Activity assignment

Sebastian Ospina Valencia

4 de febrero de 2018

Loading the data

The first step to analyze the data is to load the dataset previously downloaded from https://www.coursera.org/learn/reproducible-research/peer/gYyPt/course-project-1.

```
activity<-read.csv("activity.csv")
head(activity)
```

```
##   steps       date interval
## 1    NA 2012-10-01        0
## 2    NA 2012-10-01        5
## 3    NA 2012-10-01       10
## 4    NA 2012-10-01       15
## 5    NA 2012-10-01       20
## 6    NA 2012-10-01       25
```

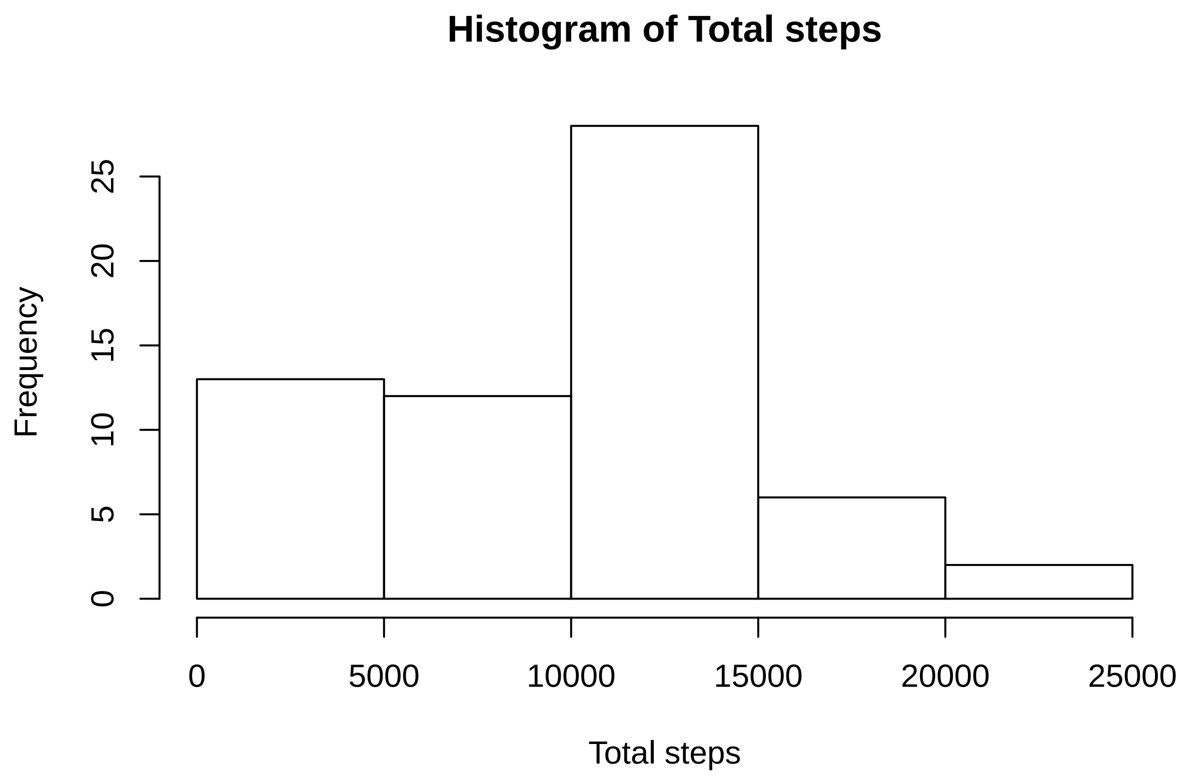
Calculate mean total number of steps per day

In order to calculate the mean total of steps per day we will first transform the dates as a factor, then we will use tapply to return a dataframe with the total number of steps per day that will be named as "totaldates".

```
activity$date<-as.factor(activity$date)
totaldates<-as.data.frame(tapply(activity$steps,activity$date,FUN=sum,na.rm=TRUE))

#we will rename the colnames of the totaldates matrix
colnames(totaldates)<-"total_steps"
```

Then we will proceed to calculate the histogram of the total steps by day in order to see de distribution of that variable:

```
hist(totaldates$total_steps,xlab = "Total steps",main = "Histogram of Total steps")
```



With this dataframe we will proceed to calculate the average and the median of the total steps.

```
mean(totaldates$total_steps)
```

```
## [1] 9354.23
```

```
median(totaldates$total_steps)
```

```
## [1] 10395
```

What is the average daily activity pattern?

