Activity Monitoring Analysis

Karl P. Lacher

August 14, 2016

Activity Monitoring Analysis

Introduction

(Copied from the Assignment 2 instructions for the Coursera course: "Reproducible Research" from Johns Hopkins University)

It is now possible to collect a large amount of data about personal movement using activity monitoring devices such as a Fitbit, Nike Fuelband, or Jawbone Up. These type of devices are part of the "quantified self" movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. But these data remain under-utilized both because the raw data are hard to obtain and there is a lack of statistical methods and software for processing and interpreting the data.

This assignment makes use of data from a personal activity monitoring device. This device collects data at 5 minute intervals throughout the day. The data consists of two months of data from an anonymous individual collected during the months of October and November, 2012 and includes the number of steps taken in 5 minute intervals each day.

About the Data

The data for this assignment was obtained from a fork of the the GitHub repository - https://github.com/rdpeng/RepData_PeerAssessment1

Dataset: activity.zip There are 17,568 observations in the dataset which is stored as a comma-separated-value (CSV) file.

Variables in the dataset:

- steps: Number of steps taking in a 5-minute interval (missing values are coded as NA)
- date: Date on which the measurement was taken in YYYY-MM-DD format
  - Dates of measurement occur between October 1, 2012 and November 30, 2012
- interval: Identifier for the 5-minute interval within a 24 hour period in which the measurement was taken

Load the R libraries required for the analysis and read the data

```
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(ggplot2)

## Read the data
df <- read.csv("activity.csv")
```

Get the date range of the dataset containing the 24 hour periods

```
minDate <- as.character(min(as.Date(df$date)))
maxDate <- as.character(max(as.Date(df$date)))
sprintf("All activity measurements were taken between %s and %s", minDate,
maxDate)

## [1] "All activity measurements were taken between 2012-10-01 and 2012-11-
30"
```

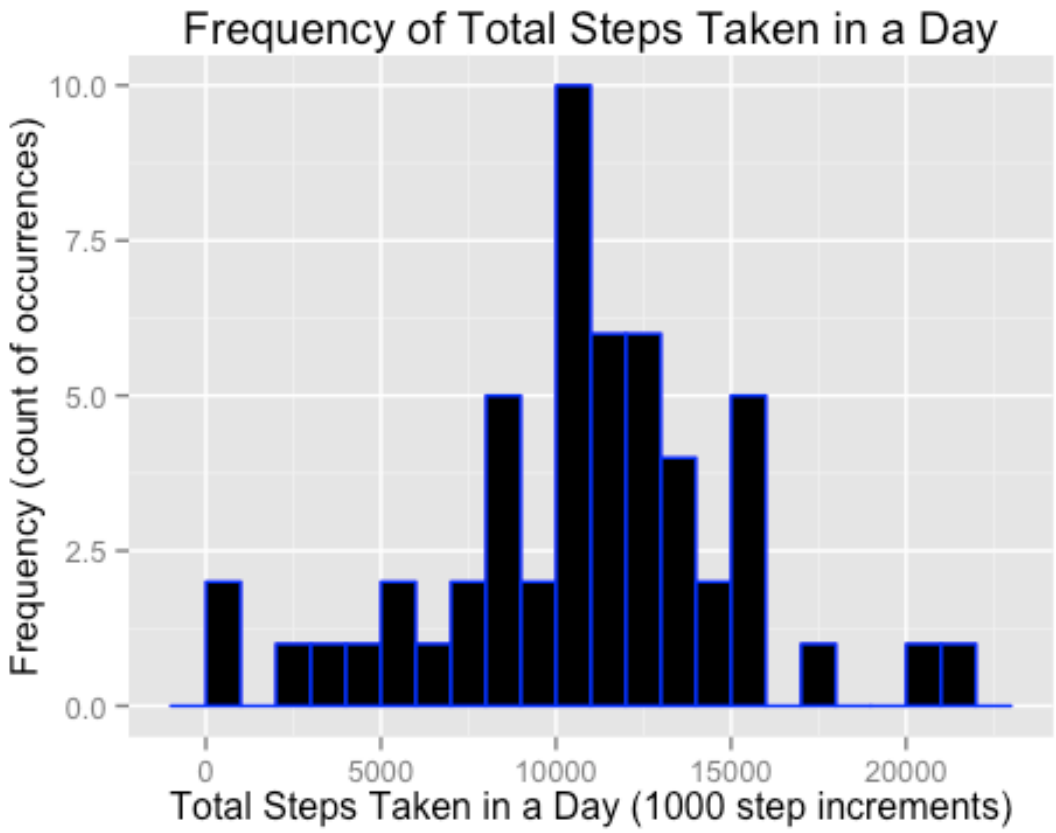
Calculate the mean total number of steps taken per day

