


Reproducible Research: Week 2

Course Project

1.Loading and preprocessing the data

```
setwd("C:/Users/Mohsen/Desktop/Data.Science.Certificate.Coursera2020/Course 5. Reprod  
  
mydata0<-read.csv("activity.csv")  
  
str(mydata0)
```



```
## 'data.frame':   17568 obs. of  3 variables:  
## $ steps    : int   NA NA NA NA NA NA NA NA NA ...  
## $ date      : chr   "2012-10-01" "2012-10-01" "2012-10-01" "2012-10-01" ...  
## $ interval: int    0 5 10 15 20 25 30 35 40 45 ...
```

2.What is mean total number of steps taken per day?

2.1.Calculate the total number of steps taken per day

```
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
```

```
mydata1 <-mydata0[complete.cases(mydata0),]%>%  
  group_by(date) %>%  
  summarise(Total_Steps=sum(steps) )
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

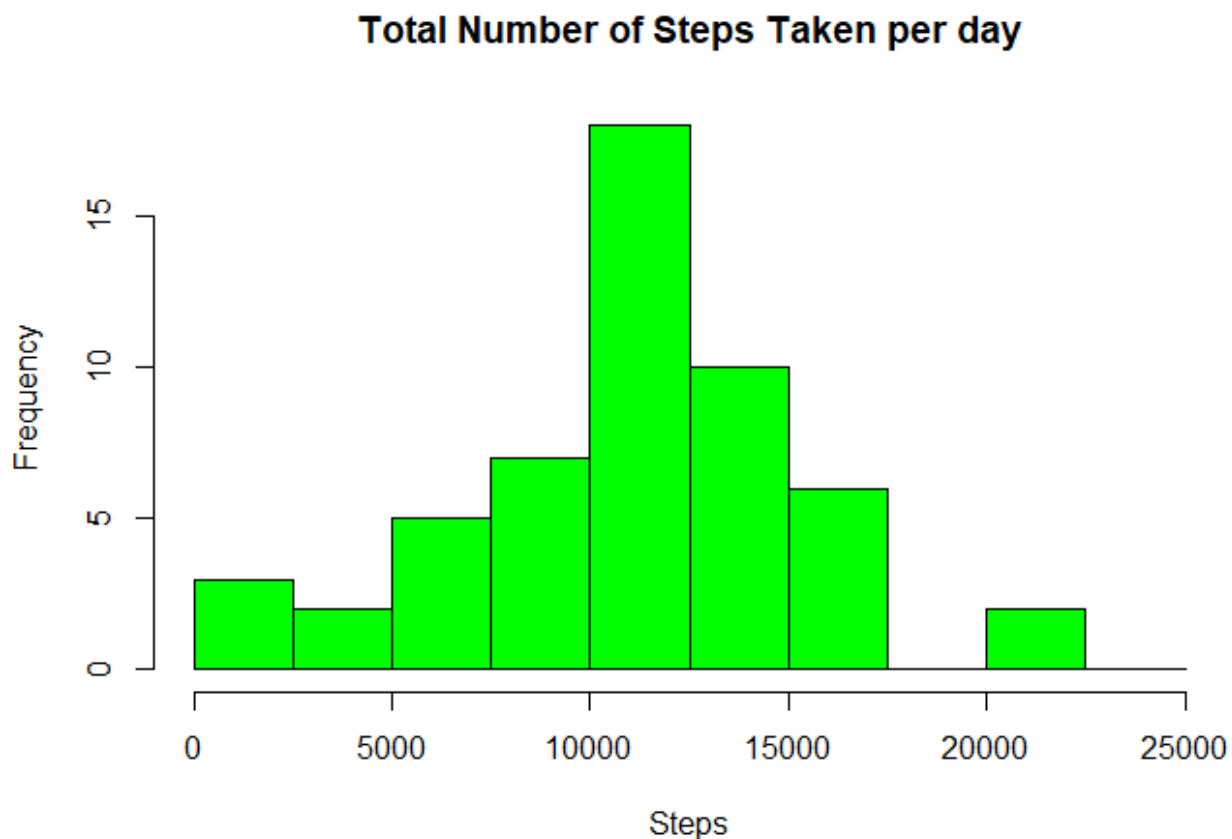
```
dim(mydata1)
```

