PA1_template

Stella

10/18/2022

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

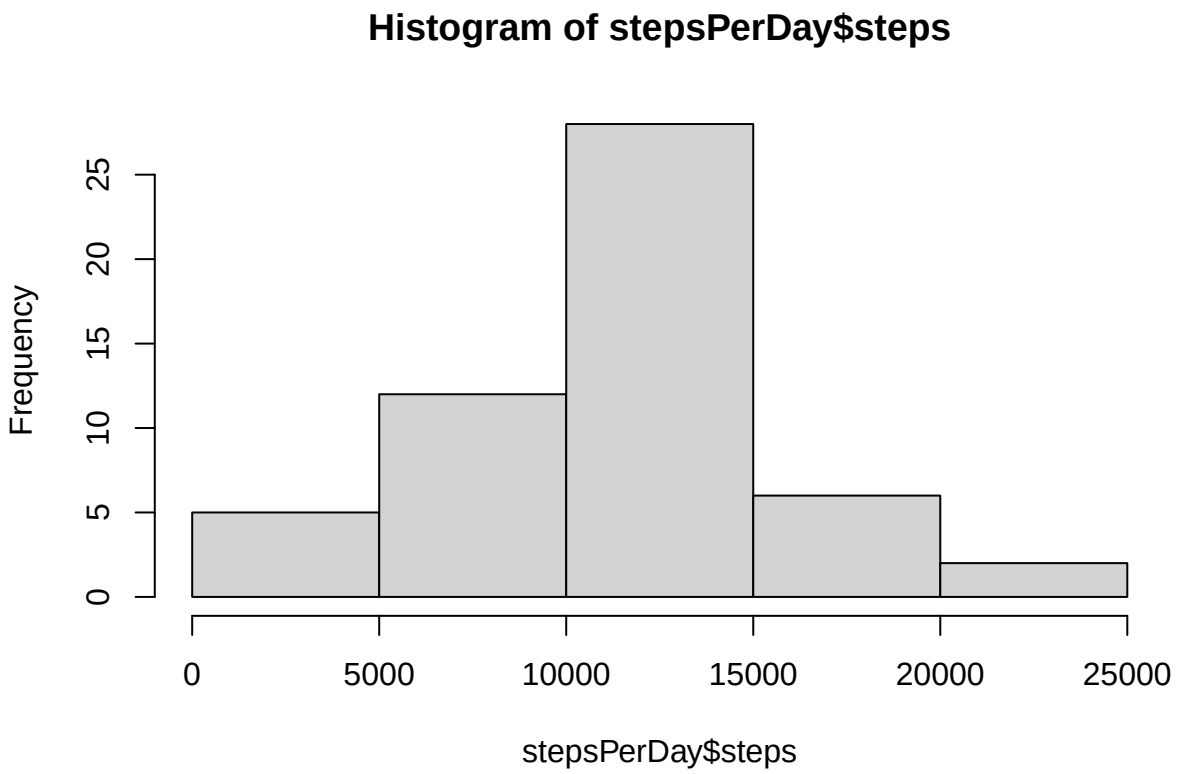
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
unzip("./activity.zip")
activityData <- read.csv("./activity.csv")
summary(activityData)
```

```
##      steps            date              interval
##  Min.   :  0.00   Length:17568       Min.   :   0.0
##  1st Qu.:  0.00   Class :character   1st Qu.: 588.8
##  Median :  0.00   Mode  :character   Median :1177.5
##  Mean   : 37.38                      Mean   :1177.5
##  3rd Qu.: 12.00                      3rd Qu.:1766.2
##  Max.   :806.00                      Max.   :2355.0
##  NA's   :2304
```

#part 2 total number of steps

```
stepsPerDay <- aggregate(steps ~ date, activityData, sum, na.rm=TRUE)
hist(stepsPerDay$steps)
```