Display the information as a histogram

Note that NA values are ignored for this part of the analysis.

```
## Calculate total steps per day (ignores NA values)
dfSum <- aggregate(x=df$steps, by=list(df$date), FUN=sum)
## Rename aggregate data frame columns
dfSum <- rename(dfSum, date=Group.1, steps=x)

## Part 1 - Make a histogram of the total number of steps taken each day
## Histogram of total steps taken each day
## Construct the plot
g <- ggplot(dfSum, aes(steps))

## Histogram of total steps frequency
g <- g + geom_histogram(binwidth=1000, colour = "blue", fill = "black")

## Add main label
g <- g + ggtitle("Frequency of Total Steps Taken in a Day")

## Add x and y axis labels
g <- g + xlab("Total Steps Taken in a Day (1000 step increments)")
g <- g + ylab("Frequency (count of occurrences)")
```

```
## Display histogram
print(g)
```



Calculate mean and median of total number of steps taken per day

Note that NA values are being ignored for this part of the analysis.

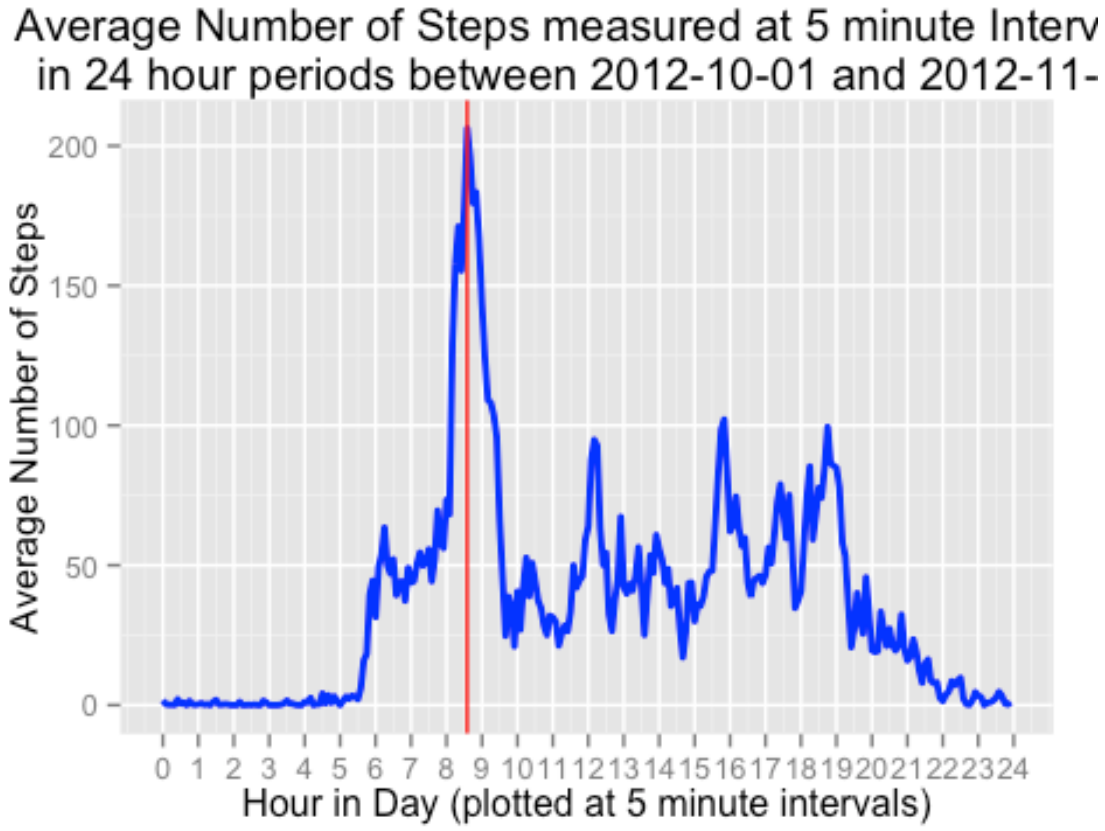
```
meanSteps <- mean(dfSum$steps, na.rm=TRUE)
medianSteps <- median(dfSum$steps, na.rm=TRUE)
sprintf("Mean total steps per day: %.1f", meanSteps)

## [1] "Mean total steps per day: 10766.2"

sprintf("Median total steps per day: %.1f", medianSteps)

## [1] "Median total steps per day: 10765.0"
```