```
write.csv(mydata1,"mydata1.csv")
```

2.2.Make a histogram of the total number of steps taken each day

```
Steps<- mydata1$Total_Steps
```

```
hist(Steps, col = "green", main = "Total Number of Steps Taken per day", breaks = c(0
```



2.3. Calculate and report the mean and median of the total number of steps taken per day

```
Mean_Steps=mean(Steps)
Median_Steps=median(Steps)
A=matrix(c("Mean", "Median", Mean_Steps, Median_Steps),nrow = 2, byrow = T)
print(A)
```

```
##      [,1]      [,2]
## [1,] "Mean"    "Median"
## [2,] "10766.1886792453" "10765"
```

3. What is the average daily activity pattern?

3.1. Make a time series plot (i.e. type = "l" type = "l") of the 5-minute interval (x-axis) and the average number of steps taken, averaged across all days (y-axis)

```
library(dplyr)
mydata2 <- mydata0[complete.cases(mydata0), ] %>%
  group_by(interval) %>%
  summarise(Mean_Steps=mean(steps) )

## `summarise()` ungrouping output (override with `.groups` argument)

dim(mydata2)

## [1] 288 2

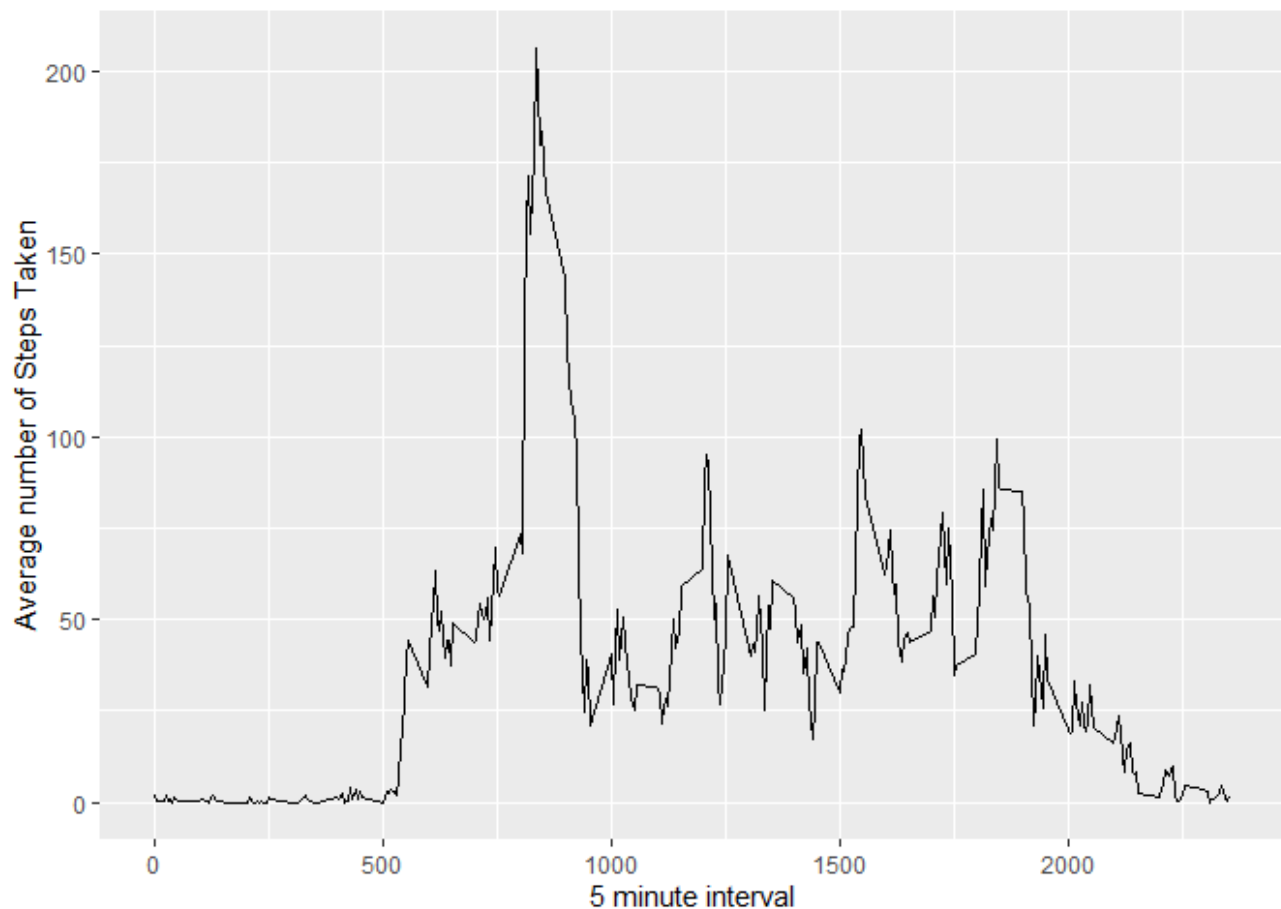
write.csv(mydata2, "mydata2.csv")