```
png("plot 1.png")
```

part 2 report mean

```
meanStepsPerDay <- mean(stepsPerDay$steps)
meanStepsPerDay
```

```
## [1] 10766.19
```

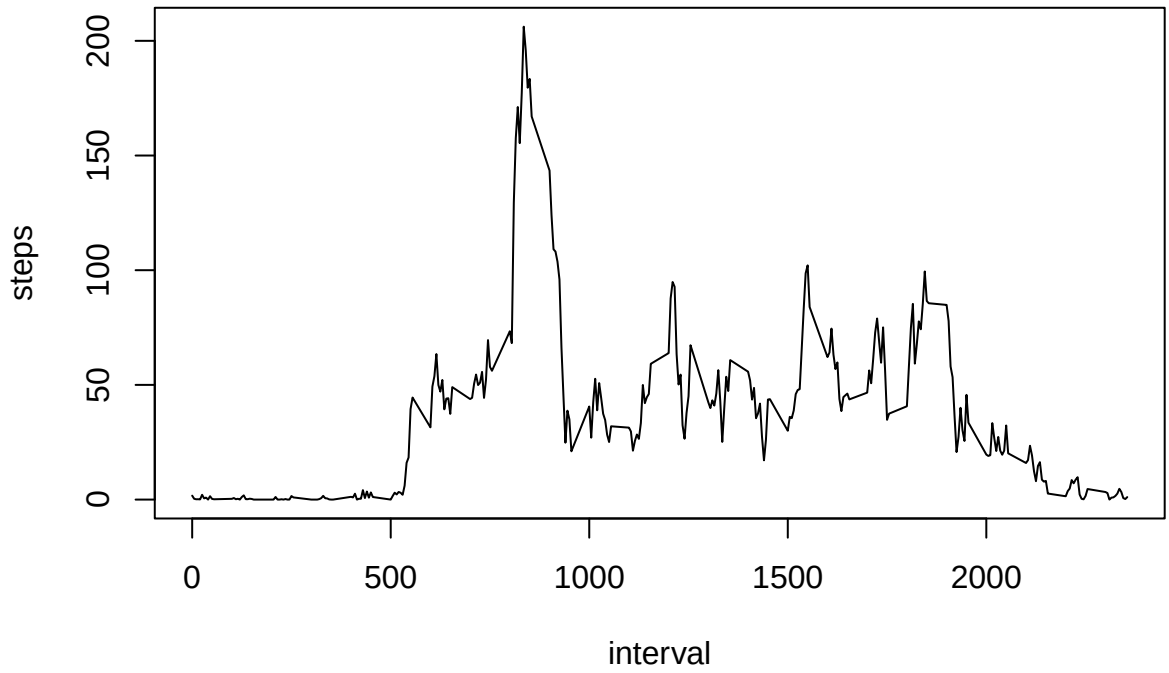
#part 2 report median

```
medianStepsPerDay <- median(stepsPerDay$steps)
medianStepsPerDay
```

```
## [1] 10765
```

#part 3 average daily activity pattern

```
stepsPerInterval<-aggregate(steps~interval, data=activityData, mean, na.rm=TRUE)
plot(steps~interval, data=stepsPerInterval, type="l")
```




```
png("plot 2.png")
```

#part 3 max steps

```
intervalWithMaxSteps <- stepsPerInterval[which.max(stepsPerInterval$steps),]$interval
intervalWithMaxSteps
```

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

#part 4 inputting missing values

