Reproducible Research: Peer Assessment 1

Mike Kelly

Tuesday, February 10, 2015

# Exploratory Analysis of Raw data for Activity Monitoring

This is the Peer Assigment #1 for the Cousera class "Reproducible Research". Please note that I have included a couple of extra plots and calculations.

## Loading and preprocessing the data

The first step here is to load the data. The data can be found at (<https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2Factivity.zip>). The program checks to see if the file activity.csv already exists in the current working directory. If it does not, it then downloads the zip file and unzips it into the current working directory.

setwd("C:/Users/kellym/coursera/RepData\_PeerAssesment1/RepData\_PeerAssessment1")  
library("ggplot2")  
#############################################################  
# Set the file name and url variables.  
#############################################################  
zipName <- "activity.zip"  
fileName <- "activity.csv"  
fileUrl <- "https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2Factivity.zip"  
  
#############################################################  
# Check to see if the csv file exists in the current working   
# directory. If not, download the zip file and unzip it.  
#############################################################  
if(!file.exists(fileName)){  
 download.file(fileUrl,destfile=zipName, mode="wb")  
 unzip(zipName)  
}

After downloading the file, it is loaded into a dataframe (DF) **data** using read.table.

#############################################################  
# Load the activity.csv data into the data df.  
#############################################################  
data <- read.table(fileName  
 ,quote = "\""  
 , sep=","  
 , header=TRUE  
 , na.strings="NA"  
 , colClasses=c("numeric","Date","integer")  
)

A column is added to the **data** DF in order to view the interval and date as a full date-time value. In order to do this, the function interval\_date.Time was used.

interval\_date.Time = function(date, interval){  
 strptime(paste(date  
 ,sprintf("%04d",interval)  
 )  
 ,"%Y-%m-%d %H%M"  
 )  
}  
#  
### Add a combined columns to data for date and time  
#  
data$date.Time <- interval\_date.Time(data$date,data$interval)

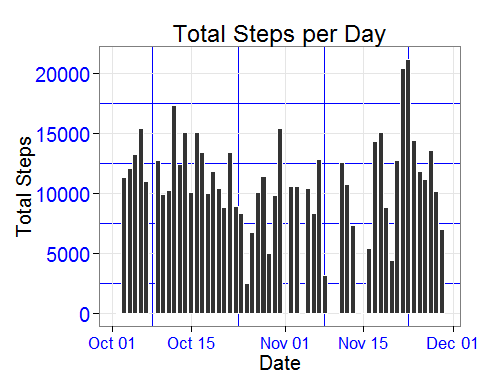
## What is mean total number of steps taken per day?

Part A, In order to get the total number of Steps per day we can ignore missing data. We first aggregate the DF data into a new DF called **steps\_by\_day**.

# Calculate the total steps per day  
steps\_by\_day <- aggregate(. ~date, data[,c("steps","date")], sum, na.rm=TRUE)

A plot of the total number of steps per day in the "activity.csv" data set is shown below.

p <- ggplot(steps\_by\_day, aes(date, steps))  
p <- p + geom\_histogram(stat="identity",colour="white")  
p <- p + theme\_bw( base\_size=15 ) +   
 labs(title="Total Steps per Day"  
 ,x="Date"  
 ,y="Total Steps"  
 ) +  
   
 theme(axis.text.x = element\_text(color="blue"  
 ,size=13  
 ,angle=0  
 ,vjust=0.5  
 ) ,  
 axis.text.y = element\_text(color="blue"  
 ,size=15  
 ,angle=0  
 ,vjust=0.5  
 ) ,  
 panel.grid.minor.x=element\_line(size = 0.1,color="blue"),  
 panel.grid.minor.y=element\_line(size = 0.1,color="blue")  
)   
print(p)



The mean Number ofsteps per day is:

print(mean(as.numeric(steps\_by\_day$steps), na.rm=TRUE))

## [1] 10766.19

and the median of steps per day is:

print(median(as.numeric(steps\_by\_day$steps), na.rm=TRUE))

## [1] 10765

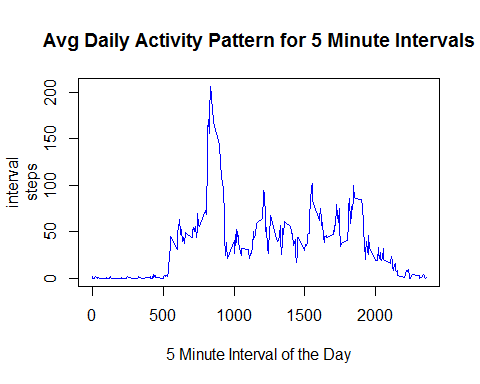
## What is the average daily activity pattern?