## Plot
library(ggplot2)
par(mfrow=c(1, 1))
```

```
x<-ts(mydata2$interval)
y<-ts(mydata2$Mean_Steps)
ggplot(mydata2, aes(x, y)) +
  geom_line(na.rm=TRUE) +
  xlab("5 minute interval") + ylab("Average number of Steps Taken")
```

Don't know how to automatically pick scale for object of type ts. Defaulting to co

Don't know how to automatically pick scale for object of type ts. Defaulting to co



3.2.Which 5-minute interval, on average across all the days in the dataset, contains the maximum number of steps?

```
a<-c()
b<-c()
M=max(mydata2$Mean_Steps)
l=length(mydata2$Mean_Steps)
for(i in 1:l){
  if (mydata2$Mean_Steps[i]==M){a<-mydata2$interval[i] }
  if (mydata2$Mean_Steps[i]==M){ b<- mydata2$interval[i]+5 }
```

```
}  
c(a,b)
```

```
## [1] 835 840
```

4.Imputing missing values

4.1.Calculate and report the total number of missing values in the dataset (i.e. the total number of rows with NA)

```
#library(dplyr)  
sum(Reduce('|', lapply(mydata0, is.na)))
```

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

4.2.Devise a strategy for filling in all of the missing values in the dataset. The strategy does not need to be sophisticated. For example, you could use the mean/median for that day, or the mean for that 5-minute interval, etc.

```
# impute with mean value  
library(Hmisc)
```

```
## Warning: package 'Hmisc' was built under R version 4.0.2
```

```
## Loading required package: lattice
```

```
## Loading required package: survival
```

```
## Loading required package: Formula
```

```
##
```

```
## Attaching package: 'Hmisc'
```

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

```
##
```

```
##      src, summarize
```

```
## The following objects are masked from 'package:base':
##
##   format.pval, units
```

```
mydata0$steps<-with(mydata0, impute(steps,mean))
mydata0$date<-with(mydata0, impute(date,mean))
mydata0$interval<-with(mydata0, impute(interval,mean))
```

4.3.Create a new dataset that is equal to the original dataset but with the missing data filled in.

```
mydata3impute<-mydata0
write.csv(mydata3impute, "mydata3impute.csv")
head(mydata3impute)
```

```
##      steps      date interval
## 1 37.3826 2012-10-01         0
## 2 37.3826 2012-10-01         5
## 3 37.3826 2012-10-01        10
## 4 37.3826 2012-10-01        15
## 5 37.3826 2012-10-01        20
## 6 37.3826 2012-10-01        25
```

4.4.Make a histogram of the total number of steps taken each day and Calculate and report the mean and median total number of steps taken per day. Do these values differ from the estimates from the first part of the assignment? What is the impact of imputing missing data on the estimates of the total daily number of steps?