Profile the average daily activity pattern

Note that NA values are being ignored for this part of the analysis.

```
dfADAP <- aggregate(x=df$steps, by=list(as.factor(df$interval)), FUN=mean,
na.rm=TRUE, na.action=omit)
dfADAP <- rename(dfADAP, interval=Group.1, avgSteps=x)
dfADAP <- mutate(dfADAP, hour=paste('0000', as.character(dfADAP$interval),
sep=""))
```

```r
dfADAP$hour <- substr(dfADAP$hour, nchar(dfADAP$hour)-3, nchar(dfADAP$hour))
dfADAP$hour <- as.numeric(substr(dfADAP$hour, 1, 2)) +
as.numeric(substr(dfADAP$hour, 3, 4)) / 60.0

## Construct a line plot of the average daily activity pattern
g <- ggplot(dfADAP, aes(hour, avgSteps))

## Line plot
g <- g + geom_line(aes(group=1), colour="blue", size=1)
g <- g + scale_x_continuous(breaks = seq(0, 24, by=1))

## Add vertical intercept for interval with peak activity
maxSteps <- filter(dfADAP, avgSteps == max(avgSteps) )
g <- g + geom_vline(xintercept = maxSteps$hour, colour="red")

## Add main title
## Add main label
maintitle1 <- "Average Number of Steps measured at 5 minute Intervals\n in 24
hour periods"
maintitle2 <- paste("between",minDate,"and",maxDate)
title <- paste(maintitle1,maintitle2)
g <- g + ggtitle(title)

## Add x and y axis labels
g <- g + xlab("Hour in Day (plotted at 5 minute intervals)")
g <- g + ylab("Average Number of Steps")

## Display plot
print(g)
```




```
## Identify which 5-minute interval, on average across all the days in the
dataset, contains the maximum number of steps
sprintf("5-minute interval with max number of steps (averaged over all days):
%s", maxSteps$interval)

## [1] "5-minute interval with max number of steps (averaged over all days):
835"
```

The vertical red line in the line plot above marks this interval.