Next, we determine the overall daily activity pattern, we will calculate the average number of steps that were taken during each of the 5 minute intervals throughout the day. In order to do this we first aggregate the DF data into a new DF called **mean\_steps\_per\_interval**. We calculate the interval means as:

mean\_steps\_per\_interval <- aggregate(. ~interval, data[,c("steps","interval")], mean, na.rm=TRUE)

Here is a plot for the average daily activity:

plot(mean\_steps\_per\_interval, aes(interval, steps), type="l"  
 ,xlab="5 Minute Interval of the Day"  
 ,col="blue"  
)  
title ("Avg Daily Activity Pattern for 5 Minute Intervals")



The 5 minute interval that has the maximun average number of steps is:

max\_avg <- max(as.numeric(mean\_steps\_per\_interval$steps), na.rm=TRUE)   
print(max\_avg)

## [1] 206.1698

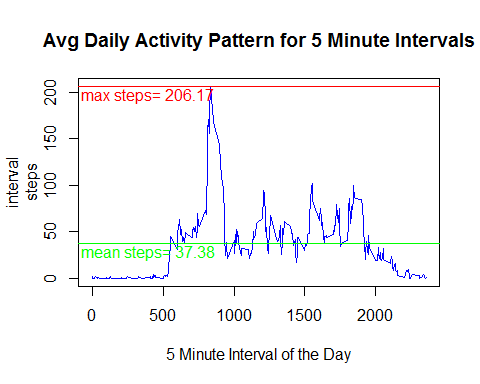
Let us also calcuate the mean average number of steps taken per interval:

mean\_avg <- mean(as.numeric(mean\_steps\_per\_interval$steps), na.rm=TRUE)   
print(mean\_avg)

## [1] 37.3826

Now let us replot the daily activity again and overlay it with a line at the max value and the mean values:

plot(mean\_steps\_per\_interval, aes(interval, steps), type="l"  
 ,xlab="5 Minute Interval of the Day"  
 ,col="blue"  
)  
title ("Avg Daily Activity Pattern for 5 Minute Intervals")  
abline(col="red", ,text(400,(max\_avg - 10 ), paste("max steps=",round(max\_avg,digits=2)), col = "red"), h=max\_avg)  
abline(col="green", ,text(400,(mean\_avg - 10 ), paste("mean steps=",round(mean\_avg,digits=2)), col = "Green"), h=mean\_avg)



## Imputing missing values

We find that the data set has a large number of missing values for the steps variable. The total number of records with missing step values is:

number\_missing\_values <- sum(is.na(data$steps)==TRUE)  
print(number\_missing\_values)

## [1] 2304

Next, we impute the missing values into a new data set called **new\_data**. First we create **new\_data** as a copy of **data**. Now the strategy used to populate the missing step values is to calculate the mean value of its interval during the day across the entire data set. This information is already stored in the dataset **mean\_steps\_per\_interval**. We loop over the rows in **data** and copy the step value if it exists or replace it with the calculated mean if is missing with the value for that interval in **mean\_steps\_per\_interval**.

new\_data <- data   
x <- mean\_steps\_per\_interval # create a temporary df x  
for(i in 1:nrow(data) ){  
 if(is.na(data[i,1])){  
 i\_interval <- data[i,"interval"]  
 new\_data[i,1] <- x[x$interval==i\_interval,]$steps  
 }  
 else{ new\_data[i,1] <- data[i,1]}  
}  
rm(x)

A check of **new\_data** for missing values shows that it does not have any missing values. The number of new\_data missing vaules is:

new\_data\_missing\_values<- sum(is.na(new\_data$steps)==TRUE)  
print(new\_data\_missing\_values)

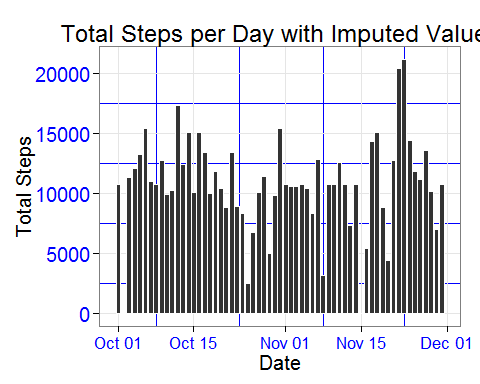
## [1] 0

Next we want to create a histogram of the total steps per day for the new data set **new\_data** and then compare it to the data set with the missing values. First aggregate the new data DF into a new DF called **new\_steps\_by\_day** over steps and date as we did in the original histogram.