```
library(dplyr)
mydata4 <-mydata3impute%>%
  group_by(date) %>%
  summarise(Total_Steps=sum(steps) )

## `summarise()` ungrouping output (override with `.groups` argument)

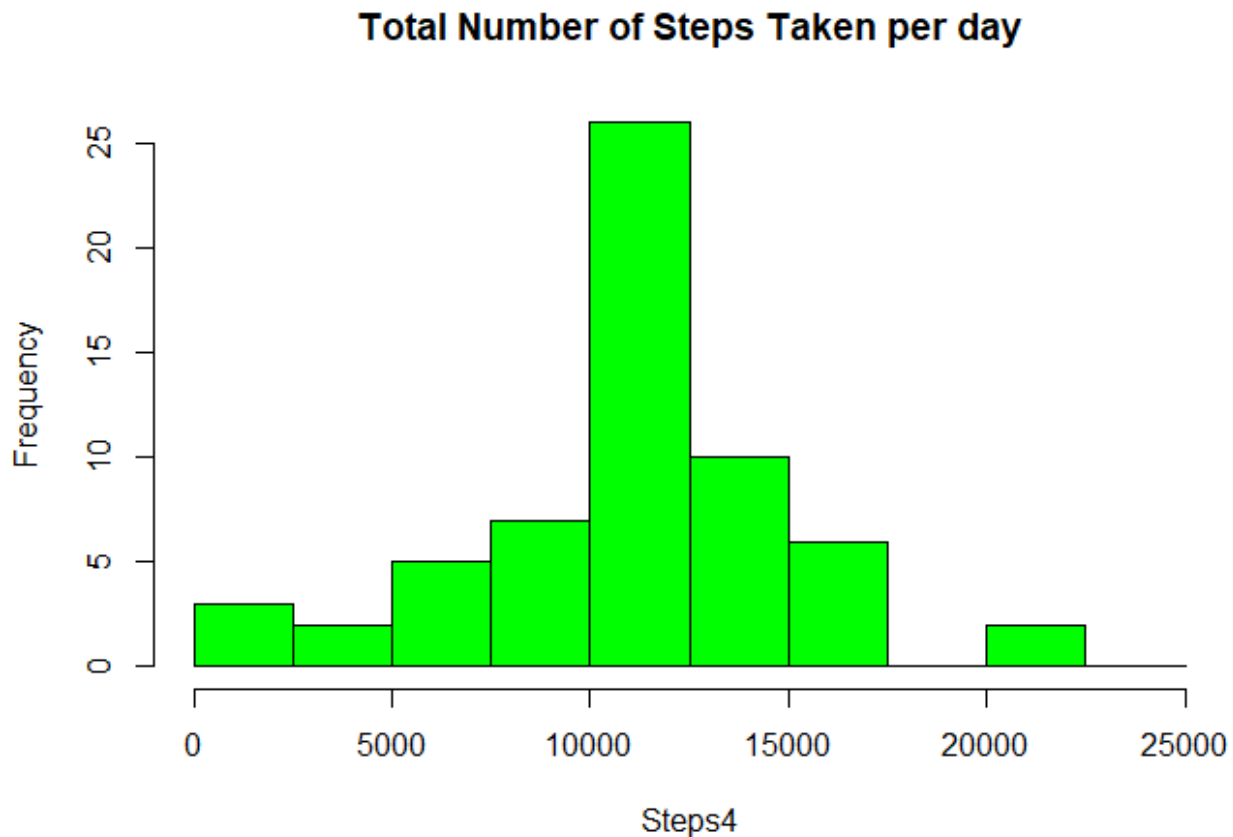
dim(mydata4)
```

```
## [1] 61 2
```

```
write.csv(mydata4,"mydata4.csv")
```

```
Steps4<- mydata4$Total_Steps
```

```
hist(Steps4, col = "green", main = "Total Number of Steps Taken per day", breaks = c(
```

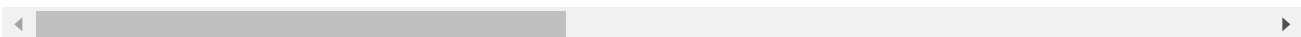


