Reproducible Research:

Peer Assessment #1

Loading and preprocessing the data

Conditionally (if not already present) download, unzip, and read-in data.

```
library(dplyr, warn.conflicts=FALSE)
dataURL <- paste0("https://d396qusza40orc.cloudfront.",
                  "net/repdata%2Fdata%2Factivity.zip")
targFileName <- "data/repdata\ data\ activity.zip"
dataFileName <- "data/activity.csv"

if(!file.exists(dataFileName)) {            ## If not present, download file
  download.file(dataURL,                    ## from web and decompress
                destfile = targFileName,
                method = "curl")
  unzip(targFileName)
}

if(!"actDat" %in% ls()){                    ## If not already loaded,
  actDat <- read.csv(dataFileName)          ## read-in csv data file.
  actDat$date <- as.Date(actDat$date,       ## Change date to Posix format
                         format="%Y-%m-%d")
}
rm(dataURL, targFileName, dataFileName)     ## Cleanup temp data objects
```

What is mean total number of steps taken per day?

Make a histogram of the total number of steps taken each day:

```
plot(                                       ## H-plot the total steps/day
  na.omit(                                  ## Omits NaN and NA values
    summarize(
      group_by(
        actDat, date),
          sum(steps, na.rm=TRUE))),
  type="h",
  ylab="Total Steps",
  xlab="Date")


dev.copy(png, file = "meanStepsPlot.png")
```

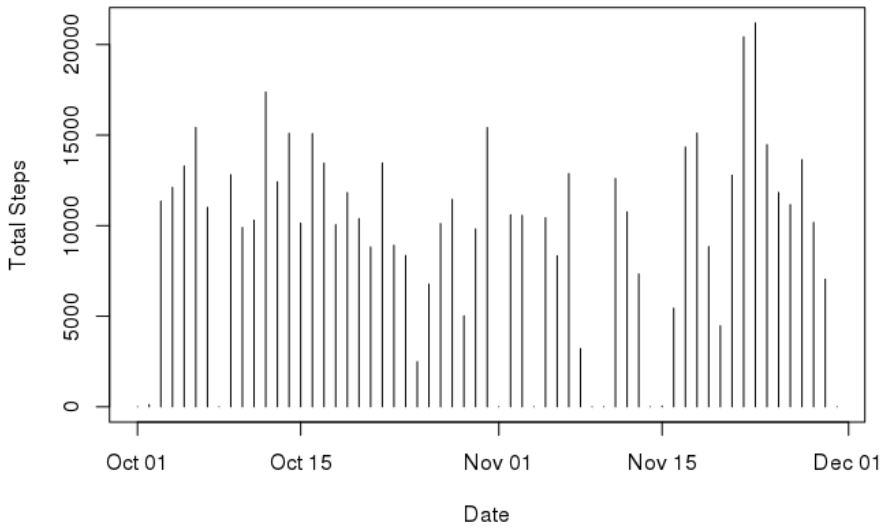

Figure 1: plot of chunk meanStepsPlot

```
## png
##   3

dev.off()

## pdf
##   2
```

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

```
print(
  paste(
    "Average Steps Per Day",
    mean(                                   ## Computes the mean of the sums
      na.omit(                              ## for each day.  Omits NaN and
        summarize(                          ## NA values.
          group_by(
            actDat,
            date),
          sum(steps,
              na.rm=TRUE)))$sum)))
```

```
## [1] "Average Steps Per Day 9354.22950819672"
```

```
print(
  paste(
    "Median Steps Per Day",
  median(                                    ## Computes the mid-point of the
    na.omit(                                 ## sums for each day.  Omits NaN
      summarize(                             ## and NA values.
        group_by(
          actDat,
          date),
        sum(
          steps,
          na.rm=TRUE)))$sum)))
```

```
## [1] "Median Steps Per Day 10395"
```

What is the average daily activity pattern?

Make 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 days (y-axis)

```
plot(                                      ## L-plot the average steps/interval
  na.omit(                                 ## Omits NaN and NA values
    summarize(
      group_by(
        actDat, interval),
          mean(steps, na.rm=TRUE))),
  type="l",
  ylab="Total Steps",
  xlab="5-Minute Interval")
```

```
dev.copy(png, file = "plotStepsInterval.png")
```

