Activity Monitoring Project

Maya Acharya

November 10, 2014

Loading and Processing Data

Use read.csv to load the data into a variable, called **activity**. Print the column names of the data to see what you're working with.

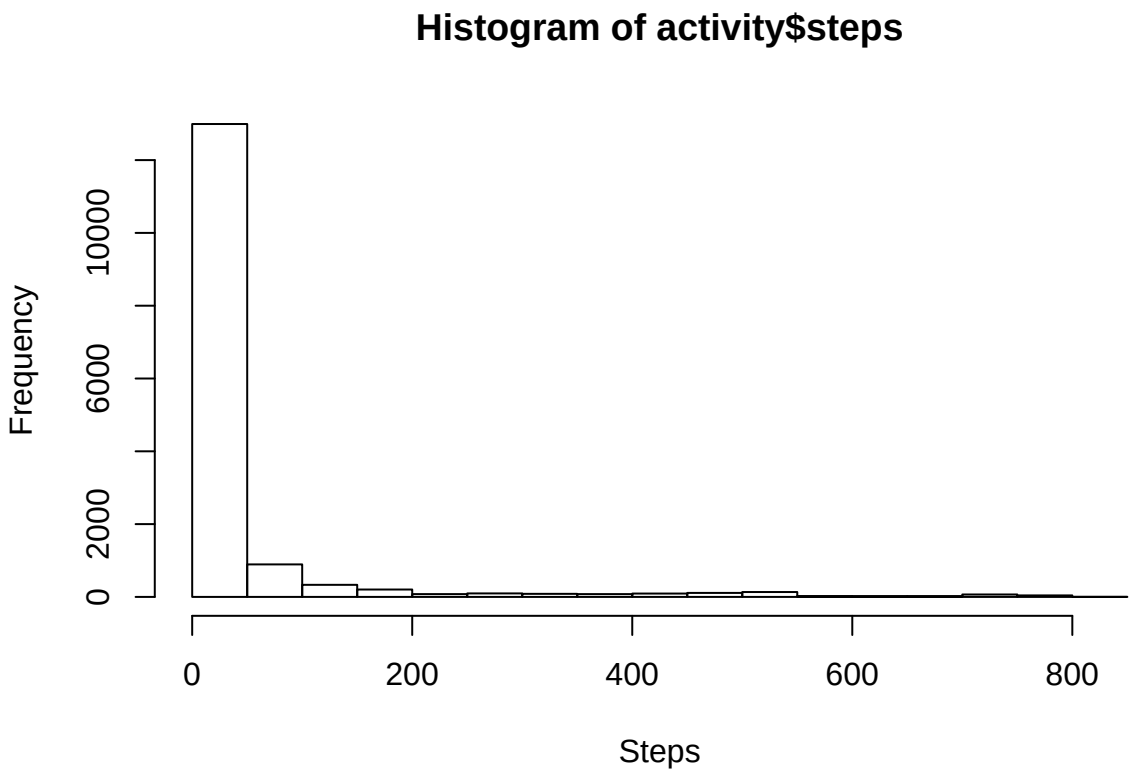
```
library(dplyr)
library(tidyr)
library(chron)
library(lattice)
library(ggplot2)
activity <- read.csv("activity.csv")
activity <- tbl_df(activity)
colnames(activity)
```

```
## [1] "steps"    "date"     "interval"
```

Summarizing the steps data

Make a histogram of the *steps* column of **activity** to visualize the data.

```
hist(activity$steps, xlab="Steps")
```



Find the *mean* and *median* of the *steps* data.

Use the *mean* and *median* functions on the **steps** column of the **activity** data.

```
step_mean <- mean(activity$steps, na.rm=TRUE)
step_med <- median(activity$steps, na.rm=TRUE)
{print(step_mean)
 print(step_med)}
```

```
## [1] 37.3826
## [1] 0
```