Imputing Missing Values

Calculate and report the total number of missing values in the dataset (# of NA values)

```
totalNAs <- sum(is.na(df$steps))
sprintf("Total number of NA values in dataset: %.0f", totalNAs)

## [1] "Total number of NA values in dataset: 2304"
```

Strategy for filling in missing dataset values

The strategy for filling missing values in the dataset is to use the mean value of the number of steps for the 5-minute intervals. A new dataset is calculated from the original dataset

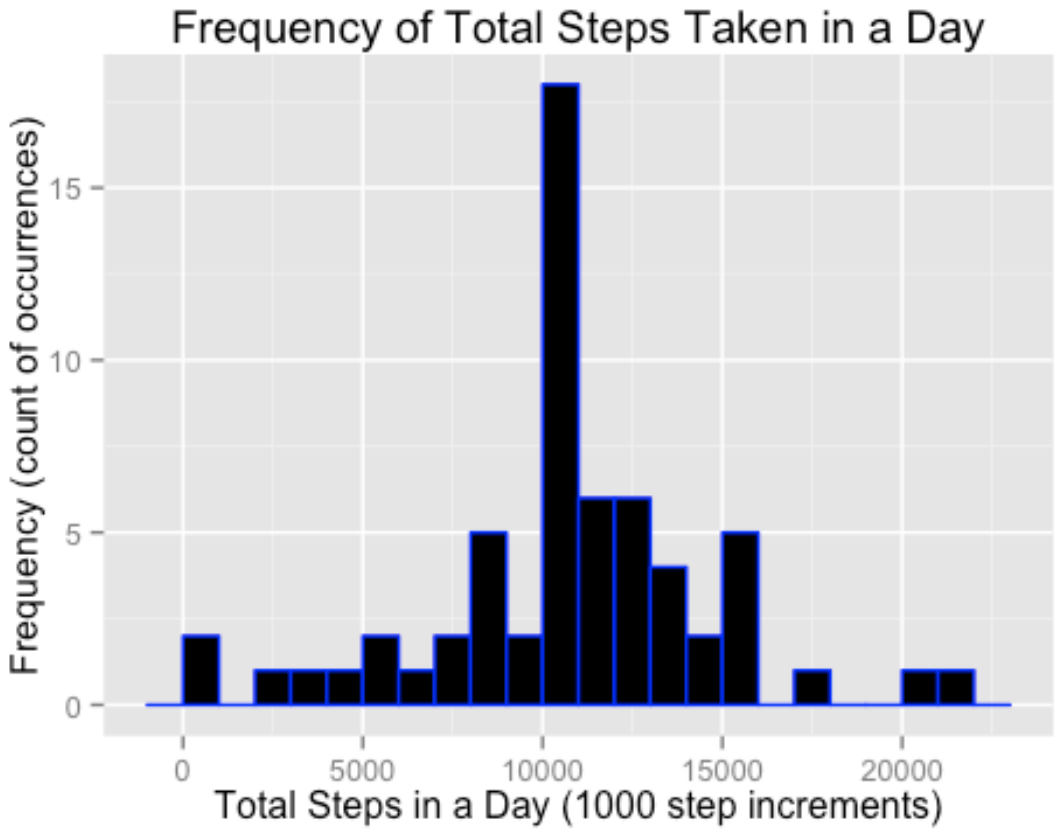
with the missing data filled in from the corresponding calculated 5-minute interval mean value.

```
## Copy original dataset
cdf <- df
## Add a column with interval as a factor
cdf <- mutate(cdf, finterval=as.factor(interval))
## Fill in missing value with mean average steps from corresponding 5 minute
interval
cdf$steps[is.na(cdf$steps)] <-
dfADAP$avgSteps[match(cdf$finterval[is.na(cdf$steps)],dfADAP$interval)]
```

Plot a histogram of the mean total number of steps taken per day.

Note that no data values are missing from this point on in the analysis.

```
## Calculate total steps per day
cdfSum <- aggregate(x=cdf$steps, by=list(cdf$date), FUN=sum)
## Rename aggregate data frame columns
cdfSum <- rename(cdfSum, date=Group.1, steps=x)

## Part 1 - Make a histogram of the total number of steps taken each day
## Histogram of total steps taken each day
## Construct the plot
g <- ggplot(cdfSum, aes(steps))

## Histogram of total steps frequency
g <- g + geom_histogram(binwidth=1000, colour = "blue", fill = "black")

## Add main label
g <- g + ggtitle("Frequency of Total Steps Taken in a Day")

## Add x and y axis labels
g <- g + xlab("Total Steps in a Day (1000 step increments)")
g <- g + ylab("Frequency (count of occurrences)")

## Display plot
print(g)
```



Calculate mean and median of total number of steps taken per day for complete dataset.

