PA1-template.Rmd

David N. Cohron

January 13, 2016

Activity Monitoring Analysis

Reproducible Research

Johns Hopkins University- Coursera

Project #1

Overview:

This project is designed to produce a report from raw data. Data source is a personal activity monitoring device worn by an anonymous individual for 61 days during October and November 2012 and included the number of steps taken in 5 minute intervals each day.

The Data:

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.

Course Assignment:

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

```
# some general housekeeping to get started
# globally suppress warnings
oldwarn<- getOption("warn")
options(warn = -1)

# install libraries/packages
library(plyr)
library(dplyr)
library(lubridate)
```

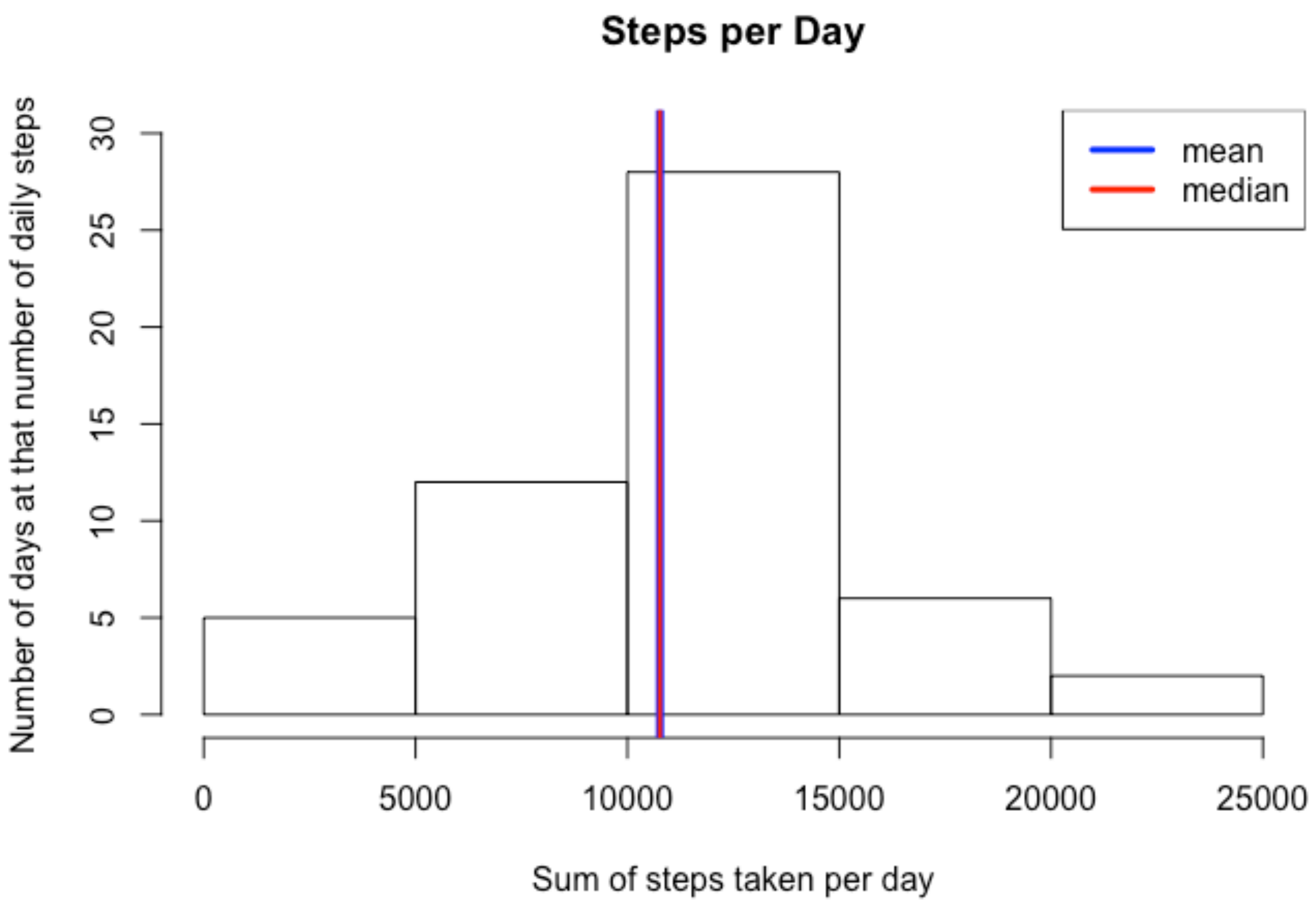
```
# read file into data frame for manipulation
file<- "activity.csv"
main<- read.csv(file)

# refactor raw date to type date
main<- mutate(main, date=as.Date(as.character(main$date)))
```