new\_steps\_by\_day <- aggregate(. ~date, new\_data[,c("steps","date")], sum, na.rm=TRUE)

Now, plot the histogram for the total number of steps per day:

p <- ggplot(new\_steps\_by\_day, aes(date, steps))  
p <- p + geom\_histogram(stat="identity",colour="white")  
p <- p + theme\_bw( base\_size=15 ) +   
 labs(title="Total Steps per Day with Imputed Values"  
 ,x="Date"  
 ,y="Total Steps"  
 ) +  
   
 theme(axis.text.x = element\_text(color="blue"  
 ,size=13  
 ,angle=0  
 ,vjust=0.5  
 ) ,  
 axis.text.y = element\_text(color="blue"  
 ,size=15  
 ,angle=0  
 ,vjust=0.5  
 ) ,  
 panel.grid.minor.x=element\_line(size = 0.1,color="blue"),  
 panel.grid.minor.y=element\_line(size = 0.1,color="blue")  
 )   
print(p)



The mean of the new data set with the imputed values is not much different that the mean of the original data set with missing values and the same for the median. The mean new steps per day for new\_data is:

print(mean(as.numeric(new\_steps\_by\_day$steps), na.rm=TRUE))

## [1] 10766.19

The median of the New distribution of steps per day is:

print(median(as.numeric(new\_steps\_by\_day$steps), na.rm=TRUE))

## [1] 10766.19

To better see the differences, let us make a panel plot of the new and original histograms for steps per day. We will add a new factor column **imputed**, which can be Yes or No, to the **steps\_by\_day** and the **new\_steps\_by\_day** DF's.

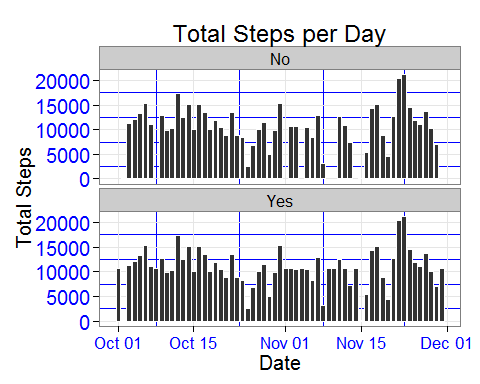
steps\_by\_day$imputed <- "No"  
new\_steps\_by\_day$imputed <- "Yes"

Then we rbind them into a new DF **all\_steps\_by\_day**

all\_steps\_by\_day <- rbind(steps\_by\_day, new\_steps\_by\_day)

Here is a panel plot of the steps per day comparing the original data set and the new imputed data set:

par(mfrow=c(2,1)) # divide window into 2  
p <- ggplot(all\_steps\_by\_day, aes(date, steps)) + facet\_wrap(~imputed,nrow=2)  
p <- p + geom\_histogram(stat="identity",colour="white")  
  
p <- p + theme\_bw( base\_size=15 ) +   
 labs(title="Total Steps per Day"  
 ,x="Date"  
 ,y="Total Steps"  
 ) +  
   
 theme(axis.text.x = element\_text(color="blue"  
 ,size=13  
 ,angle=0  
 ,vjust=0.5  
 ) ,  
 axis.text.y = element\_text(color="blue"  
 ,size=15  
 ,angle=0  
 ,vjust=0.5  
 ) ,  
 panel.grid.minor.x=element\_line(size = 0.1,color="blue"),  
 panel.grid.minor.y=element\_line(size = 0.1,color="blue")  
)   
print(p)



## Are there differences in activity patterns between weekdays and weekends?

First, let us add a factor column to each of the new\_data DF for the day of the week and the week day type weekDay or weekEnd. We use a function called dayType to determine if it is a week day or weekend.