Provide a comparison of the calculated values between the original dataset and the complete dataset.

```
cmeanSteps <- mean(cdfSum$steps)
cmedianSteps <- median(cdfSum$steps)
sprintf("Mean total steps per day from complete dataset: %.1f", cmeanSteps)

## [1] "Mean total steps per day from complete dataset: 10766.2"

sprintf("Mean total steps per day from original dataset: %.1f", meanSteps)

## [1] "Mean total steps per day from original dataset: 10766.2"

sprintf("Median total steps per day from complete dataset: %.1f",
cmedianSteps)

## [1] "Median total steps per day from complete dataset: 10766.2"

sprintf("Median total steps per day from original dataset: %.1f",
medianSteps)
```

```
## [1] "Median total steps per day from original dataset: 10765.0"
```

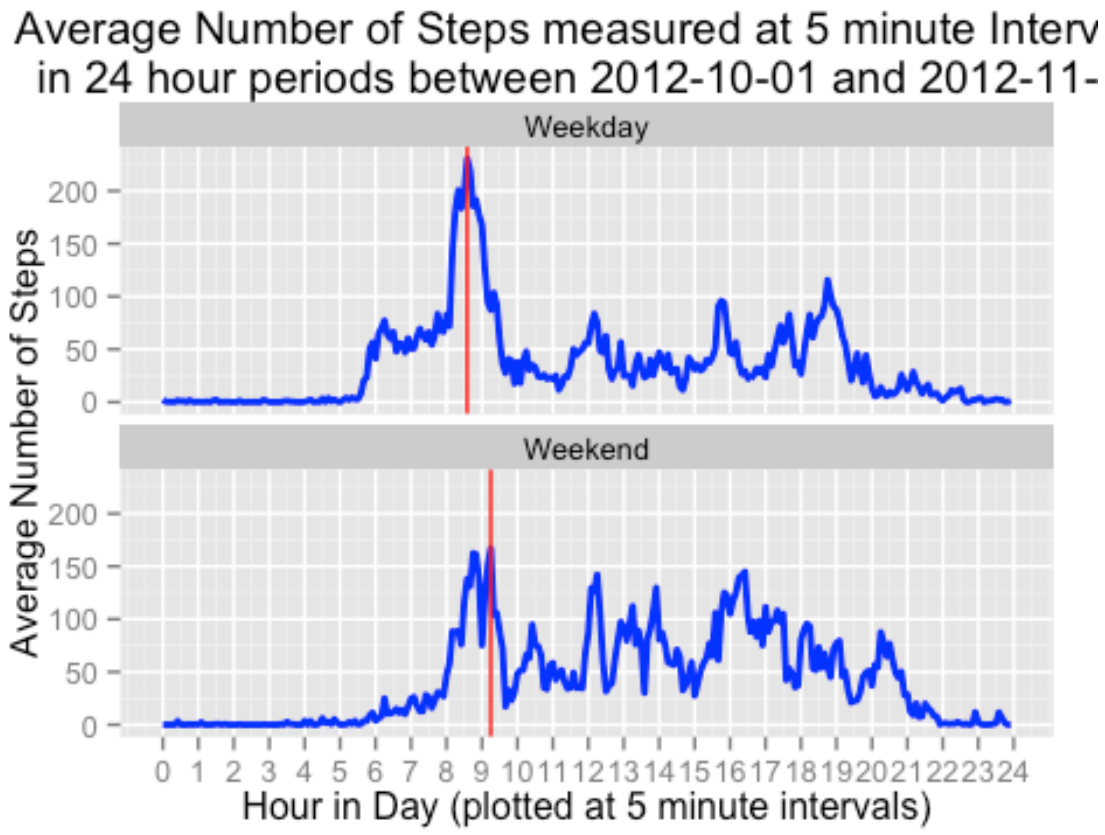
Impact of imputing missing values on mean and median calculations