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

```
# group by days to get sum of steps per day
# ignore days with no steps recorded
stepsbyday<- group_by(main, date) %>% summarize_each(funs(sum))
na.omit(stepsbyday)
```

```
## Source: local data frame [53 x 3]
##
##          date steps interval
##        (date) (int)    (int)
## 1  2012-10-02   126   339120
## 2  2012-10-03 11352   339120
## 3  2012-10-04 12116   339120
## 4  2012-10-05 13294   339120
## 5  2012-10-06 15420   339120
## 6  2012-10-07 11015   339120
## 7  2012-10-09 12811   339120
## 8  2012-10-10  9900   339120
## 9  2012-10-11 10304   339120
## 10 2012-10-12 17382   339120
## ..        ...   ...      ...
```

```
#calculate mean and median of number of steps per day
meansteps<-mean(stepsbyday$steps, na.rm=TRUE)
mediansteps<- median(stepsbyday$steps, na.rm=TRUE)

# plot histogram
hist(stepsbyday$steps,
     main = "Steps per Day",
     xlab = "Sum of steps taken per day",
     ylab = "Number of days at that number of daily steps",
     xlim = c(0, 25000),
     ylim = c(0, 30))

abline(v = meansteps, col = "blue", lwd=4)
abline(v = mediansteps, col = "red", lwd=2)

legend('topright',
       c("mean", "median"),
       col=c("blue", "red"),
       lwd=3)
```



3. Mean and median number of steps taken each day Note that the mean and the median are quite close.

```
# calculate mean and median of number of steps per day
# note: this is already reflected on the histogram
meansteps<-mean(stepsbyday$steps, na.rm=TRUE)
meansteps
```