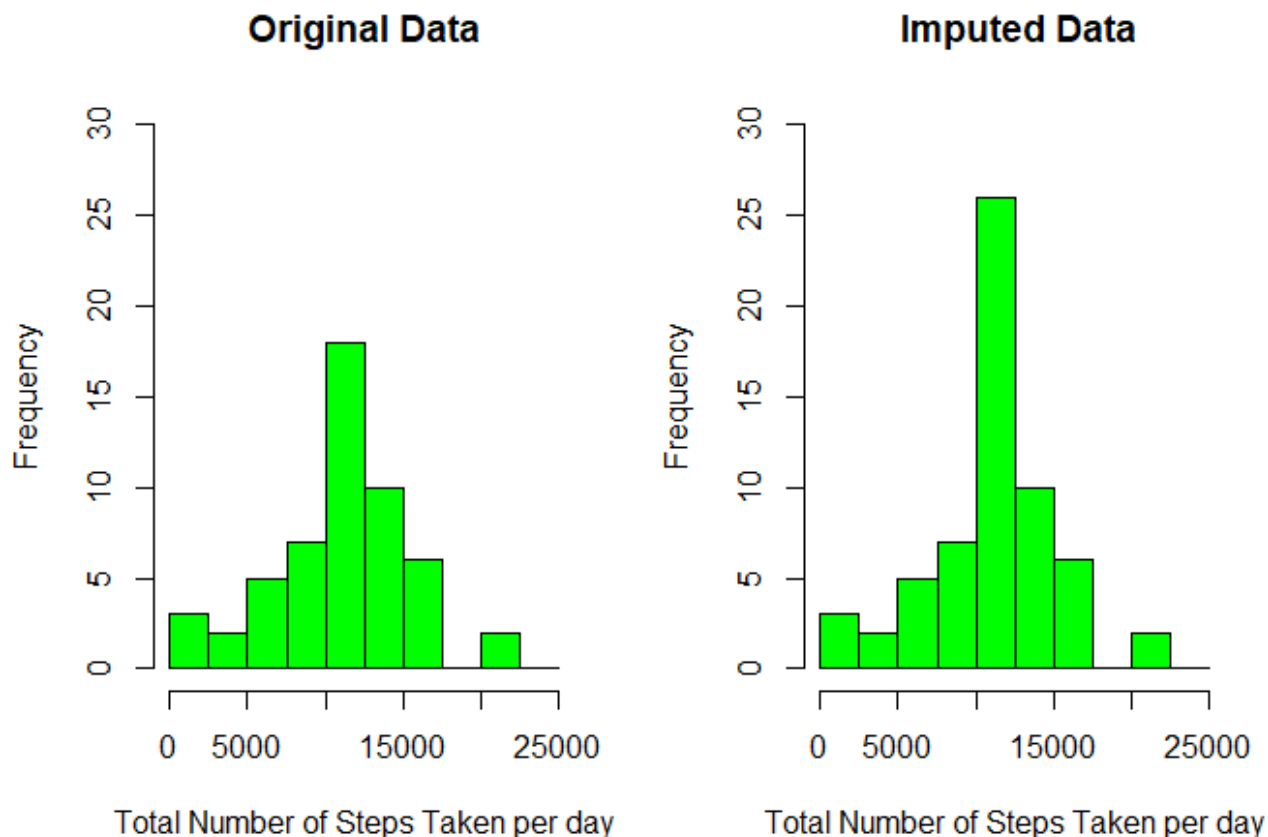
```
#Two plots together
```

```
par(mfrow=c(1, 2))
```

```
hist(mydata1$Total_Steps, ylim = c(0,30), xlab = "Total Number of Steps Taken per day"
```

```
hist(mydata4$Total_Steps, ylim = c(0,30), xlab = "Total Number of Steps Taken per day"
```





Conclusion: The mode of the imputed data is larger than that of original data.

5.Are there differences in activity patterns between weekdays and weekends?

5.1.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.

