Create a folder called assignment 1:

```
if(!file.exists("./assignment1")){dir.create("./assignment1")}
```

Download and unzip file:

```
fileurl <- "https://d396qusza40orc.cloudfront.net/repdata%2Fdata%2Factivity.zip"
download.file(fileurl, destfile = "./assignment1/assignment.zip")
unzip(zipfile = "./assignment1/assignment.zip", exdir = "./assignment1")
```

Check the class of each variables

```
library(readr)
```

```
## Warning: package 'readr' was built under R version 4.1.3
```

```
setwd("D:/Statistics/R/R data/Assignment1/assignment1")
activity <- read.csv("activity.csv")
lapply(activity, class)
```

```
## $steps
## [1] "integer"
##
## $date
## [1] "character"
##
## $interval
## [1] "integer"
```

Convert variable character date into date/time format

```
activity$date <- as.Date(activity$date)
lapply(activity, class)
```

```
## $steps
## [1] "integer"
##
## $date
## [1] "Date"
##
## $interval
## [1] "integer"
```

Figure out weekdays corresspond to time and add to dataframe

```
activity$day <- weekdays(activity$date)
lapply(activity, class)
```

```
## $steps
## [1] "integer"
##
## $date
## [1] "Date"
##
## $interval
## [1] "integer"
##
## $day
## [1] "character"
```

```
summary(activity)
```

```
##      steps             date               interval          day
##  Min.   :  0.00   Min.   :2012-10-01   Min.   :   0.0   Length:17568
##  1st Qu.:  0.00   1st Qu.:2012-10-16   1st Qu.: 588.8   Class :character
##  Median :  0.00   Median :2012-10-31   Median :1177.5   Mode  :character
##  Mean   : 37.38   Mean   :2012-10-31   Mean   :1177.5
##  3rd Qu.: 12.00   3rd Qu.:2012-11-15   3rd Qu.:1766.2
##  Max.   :806.00   Max.   :2012-11-30   Max.   :2355.0
##  NA's   :2304
```

1. Histogram of the total number of steps taken each day + mean/median

Create a dataframe that has a total of steps by day - changing column names

```
activityTotalSteps <- with(activity, aggregate(steps, by = list(date), sum, na.rm = TRUE))
names(activityTotalSteps) <- c("Date", "Steps")
class(activityTotalSteps)
```