To answer the question we will proceed with the plyr and dplyr packages in order to use other tools. Hence we will summarize the activity dataframe with the mean of steps by 5 minutes interval. So we will load the packages and then we will create a grouped data frame (grouped_data). After this, we will use the summarize function to summarise the data.

Finally in order to get the 5 minutes interval with the maximun mean of steps we will uste the which.max() function to find it

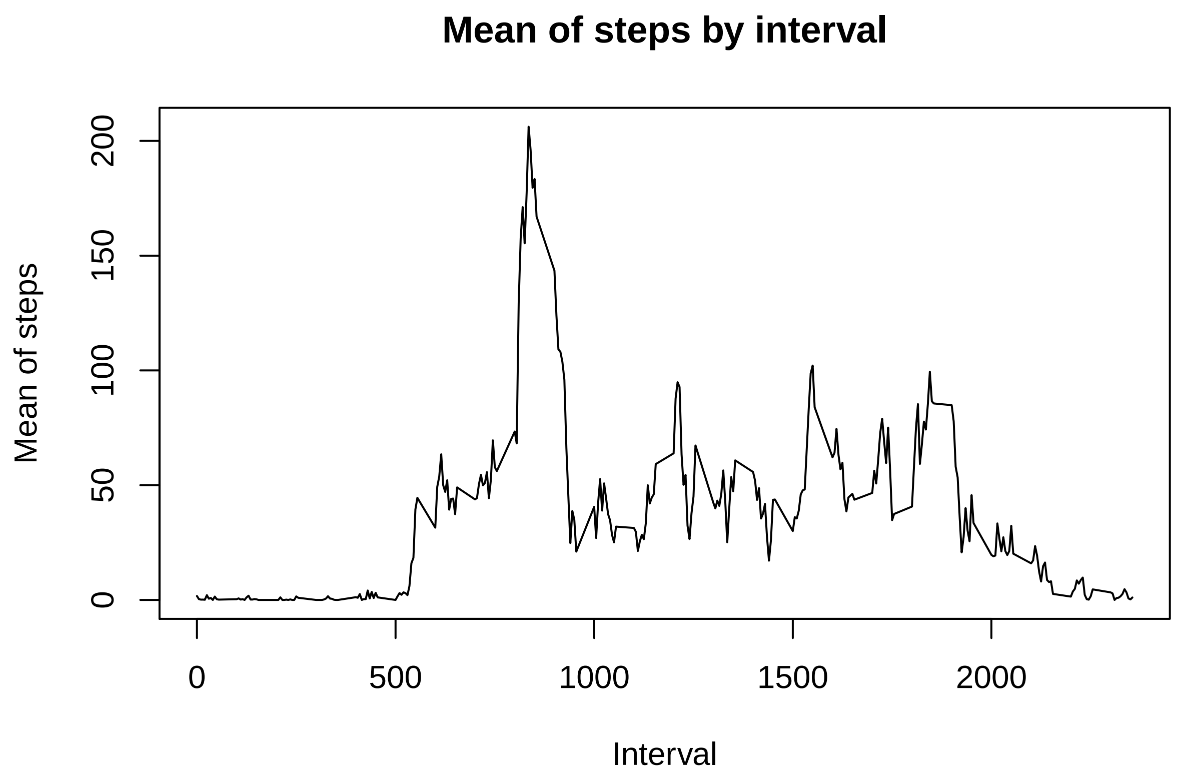
```
library(plyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:plyr':
##
```

```
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
#Grouping the data
grouped_data<-group_by(activity,activity$interval)

sumdata<-summarise(grouped_data,mean_steps_interval=mean(steps,na.rm=TRUE))

plot(sumdata$`activity$interval`,sumdata$mean_steps_interval,type = "l",main = "Mean of steps by interva
```



```
sumdata$`activity$interval`[which.max(sumdata$mean_steps_interval)]
```

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

Imputing NA's

The strategy to inpute the NA's will be done as follows:

- First we will search how many observations in the dataset there is any missing value

```
sum(is.na(activity$steps))
```

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

- With these values identified we will inpute them the mean of the five minutes in which the observation was identified.This imputation will be made on a new dataframe called complete_dataset