The mean values between the original dataset (with missing values) and the dataset with complete values are identical. This is not surprising since the mean of the 5 minute intervals was used to fill in the missing values. The median values showed a very slight difference. There is essentially no impact from supplying missing values using the mean of the 5 minute intervals.

Differences in activity patterns between weekdays and weekends

```
## Part 1 - 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.
cdf <- mutate(cdf, isWeekend=weekdays(as.Date(cdf$date)) %in% c("Saturday",
"Sunday"))
cdf <- mutate(cdf,
dayType=as.factor(ifelse(cdf$isWeekend,"Weekend","Weekday")))

## Part 2 - 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.
dfWADAP <- aggregate(x=cdf$steps, by=list(cdf$finterval,cdf$dayType),
FUN=mean)
dfWADAP <- rename(dfWADAP, interval=Group.1, dayType=Group.2, avgSteps=x)
dfWADAP <- mutate(dfWADAP, hour=paste('0000', as.character(dfWADAP$interval),
sep=""))
dfWADAP$hour <- substr(dfWADAP$hour, nchar(dfWADAP$hour)-3,
nchar(dfWADAP$hour))
dfWADAP$hour <- as.numeric(substr(dfWADAP$hour, 1, 2)) +
as.numeric(substr(dfWADAP$hour, 3, 4)) / 60.0

## Construct the two panel line plot
g <- ggplot(dfWADAP, aes(hour, avgSteps))

g <- g + facet_grid(dayType ~ .) + facet_wrap( ~dayType, ncol=1)

## Line plot
g <- g + geom_line(aes(group=1), colour="blue", size=1)
g <- g + scale_x_continuous(breaks = seq(0, 24, by=1))

# Add vertical intercept for interval with peak activity
weekdaySteps <- filter(dfWADAP, dayType=="Weekday")
weekendSteps <- filter(dfWADAP, dayType=="Weekend")
maxWeekdaySteps <- filter(weekdaySteps, avgSteps==max(avgSteps))
maxWeekendSteps <- filter(weekendSteps, avgSteps==max(avgSteps))
```

```
maxSteps <- rbind(maxWeekdaySteps, maxWeekendSteps)
g <- g + geom_vline(aes(xintercept = maxSteps$hour), maxSteps, colour="red")

## Add main label
maintitle1 <- "Average Number of Steps measured at 5 minute Intervals\n in 24
hour periods"
maintitle2 <- paste("between",minDate,"and",maxDate)
title <- paste(maintitle1,maintitle2)
g <- g + ggtitle(title)

## Add x and y axis labels
g <- g + xlab("Hour in Day (plotted at 5 minute intervals)")
g <- g + ylab("Average Number of Steps")

## Display plot
print(g)
```




```
sprintf("Weekday 5-minute interval with max number of steps (averaged over
all days): %s", maxWeekdaySteps$interval)

## [1] "Weekday 5-minute interval with max number of steps (averaged over all
days): 835"
```

```
sprintf("Weekend 5-minute interval with max number of steps (averaged over
all days): %s", maxWeekendSteps$interval)
```

```
## [1] "Weekend 5-minute interval with max number of steps (averaged over all
days): 915"
```

The 5 minute interval of maximum activity for the whole dataset is the same as the one for weekdays alone.The weekend peak activity occurs in a 5 minute interval about 2/3 hour (40 minutes) later than the weekday peak activity 5 minute interval. There is a noticeable bifurcation in the weekend plot of 5 minute intervals. On the whole, peak weekend activity occurs later than peak weekday activity.