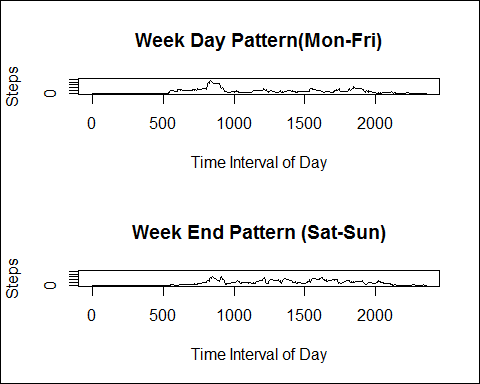
dayType <- function(weekDay){  
 if (weekDay %in% c("Mon","Tue","Wed","Thu","Fri")) dayType <- "weekDay"  
 if (weekDay %in% c("Sat","Sun")) dayType <- "weekEnd"  
 return(dayType)  
}  
  
new\_data$dayOfWeek <- weekdays(new\_data$date.Time, abbreviate = TRUE)  
new\_data$dayType <- sapply(new\_data$dayOfWeek, FUN=dayType)

Now, create two seperate DF's, **weekDays** which aggregates the 5 minute intervals all all week days (Mon,Tue,Wed,Thu,Fri) over the entire date range and a second which aggregates the 5 minute intervals all all week days (Sat,Sun)

weekDays <- aggregate(. ~interval, new\_data[new\_data$dayType=="weekDay",c("steps","interval")], mean, na.rm=TRUE)  
weekEnds <- aggregate(. ~interval, new\_data[new\_data$dayType=="weekEnd",c("steps","interval")], mean, na.rm=TRUE)  
#### Rename the Columns  
names(weekDays) <- c("interval","weekDaySteps")  
names(weekEnds) <- c("interval","weekEndSteps")

A panel plot for the week end and week day distributions is shown below:

par(mfrow=c(2,1)) # divide window into 2  
plot(weekDays$interval, weekDays$weekDaySteps,type="l"  
 ,xlab="Time Interval of Day"  
 ,ylab="Steps"  
 ,ylim=c(0,250))  
title ("Week Day Pattern(Mon-Fri)")  
plot(weekEnds$interval, weekEnds$weekEndSteps,type="l"  
 ,xlab="Time Interval of Day"  
 ,ylab="Steps"  
 ,ylim=c(0,250))  
title ("Week End Pattern (Sat-Sun)")  
box(which="outer", lty="solid")

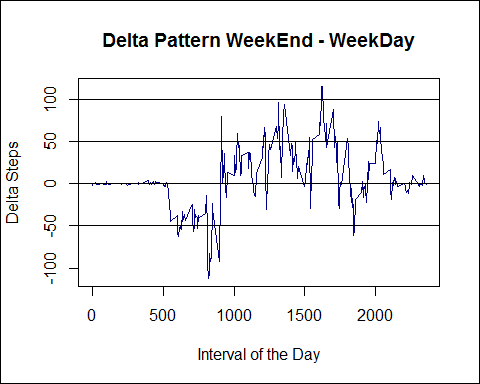


Another way to view the difference is to subtract one distribution from the other. So we first merge the two new data sets **weekDays** and **weekEnds**:

weekDay\_all <- merge(weekDays,weekEnds  
 ,by.weekDays="interval"  
 ,by.weekEnds="interval"  
 ,all=TRUE  
)

Now, we plot the difference of the interval values between the two data sets **weekDays** and **weekEnds**.

par(mfrow=c(1,1)) # divide window into 1  
plot(weekDay\_all$interval, (weekDay\_all$weekEndSteps - weekDay\_all$weekDaySteps),type="l"  
 ,xlab="Interval of the Day"  
 ,ylab="Delta Steps"  
 ,col="darkblue"  
 ,main="Delta Pattern WeekEnd - WeekDay")  
abline(h=0,abline(h=c(-50, 50,100)))  
box(which="outer", lty="solid")



Finally, we will also add an overlay plot showing the two distributions. We will add the factor dayType to the two data sets **weekDays** and **weekEnds**. The we rbind these data sets into a new data set called \*\*weekDay\_interval:

#################################  
# Now put on the same plot  
#################################  
weekDays$dayType <- "weekDay"  
weekEnds$dayType <- "weekEnd"  
names(weekDays) <- c("interval","steps","dayType")  
names(weekEnds) <- c("interval","steps","dayType")  
weekDay\_interval <- rbind(weekDays,weekEnds)

Here is the resulting overlay plot showing the two distributions:

qplot(interval, steps, data = weekDay\_interval, group = dayType, color = dayType,  
 geom = c("line"), ylab = expression("Total Steps Per Interval"),   
 xlab = "Interval", main = "Total Steps per Time Interval")