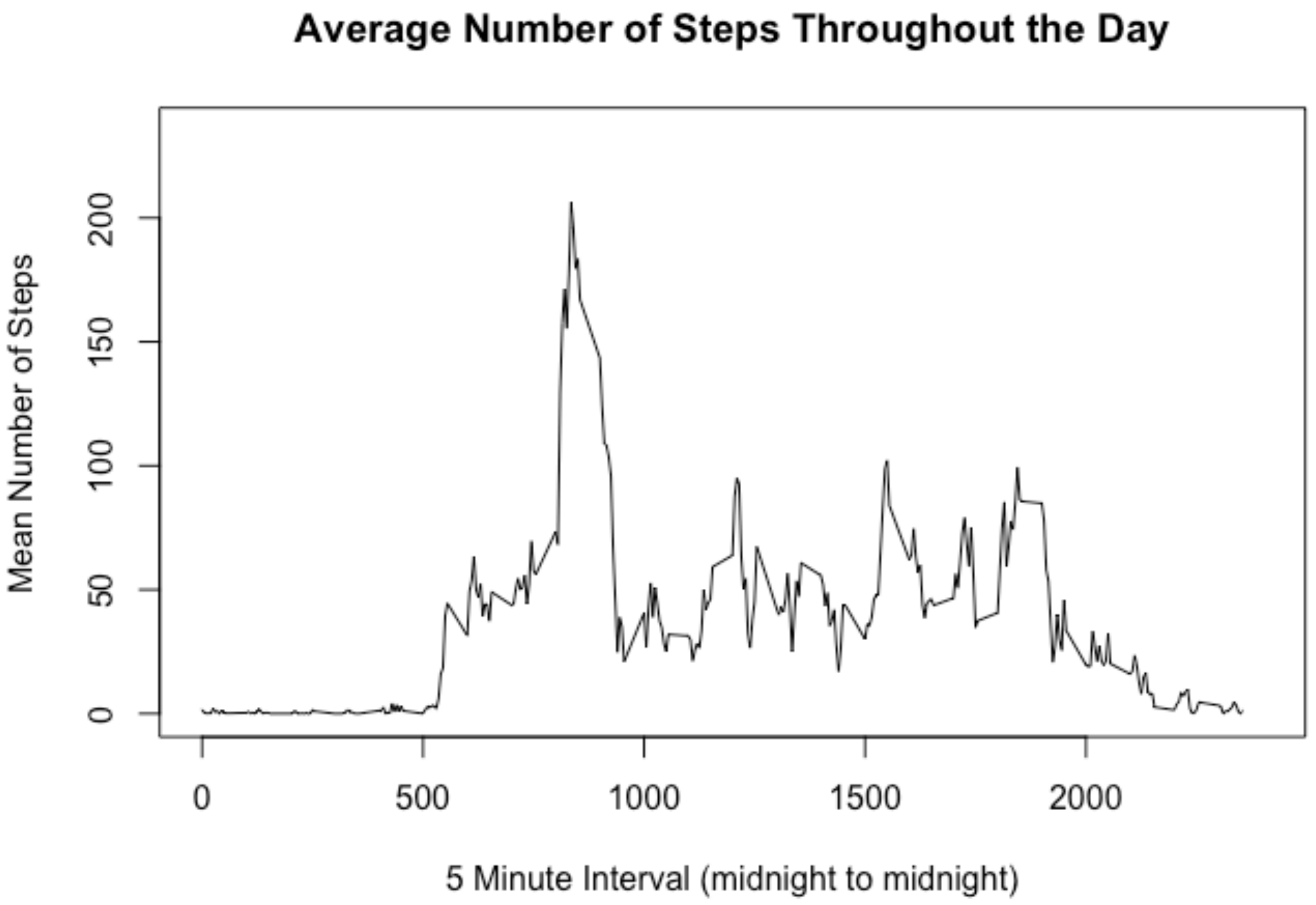
```
## [1] 10766.19
```

```
mediansteps<- median(stepsbyday$steps, na.rm=TRUE)
mediansteps
```

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

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

```
# group by 5 minute interval
# ignore intervals with no steps recorded
stepsbyinterval<- group_by(main, interval) %>% summarize_each(funs(mean(., na.rm=TRUE
)))

# plot time series of 5 minute interval average steps
with(stepsbyinterval, plot(interval, steps,
                           type = "l",
                           main = "Average Number of Steps Throughout the Day",
                           xlab = "5 Minute Interval (midnight to midnight)",
                           ylab = "Mean Number of Steps",
                           xlim = c(0, 2400),
                           ylim = c(0, 235)))
```



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

```
# calculate which 5 minute interval has highest average number of steps
maxintervalrow<- which.max(stepsbyinterval$steps)
maxinterval<- stepsbyinterval$interval[maxintervalrow]
maxinterval
```

```
## [1] 835
```

```
maxavgsteps<- stepsbyinterval$steps[maxintervalrow]
maxavgsteps
```

```
## [1] 206.1698
```

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

```
# calculate  and report the number of rows with missing (NA) data
numrowtotal<- nrow(main)
numrownona<- nrow(na.omit(main))
numrowwithna<- numrowtotal - numrownona
numrowwithna
```

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

```
# impute missing values and replace with interval mean
impute.mean <- function(x) replace(x, is.na(x), mean(x, na.rm = TRUE))
mainimputed <- ddply(main, ~ interval, transform, steps = impute.mean(steps))
```

7. Histogram of the total number of steps taken each day after missing values are imputed Note that the new mean and median are identical once we substituted the median for the interval for all of the missing values.