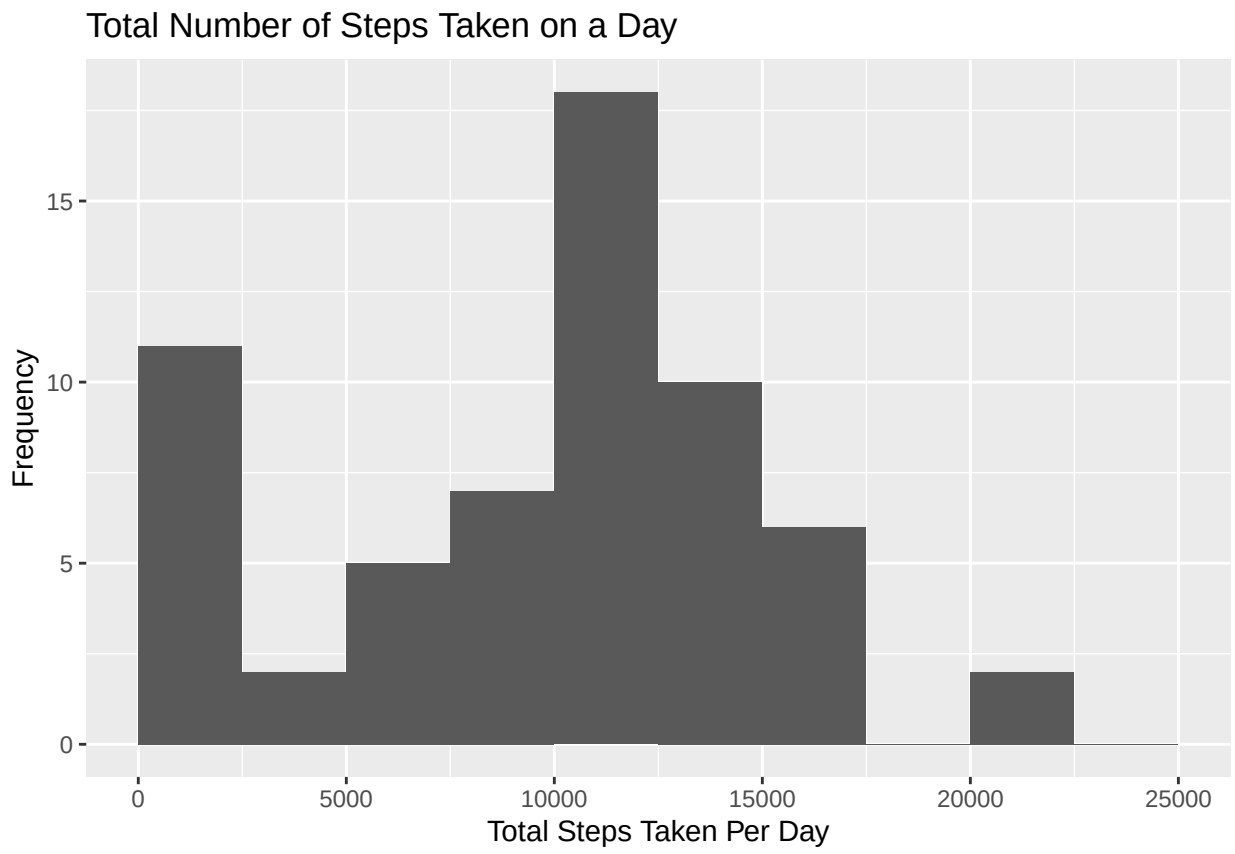
```
## [1] "data.frame"
```

Drawing the histogram

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

```
ggplot(activityTotalSteps, aes(x = Steps)) + geom_histogram(breaks = seq(0, 25000, by = 2500)) +  xlab(
```




```
dev.copy(device = png, width = 480, height = 480, file = "Plot1.png")
```

```
## png
##   3
```

```
dev.off()
```

```
## pdf
##   2
```