```
totalValuesMissings <- sum(is.na(activityData$steps))
totalValuesMissings
```

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

#part 4 filling in missing values using mean

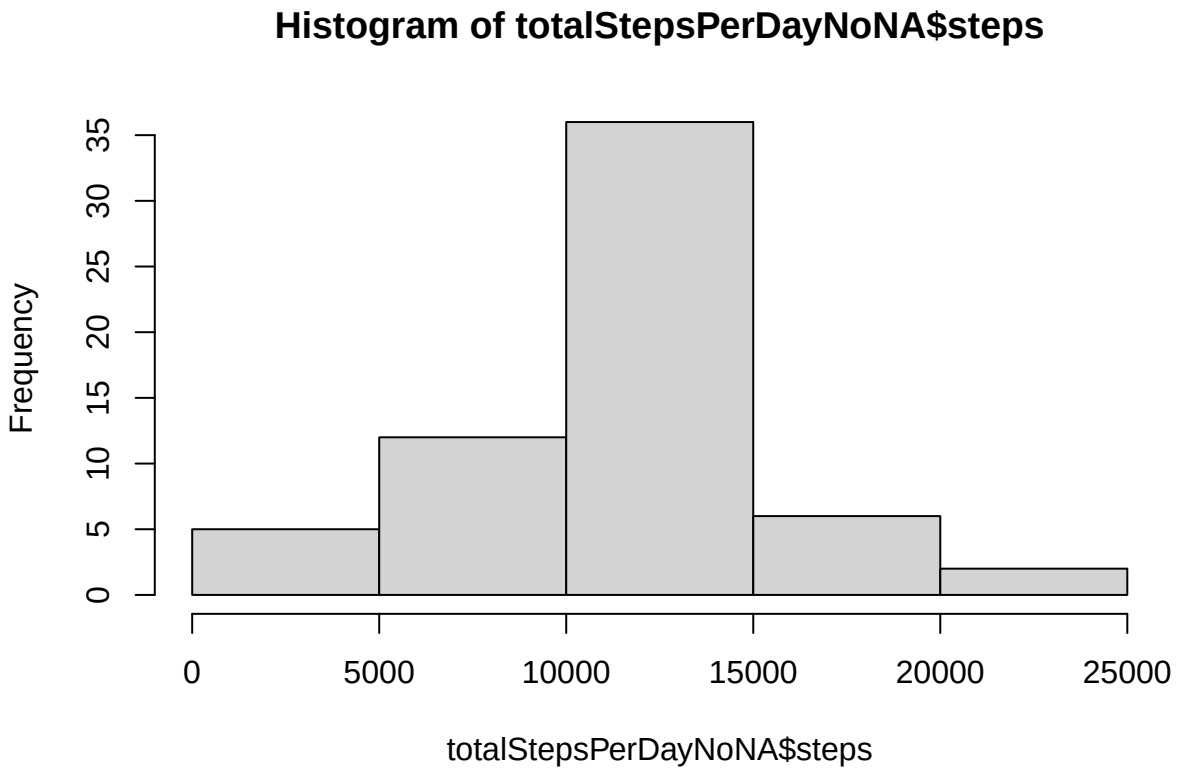
```
getMeanStepsPerInterval<-function(interval){
  stepsPerInterval[stepsPerInterval$interval==interval,]$steps
}
```

#part 4 new data set with filled in data

```
activityDataNoNA<-activityData
for(i in 1:nrow(activityDataNoNA)){
  if(is.na(activityDataNoNA[i,]$steps)){
    activityDataNoNA[i,]$steps <- getMeanStepsPerInterval(activityDataNoNA[i,]$interval)
  }
}
```

#part 4 histogram

```
  totalStepsPerDayNoNA <- aggregate(steps ~ date, data=activityDataNoNA, sum)
  hist(totalStepsPerDayNoNA$steps)
```