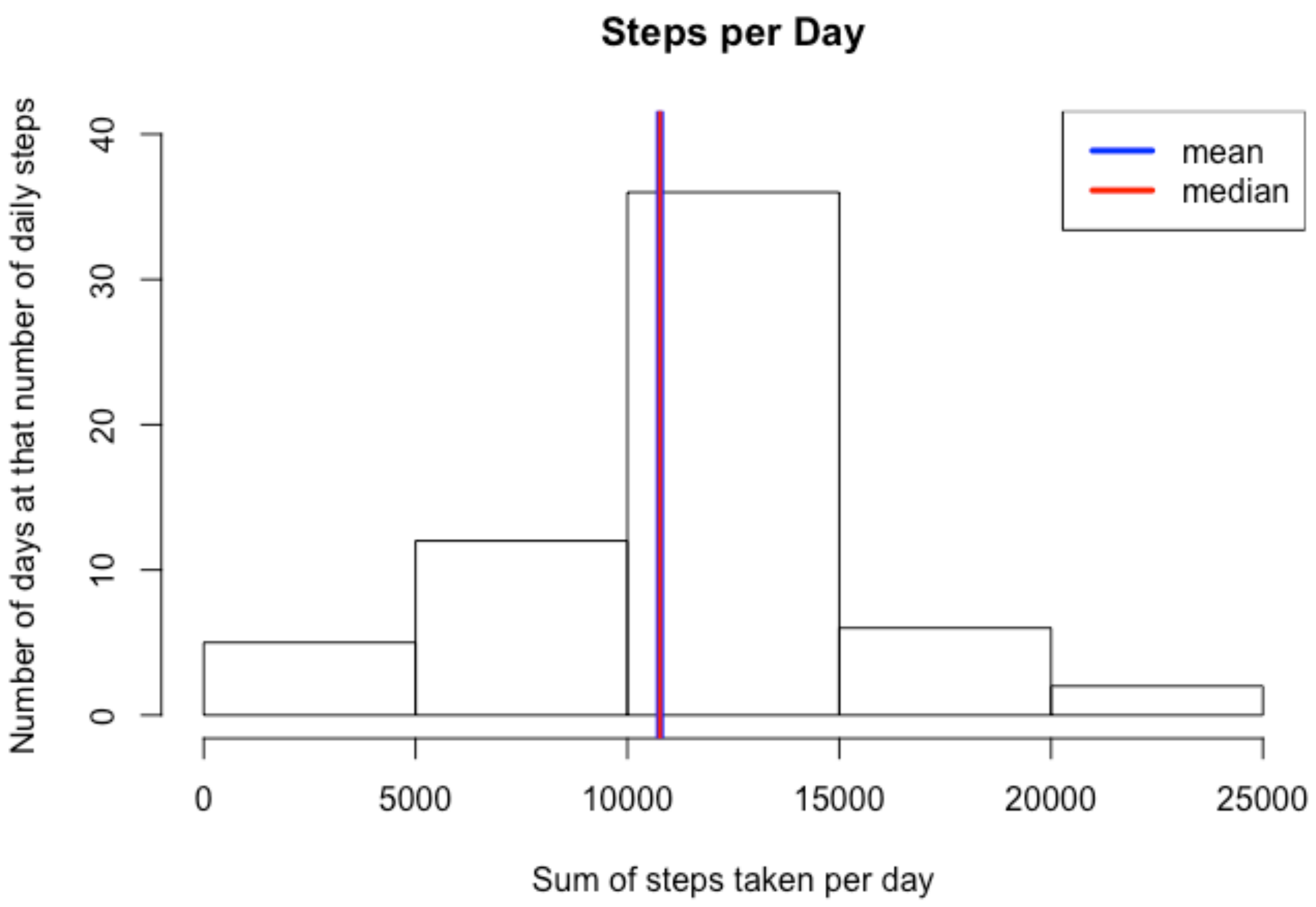
```
# group new data frame by days to get sum of steps per day
# all days/intervals have steps recorded
stepsbyday2<- group_by(mainimputed, date) %>% summarize_each(funs(sum))

# calculate mean and median of number of steps per day
meansteps2<-mean(stepsbyday2$steps)
meansteps2
```