Calculate the mean

```
mean(activityTotalSteps$Steps)
```

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

Calculate the median

```
median(activityTotalSteps$Steps)
```

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

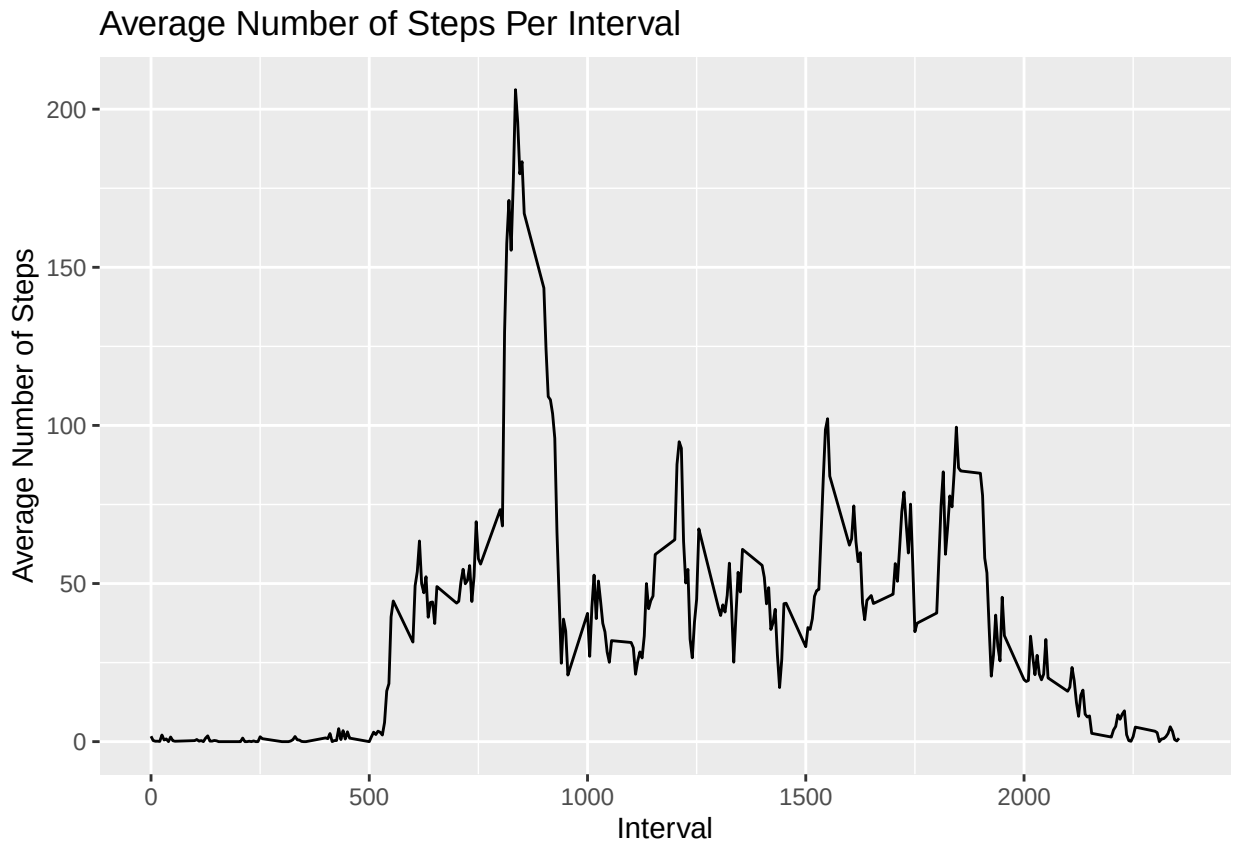
2. What is the average daily activity pattern?

Extracting data (Interval, steps) from activity and calculate the mean value

```
averageDailyActivity <- aggregate(activity$steps, by = list(activity$interval), FUN = mean, na.rm = TRU
names(averageDailyActivity) <- c("Interval", "Mean")
```

Drawing graph

```
ggplot(averageDailyActivity, mapping = aes(Interval, Mean)) + geom_line() + xlab("Interval") + ylab("Av
```




```
dev.copy(device = png, width = 480, height = 480, file = "Plot2.png")
```

```
## png
##   3
```

```
dev.off()
```

```
## pdf
##   2
```

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

```
averageDailyActivity[which.max(averageDailyActivity$Mean), ]
```

```
##     Interval     Mean
## 104      835 206.1698
```

3. Imputing Missing Values

3,1 Calculate and report the total number of missing values in the dataset (i.e. the total number of rows with NAs)

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

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

3.2 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.

Create value which matched the mean of averageDailyActivity with activity dataframe

```
imputedSteps <- averageDailyActivity$Mean[match(activity$interval, averageDailyActivity$Interval)]
```

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

```
activityImputed <- transform(activity, steps = ifelse(is.na(activity$steps), yes = imputedSteps, no = a
```