```
png("plot 3.png")
```

#part 4 report mean and median

```
meanStepsPerDayNoNA <- mean(totalStepsPerDayNoNA$steps)
medianStepsPerDayNoNA <- median(totalStepsPerDayNoNA$steps)
```

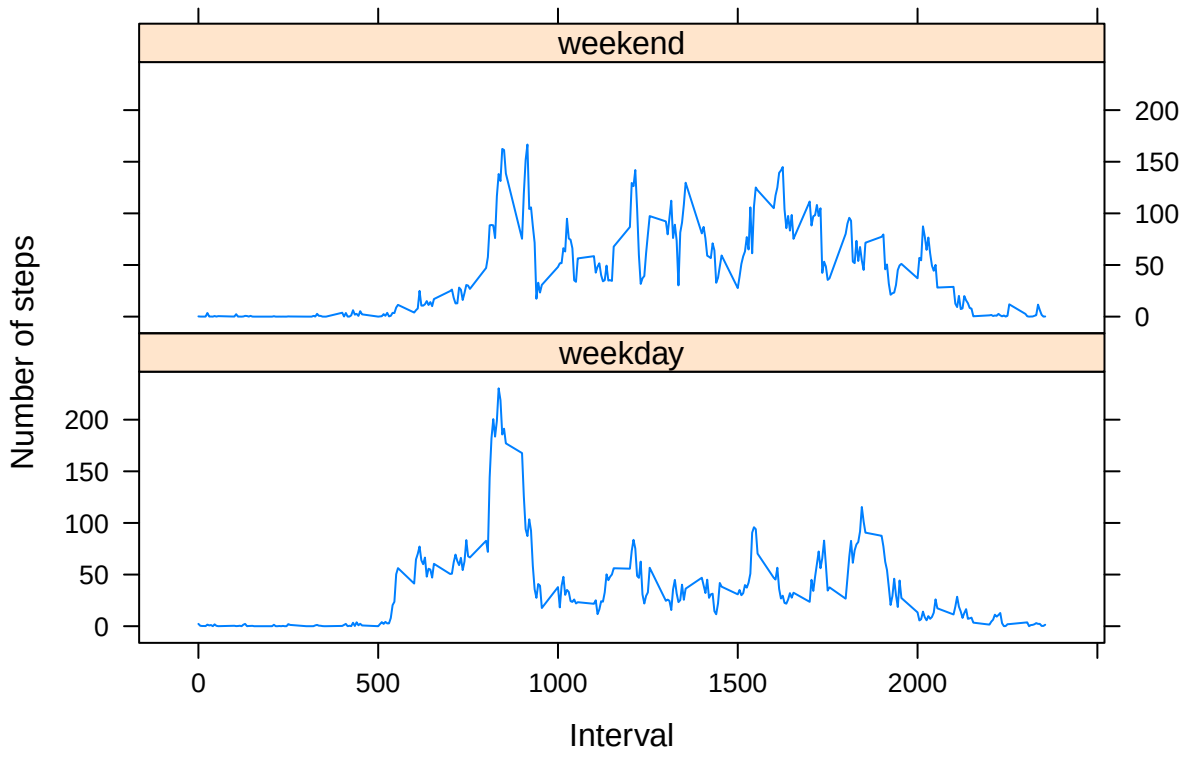
found that the mean does not change after the change, but the median does by a small value.

#part 5 distinguish weekend and weekday

```
activityDataNoNA$date <- as.Date(strptime(activityDataNoNA$date, format="%Y-%m-%d"))
activityDataNoNA$day <- weekdays(activityDataNoNA$date)
for (i in 1:nrow(activityDataNoNA)) {
  if (activityDataNoNA[i,]$day %in% c("Saturday","Sunday")) {
    activityDataNoNA[i,]$day<-"weekend"
  }
  else{
    activityDataNoNA[i,]$day<-"weekday"
  }
}
stepsByDay <- aggregate(activityDataNoNA$steps ~ activityDataNoNA$interval + activityDataNoNA$day, ac
```

#part 5 panel plot with time series

```
names(stepsByDay) <- c("interval", "day", "steps")
library(lattice)
xyplot(steps ~ interval | day, stepsByDay, type = "l", layout = c(1, 2),
       xlab = "Interval", ylab = "Number of steps")
```




```
png("plot 4.png")
```

Including Plots

You can also embed plots, for example:

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.