```
library(lubridate)
```

```
##
```

```
## Attaching package: 'lubridate'
```

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



```
##
##      date, intersect, setdiff, union
```

```
x=mydata3impute$date
w=wday(x, label = TRUE)
w=as.character(w)
WeekendStatus<-c()
l=length(w)
l
```

```
## [1] 17568
```

```
for(i in 1:l) { ifelse(w[i] %in% c("Sat","Sun"), WeekendStatus[i]<-"Weekend", Weekend
```

```
mydata5<- as.data.frame(cbind(mydata3impute$steps, mydata3impute$date,mydata3impute$i
names(mydata5)[names(mydata5) == "V1"] <- "Steps"
names(mydata5)[names(mydata5) == "V2"] <- "Date"
names(mydata5)[names(mydata5) == "V3"] <- "Interval"
names(mydata5)[names(mydata5) == "w"] <- "Day"
```

```
head(mydata5)
```

```
##           Steps      Date Interval Day WeekendStatus
## 1 37.3825995807128 2012-10-01         0 Mon      weekday
## 2 37.3825995807128 2012-10-01         5 Mon      weekday
## 3 37.3825995807128 2012-10-01        10 Mon      weekday
## 4 37.3825995807128 2012-10-01        15 Mon      weekday
## 5 37.3825995807128 2012-10-01        20 Mon      weekday
## 6 37.3825995807128 2012-10-01        25 Mon      weekday
```

```
#write.csv(mydata5, "mydata5.csv")
```

5.2. 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 (y-axis). See the README file in the GitHub repository to see an example of what this plot should look like using simulated data.

```
mydata6<- mydata5 %>%
  group_by(Interval, WeekendStatus) %>%
  summarise(Mean_Steps=mean(as.numeric(Steps)) )
```

```
## `summarise()` regrouping output by 'Interval' (override with `.groups` argument)
```

```
dim(mydata6)
```

```
## [1] 576 3
```

```
head(mydata6)
```

```
## # A tibble: 6 x 3
## # Groups:   Interval [3]
##   Interval WeekendStatus Mean_Steps
##   <chr>      <chr>          <dbl>
## 1 0        weekday          7.01
## 2 0        Weekend          4.67
## 3 10       weekday          5.14
## 4 10       Weekend          4.67
## 5 100      weekday          5.36
## 6 100      Weekend          4.67
```

```
#write.csv(mydata6,"mydata6.csv")
```

```
##Plots:
```

```
library(ggplot2)
```

```
mydata7<-mydata6[which(mydata6$WeekendStatus=="weekday"),]
#str(mydata7)
```

```
x1<-ts(as.numeric(mydata7$Interval))
y1<-ts(as.numeric(mydata7$Mean_Steps))
p1=ggplot(mydata7, aes(x1, y1)) +
  geom_line(na.rm=TRUE) +
  xlab("5 minute interval") + ylab("Average number of Steps Taken") +
  ggtitle("WeekDay")
```

```
mydata8<-mydata6[which(mydata6$WeekendStatus=="Weekend"),]  
#str(mydata8)
```

```
x2<-ts(as.numeric(mydata8$Interval))  
y2<-ts(as.numeric(mydata8$Mean_Steps))  
p2=ggplot(mydata8, aes(x2, y2)) +  
  geom_line(na.rm=TRUE) +  
  xlab("5 minute interval") + ylab("Average number of Steps Taken") +  
  ggtitle("Weekend")
```

```
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 4.0.2
```

```
##
```

```
## Attaching package: 'gridExtra'
```

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

```
##
```

```
##      combine
```

```
grid.arrange(p1, p2, nrow = 1)
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to co
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to co
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to co
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to co
```