Testing if there is any missing value

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

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

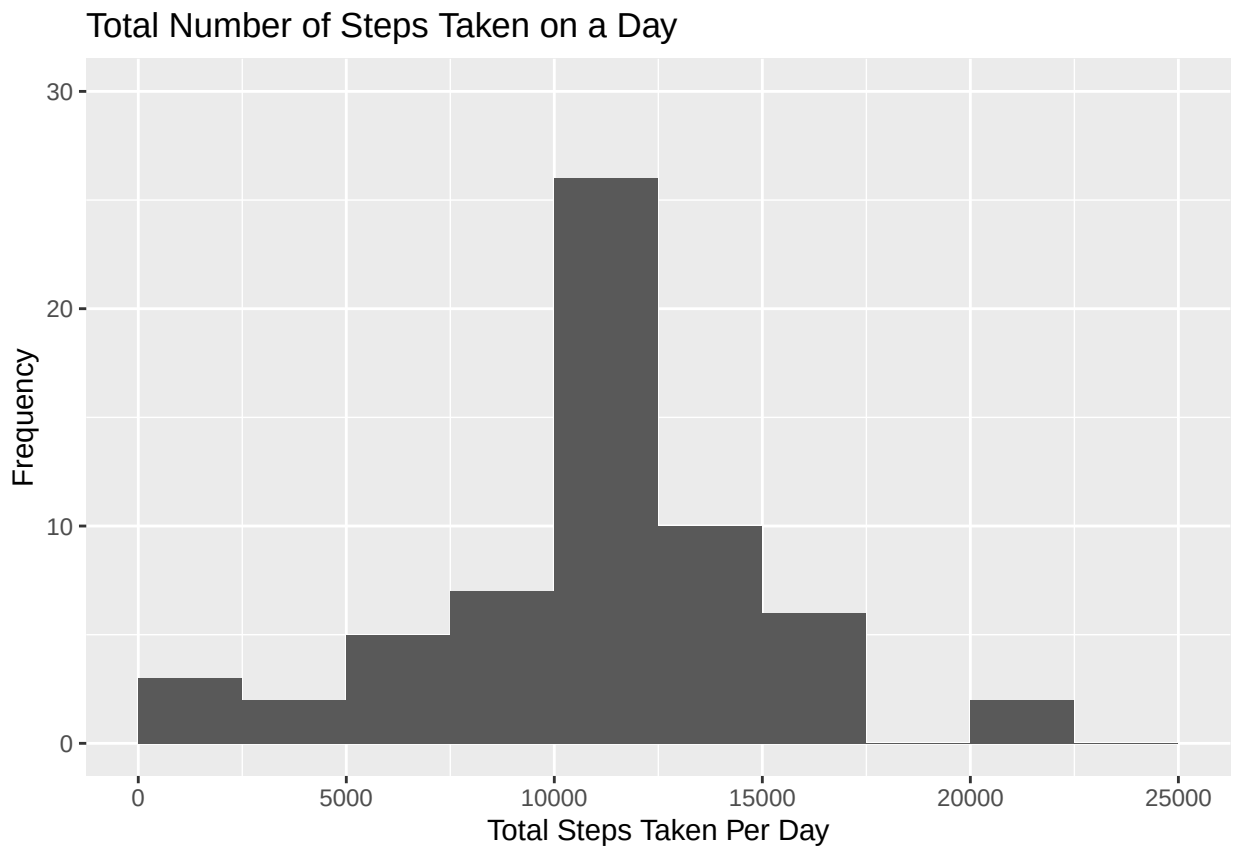
3.4 Make a histogram of the total number of steps taken each day and Calculate and report the mean and median total number of steps taken per day. Do these values differ from the estimates from the first part of the assignment? What is the impact of imputing missing data on the estimates of the total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

Extract (steps, date) from dataframe and calculate the sum value of steps

```
totalActivityImputed <- aggregate(steps ~ date, activityImputed, sum)
names(totalActivityImputed) <- c("date", "dailySteps")
```

Drawing the graph

```
ggplot(totalActivityImputed, aes(dailySteps)) + geom_histogram(breaks = seq(0, 25000, by = 2500)) +   y
```




```
dev.copy(device = png, width = 480, height = 480, file = "Plot3.png")
```

```
## png
##   3
```

```
dev.off()
```

```
## pdf
##   2
```