Mean of *steps* data: 37.3825996

Median of *steps* data: 0

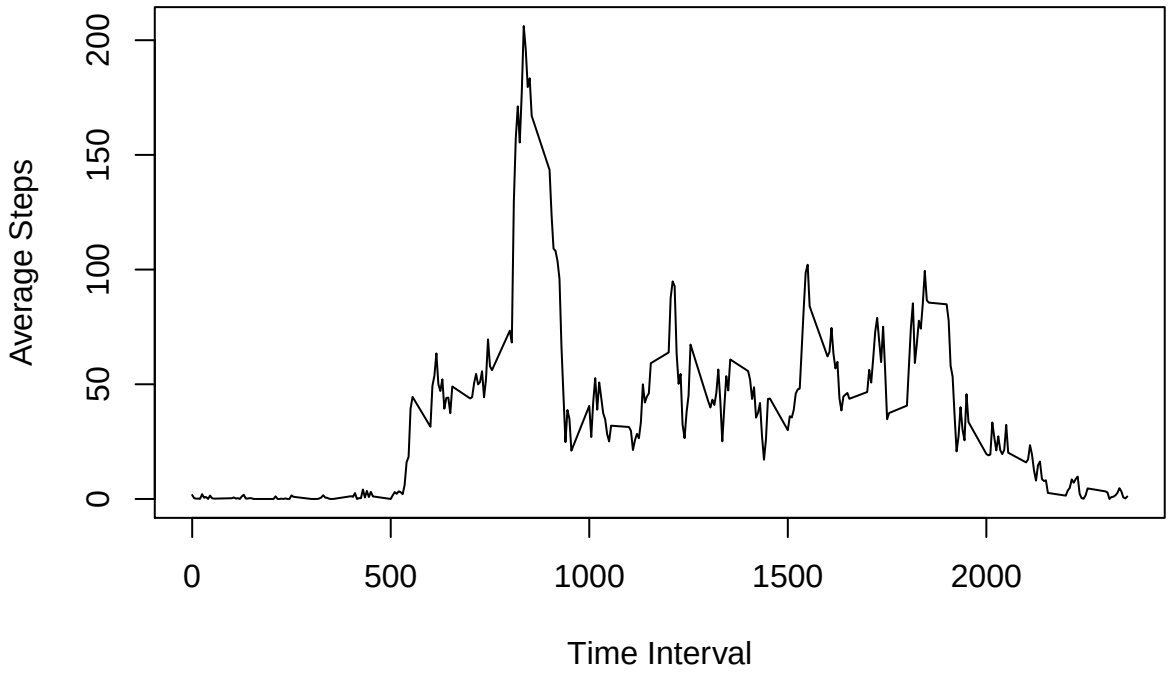
Visualizing time intervals over days

Group the data by interval. Use the *summarize* function to group the **activity** data, and find the mean of each interval.

```
time_interval_means <- summarize(group_by(activity, interval), mean(steps, na.rm=TRUE))
```

Use the *base plotting* system to plot the average steps by time interval.

```
plot(time_interval_means, type="l",xlab="Time Interval", ylab = "Average Steps")
```



To find the time interval with the highest average steps, use the *arrange* function to arrange the **time_interval_means** by descending average steps. The first row will be the time interval with the highest average steps.

```
colnames(time_interval_means) <- c("interval", "steps")
time_interval_means <- arrange(time_interval_means, desc(steps))
```

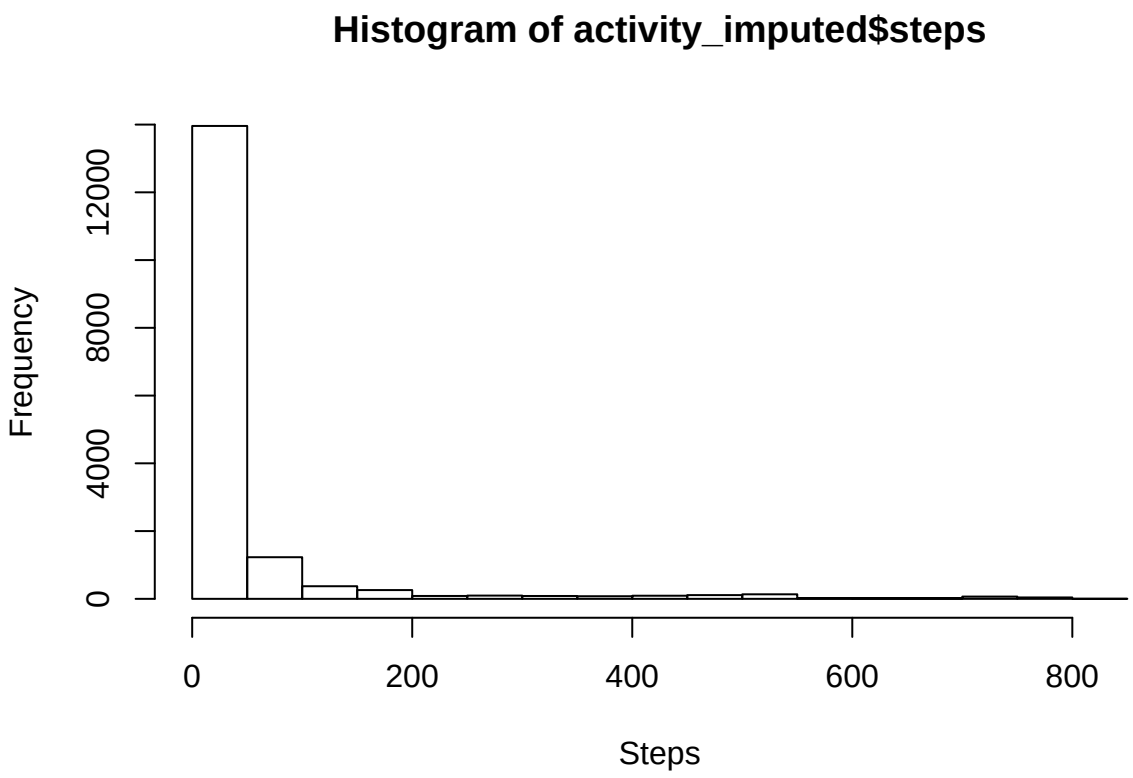
Interval with highest average steps: 835