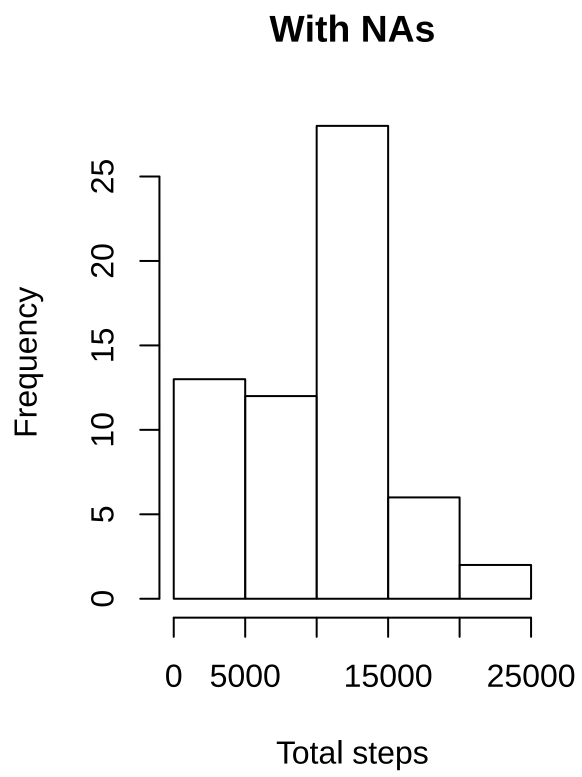
```
complete_dataset<-activity

for(i in 1:length(sumdata$`activity$interval`)){
    complete_dataset$steps[is.na(complete_dataset$steps) & complete_dataset$interval==sumdata$`activity$
}
```

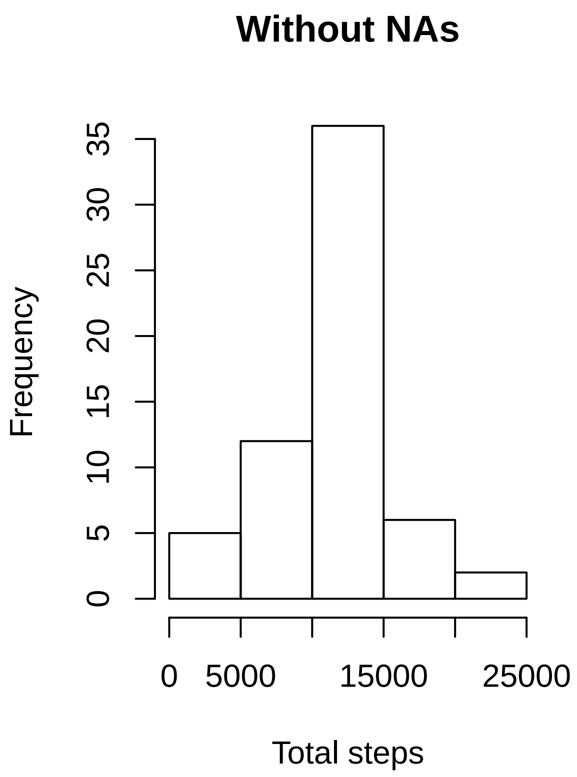
- In order to see the modifications that this imputation does to the structure of the dataset we will plot the histogram of the mean total number of steps per day whith and without the imputations. Hence, we will the first step procedure but for the complete dataset.

```
complete_dataset$date<-as.factor(activity$date)
totaldates1<-as.data.frame(tapply(complete_dataset$steps,complete_dataset$date,FUN=sum,na.rm=TRUE))

#we will rename the colnames of the totaldates matrix
colnames(totaldates1)<-"total_steps"
```

- Then we will plot both histograms

```
par(mfrow=c(1,2))
hist(totaldates$total_steps,xlab = "Total steps",main = "With NAs")

hist(totaldates1$total_steps,xlab = "Total steps",main = "Without NAs")
```





Are there differences in activity patterns between weekdays and weekends?

For this approach we will use the completed dataset, first we will create a viariable with the name of the days of the week (This names of the days are in Spanish) sabado = saturday, domingo = sunday. Then we will create the factor variable week and weekend in order to plot the behavior of the steps.

- The next chunk of code will show how we will create the days of the week. Becouse dates where previously transformed toa factor variable we have to transform the dates variable first into a character and then into a POSIXlt object as is shown next:

```
complete_dataset$wday<-weekdays(as.POSIXlt(as.character(complete_dataset$date)))
```

- After this, we can create the the weekend factor variable

```
complete_dataset$weekend<-"week"
complete_dataset$weekend[complete_dataset$wday=="sábado"]<-"weekend"
complete_dataset$weekend[complete_dataset$wday=="domingo"]<-"weekend"
complete_dataset$weekend<-as.factor(complete_dataset$weekend)
```

- Before making any plot, we have to transform our dataset into a panel data summarizing the steps by weekend factor variable and with five minutes interval. To do this we will use melt function.

```
library(reshape)
```

```
##
## Attaching package: 'reshape'
```

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

```
##
##     rename
## The following objects are masked from 'package:plyr':
##
##     rename, round_any
```

```
rearrenged_data<-melt(complete_dataset,id=c("interval","weekend"),measure.vars = "steps")
```

- Then using cast function we can summarize the arrenged dataset in order to plot the data.