Calculate the mean

```
mean(totalActivityImputed$dailySteps)
```

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

Calculate the median

```
median(totalActivityImputed$dailySteps)
```

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

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

Computing the weekdays from the date attribute

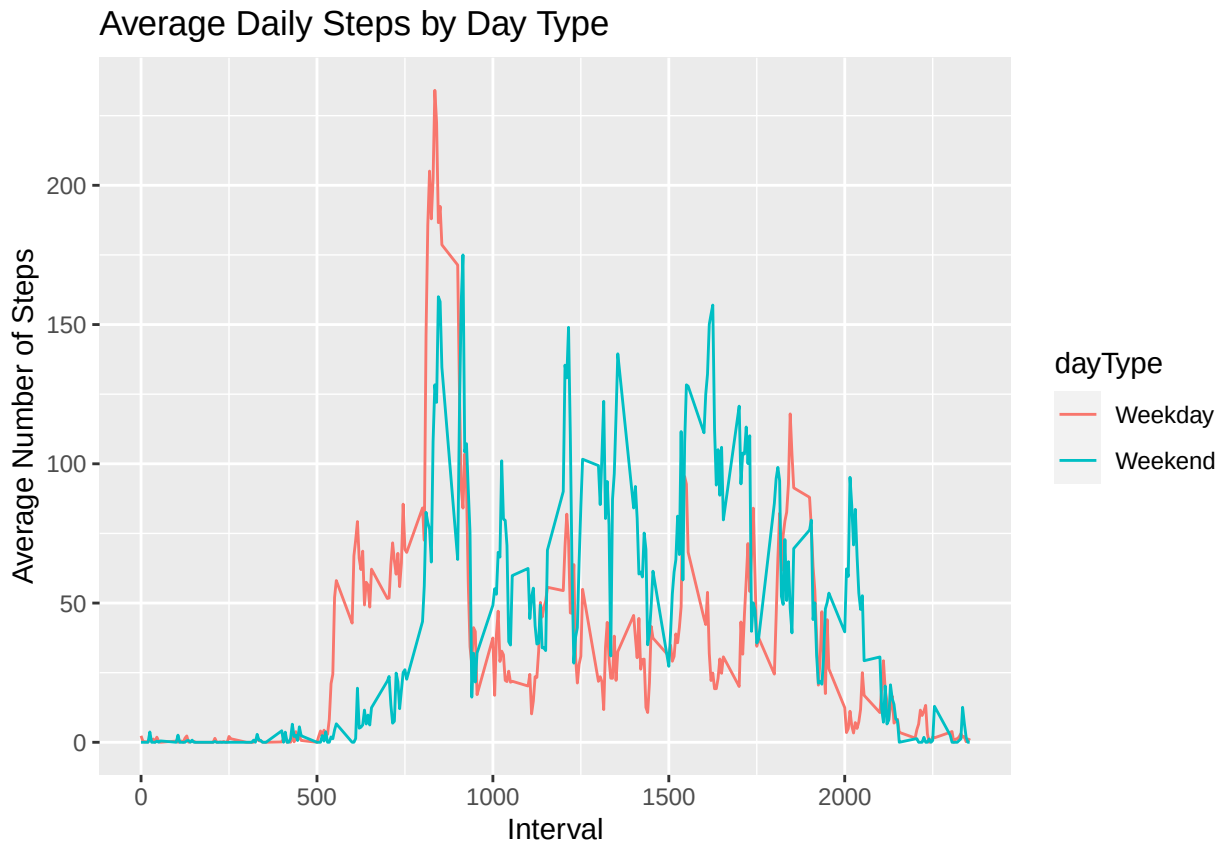
```
activity$date <- as.Date(strptime(activity$date, format="%Y-%m-%d"))
activity$dayType <- sapply(activity$date, function(x) {
  if(weekdays(x) == "Saturday" | weekdays(x) == "Sunday")
  {y <- "Weekend"}
  else {y <- "Weekday"}
  y
})
```

Creating a dataframe with interval, daytype and steps

```
activityByDay <-  aggregate(steps ~ interval + dayType, activity, mean, na.rm = TRUE)
```

Drawing the plot

```
ggplot(activityByDay, aes(x = interval , y = steps, color = dayType)) + geom_line() + ggtitle("Average
```




```
dev.copy(device = png, width = 480, height = 480, file = "Plot4.png")
```

```
## png
##   3
```

```
dev.off()
```

```
## pdf
##   2
```