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

```
mediansteps2<- median(stepsbyday2$steps)
mediansteps2
```

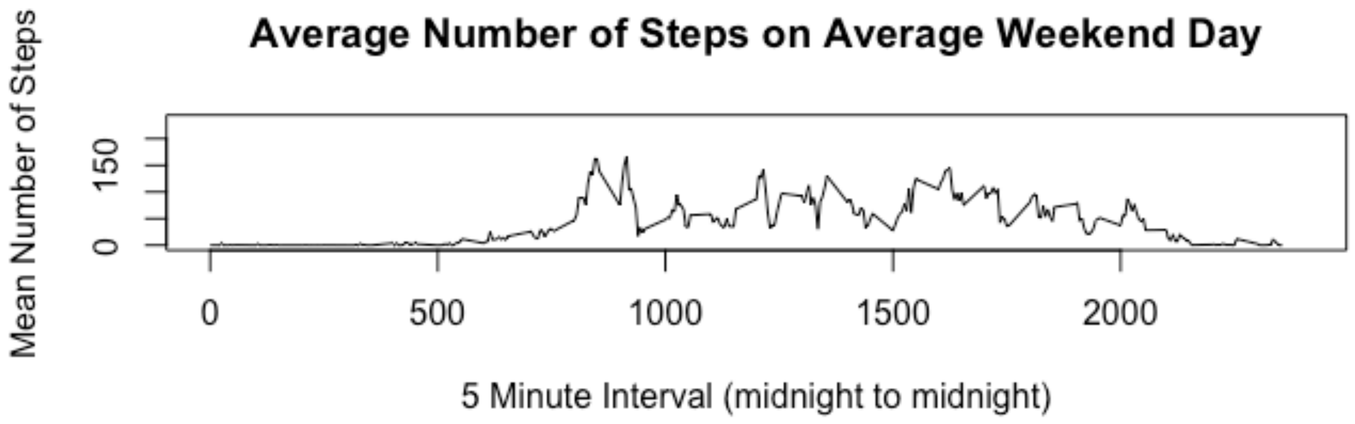
```
## [1] 10766.19
```

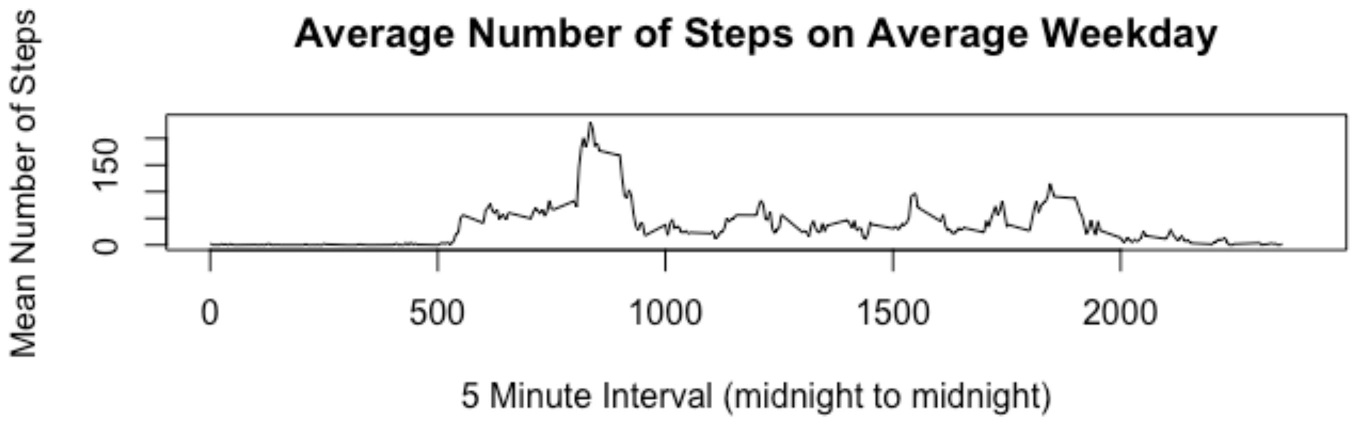
```
# plot histogram on imputed data
hist(stepsbyday2$steps,
     main = "Steps per Day",
     xlab = "Sum of steps taken per day",
     ylab = "Number of days at that number of daily steps",
     xlim = c(0, 25000),
     ylim = c(0, 40))

abline(v = meansteps2, col = "blue", lwd=4)
abline(v = mediansteps2, col = "red", lwd=2)

legend('topright',
       c("mean", "median"),
       col=c("blue", "red"),
       lwd=3)
```



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

```
# add columns showing day of the week
stepsbyday3<- mutate(mainimputed, dayofweek = weekdays(date, abbreviate=TRUE))

# add column with logical vector as to weekday/weekend
stepsbyday3<- mutate(stepsbyday3, weekend = (dayofweek == "Sun" | dayofweek == "Sat")
)

# subset for weekday and weekend in new data frames
weekdaysteps<- subset(stepsbyday3, !weekend)
weekendsteps<- subset(stepsbyday3, weekend)

# convert logical vector to factors with labels
# Note: I did not need to do this for my analysis,
#       but it was a requirement of the project
stepsbyday3$weekend<-factor(stepsbyday3$weekend,levels=c(FALSE, TRUE), labels=c('Week
day', 'Weekend'))

#group by date
weekdaystepsum<- group_by(weekdaysteps, date) %>% summarize_each(funs(sum(steps)))
weekendstepsum<- group_by(weekendsteps, date) %>% summarize_each(funs(sum(steps)))

# calculate mean and median of number of steps per day
meanstepsweekday<-mean(weekdaystepsum$steps)