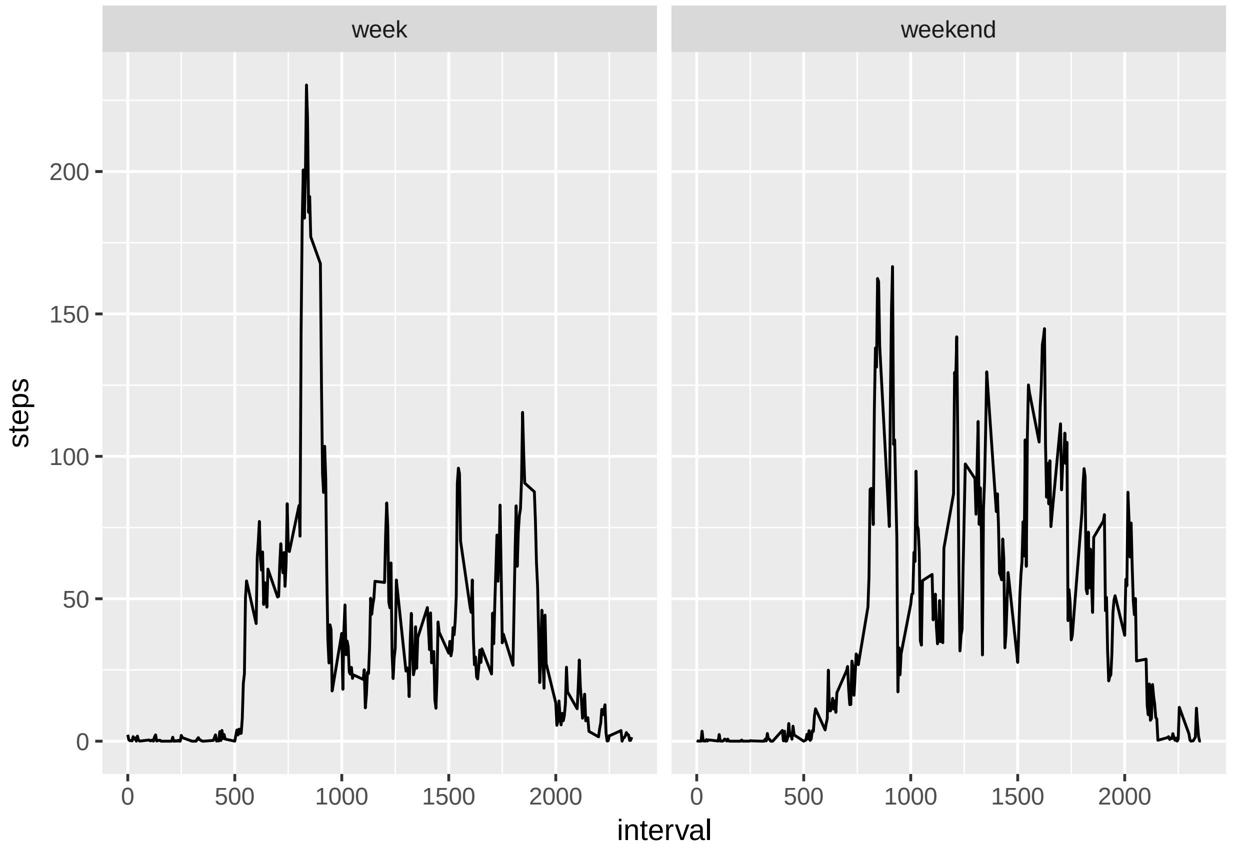
```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
## The following objects are masked from 'package:reshape':
##
##     colsplit, melt, recast
```

```
castedData<-dcast(rearrenged_data,interval+weekend~variable,value.var = "value",mean)
```

- To this particular case we will use ggplot2 package to create a fancy (an without much effort ;)) plot

```
library(ggplot2)

dcastedplot<-ggplot(castedData,aes(interval,steps))+geom_line()+facet_grid(.~weekend)
dcastedplot
```


week
weekend
steps
interval