(206.1698113 steps)

Imputing Missing Values

To impute missing values, I made each NA steps value equal to the average steps value for that time interval.

```
time_interval_means <- arrange(time_interval_means, interval)
activity_imputed <- activity
for(i in 1:nrow(activity)){
  if(is.na(activity[i,1])){
    activity_imputed[i,1] <- as.numeric(time_interval_means[(activity[i,3]/5)+1,2])
  }
}
hist(activity_imputed$steps, xlab="Steps")
```




```
step_mean_imputed <- mean(activity_imputed$steps, na.rm=TRUE)
step_med_imputed <- median(activity_imputed$steps, na.rm=TRUE)
{print(step_mean_imputed)
 print(step_med_imputed)}
```

```
## [1] 37.6206
## [1] 0
```

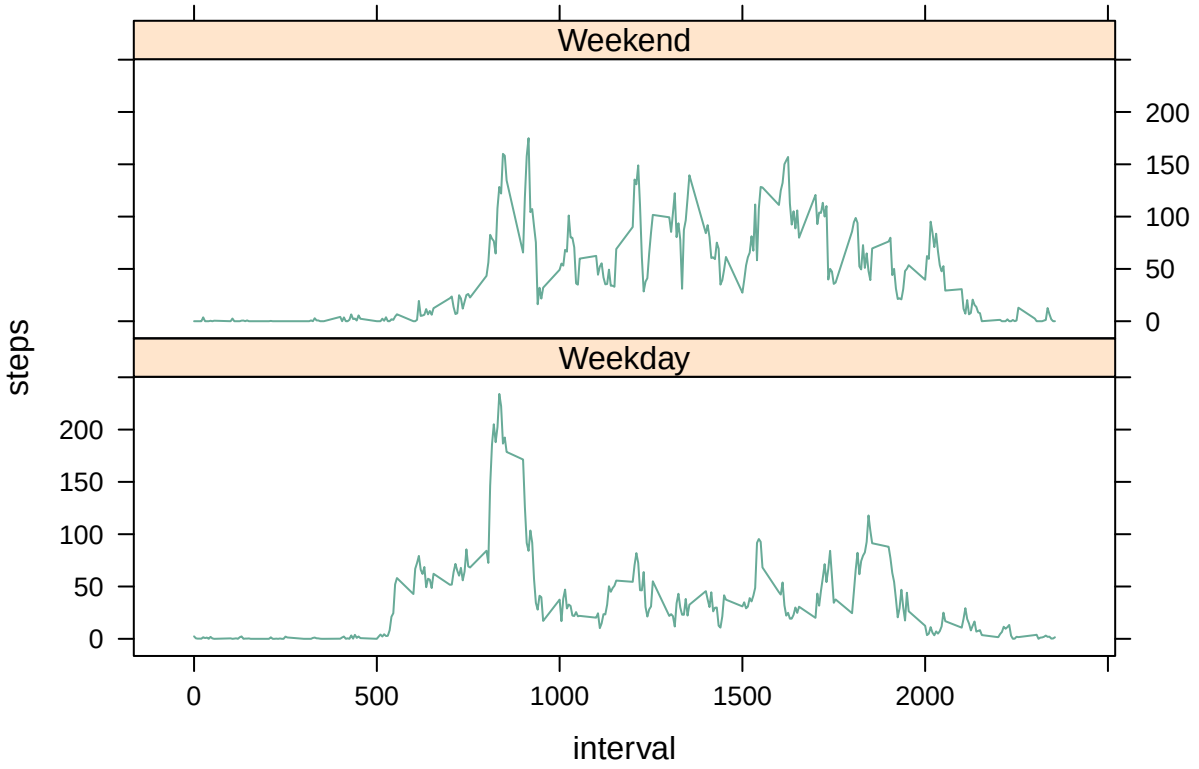
Mean of *steps* data with imputed missing values: 37.6206044

Median of *steps* data with imputed missing values: 0

Looking at Weekdays vs. Weekends

Use *dplyr*'s mutate function to split the **date** column into two factors, **TRUE** for days that are weekends, or **FALSE** for days that are during the work week. Store this in a new column called **day**. Then, use the *factor* function to relabel **day** with **Weekday** and **Weekend**. Use *lattice*'s *xyplot* function to create a time plot of the average steps per interval split over **Weekend**s and **Weekday**s

```
dayActivity <- mutate(activity, day=is.weekend(as.Date(activity$date)))
dayActivity$day <- factor(dayActivity$day, label=c("Weekday", "Weekend"))
dayActivity <- summarize(group_by(dayActivity, interval, day), mean(steps, na.rm=TRUE))
colnames(dayActivity) <- c("interval", "day", "steps")
xyplot(steps ~ interval | day, col="#68ab98", type="l", layout=c(1,2), data=dayActivity)
```



There is a clearly visible difference between the **steps** data for weekends and for weekdays. On weekends, the steps taken between the 500^{th} and 2000^{th} time intervals are consistently higher than the other time intervals. On weekdays, there is a sizeable spike from the 500^{th} to the 1000^{th} time interval.