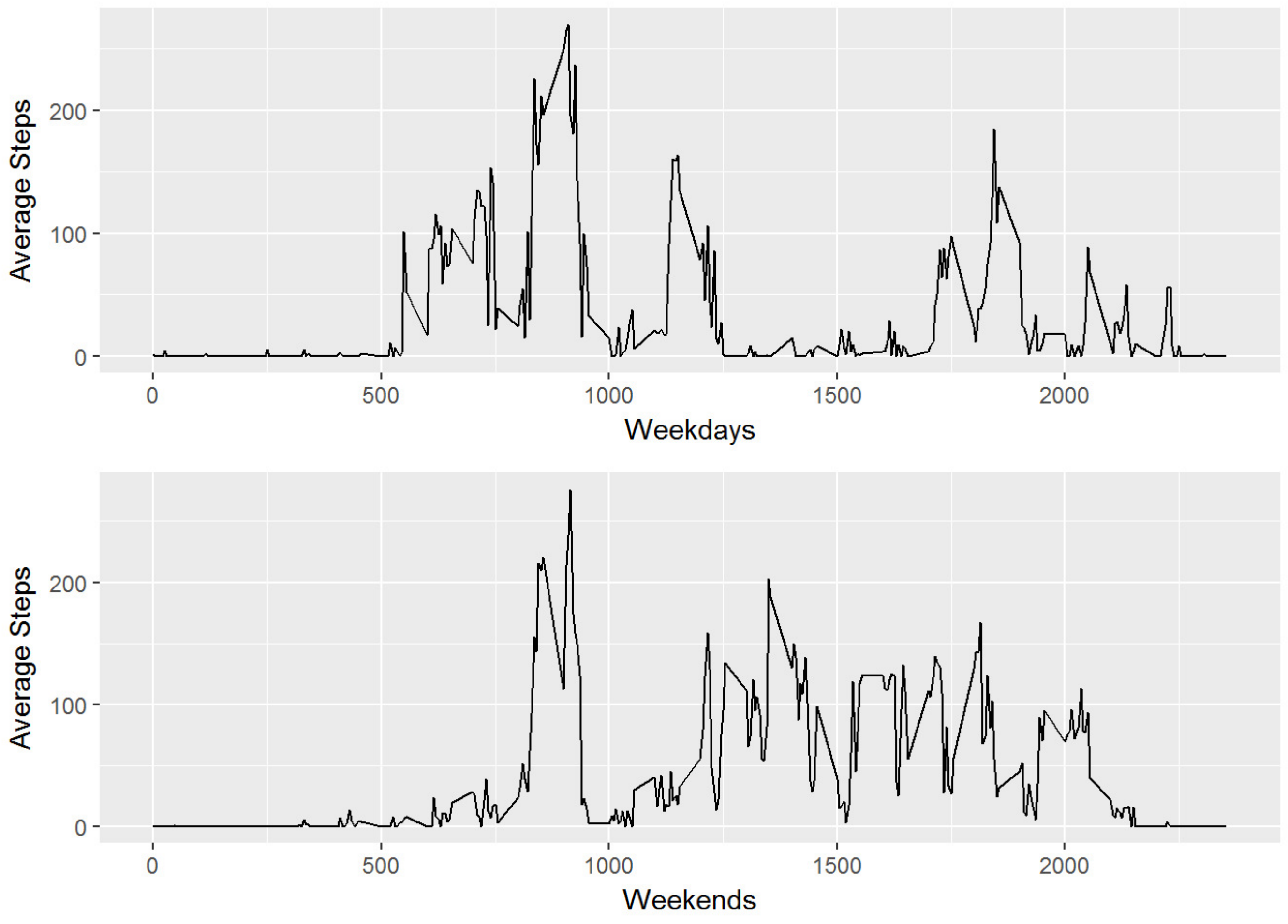
```
## Joining by: steps, date, interval, dayofweek
## Joining by: steps, date, interval, dayofweek
## Joining by: steps, date, interval, dayofweek
## Joining by: steps, date, interval, dayofweek
```

```
SatSun<-join(Saturday, Sunday)
```

```
## Joining by: steps, date, interval, dayofweek
```

In this code chunk, we make a two time series plots of the average number of steps taken in each five-minute interval on weekdays versus weekends.

```
intervalsweekday<-aggregate(MTWHF$steps, FUN=mean,
                     by=list(Group.Interval=MTWHF$interval), na.rm=TRUE)
intervalsweekend<-aggregate(SatSun$steps, FUN=mean,
                     by=list(Group.Interval=SatSun$interval), na.rm=TRUE)
p1<-ggplot(intervalsweekday, aes(Group.Interval, x))+
    geom_line() +
    xlab("Weekdays") + ylab("Average Steps")
p2<-ggplot(intervalsweekend, aes(Group.Interval, x))+
    geom_line() + xlab("Weekends") + ylab("Average Steps")
grid.arrange(p1, p2, nrow=2)
```



From these plots, we see a later start to steps on the weekends (probably the person is sleeping in on the weekends) and more steps in the middle of the day on weekends (probably because the person is not working at a desk).