medianstepsweekend<- median(weekendstepsum$steps)

# group by 5 minute interval
# ignore intervals with no steps recorded
weekdaystepsbyinterval<- group_by(weekdaysteps, interval) %>% summarize_each(funs(mea
n(., na.rm=TRUE)))
weekendstepsbyinterval<- group_by(weekendsteps, interval) %>% summarize_each(funs(mea
n(., na.rm=TRUE)))

# plot time series of 5 minute interval average steps for weekday and weekends
# Note:  I did not try to duplicate the example exactly. I believe this is more clear
.
layout(matrix(c(2,1), byrow = TRUE),
       widths=c(1,1), heights=c(3,3))
with(weekdaystepsbyinterval, plot(interval, steps,
                                  type = "l",
                                  main = "Average Number of Steps on Average Weekday"
,
                                  xlab = "5 Minute Interval (midnight to midnight)",
                                  ylab = "Mean Number of Steps",
                                  xlim = c(0, 2400),
                                  ylim = c(0, 235)))

with(weekendstepsbyinterval, plot(interval, steps,
                                  type = "l",
                                  main = "Average Number of Steps on Average Weekend
Day",
                                  xlab = "5 Minute Interval (midnight to midnight)",
                                  ylab = "Mean Number of Steps",
                                  xlim = c(0, 2400),
                                  ylim = c(0, 235)))
```





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

```
# restore warnings
options(warn = oldwarn)
```