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

```
dev.off()
```

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

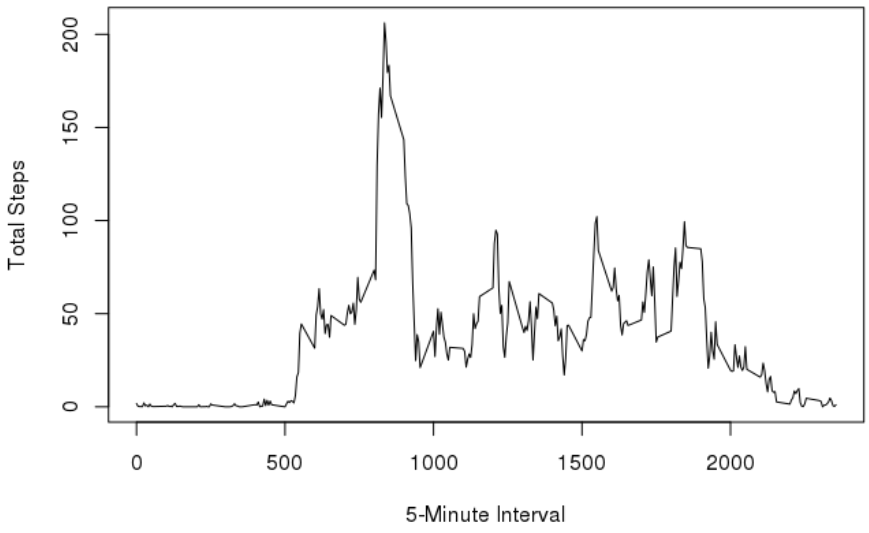

Figure 2: plot of chunk plotStepsInterval

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

```
totals <- summarize(
            group_by(actDat, interval),
            sum(steps, na.rm=TRUE))
maxStep <- totals[totals$sum == max(totals$sum), c(1,2)]
hour <- as.integer(maxStep$interval/60)
minute <- ((maxStep$interval/60 - hour)/100) *60
print(paste0("Maximum activity is ",
            maxStep$sum,
            " steps at ",
            hour,
            ":",
            minute))
```

```
## [1] "Maximum activity is 10927 steps at 13:0.55"
```

```
rm(totals, maxStep, hour, minute)           ## Explicit cleanup unused vars.
```

Input missing values

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

```
# note that there is no such thing as "imputting"
sum(as.numeric(is.na(actDat[,1])))
```

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

Devise a strategy for filling in all of the missing values in the datase. Create a new dataset that is equal to the original dataset but with the missing data filled in.

It is easily determined that NA values are clustered for entire days: in other words, many days have no observations for steps.

```
for(date in unique(actDat$date)){
  print(
    paste0(
      mean(actDat[actDat$date==date, 1]),
           " ",
           date))
  }
```

```
## code chunk demonstrates that NA
## values for step observations are
## covering entire days by not using
## na.omit with the mean function.
## Computes the step mean for each
## day of collections. Displays mean
## and POSIX unformatted day.
```

```
## [1] "NA 15614"
## [1] "0.4375 15615"
## [1] "39.4166666666667 15616"
## [1] "42.0694444444444 15617"
## [1] "46.1597222222222 15618"
## [1] "53.5416666666667 15619"
## [1] "38.2465277777778 15620"
## [1] "NA 15621"
## [1] "44.4826388888889 15622"
## [1] "34.375 15623"
## [1] "35.7777777777778 15624"
## [1] "60.3541666666667 15625"
## [1] "43.1458333333333 15626"
## [1] "52.4236111111111 15627"
## [1] "35.2048611111111 15628"
## [1] "52.375 15629"
## [1] "46.7083333333333 15630"
## [1] "34.9166666666667 15631"
## [1] "41.0729166666667 15632"
## [1] "36.09375 15633"
```

```
## [1] "30.6284722222222 15634"
## [1] "46.7361111111111 15635"
## [1] "30.9652777777778 15636"
## [1] "29.0104166666667 15637"
## [1] "8.65277777777778 15638"
## [1] "23.5347222222222 15639"
## [1] "35.1354166666667 15640"
## [1] "39.7847222222222 15641"
## [1] "17.4236111111111 15642"
## [1] "34.09375 15643"
## [1] "53.5208333333333 15644"
## [1] "NA 15645"
## [1] "36.8055555555556 15646"
## [1] "36.7048611111111 15647"
## [1] "NA 15648"
## [1] "36.2465277777778 15649"
## [1] "28.9375 15650"
## [1] "44.7326388888889 15651"
## [1] "11.1770833333333 15652"
## [1] "NA 15653"
## [1] "NA 15654"
## [1] "43.7777777777778 15655"
## [1] "37.3784722222222 15656"
## [1] "25.4722222222222 15657"
## [1] "NA 15658"
## [1] "0.142361111111111 15659"
## [1] "18.8923611111111 15660"
## [1] "49.7881944444444 15661"
## [1] "52.4652777777778 15662"
## [1] "30.6979166666667 15663"
## [1] "15.5277777777778 15664"
## [1] "44.3993055555556 15665"
## [1] "70.9270833333333 15666"
## [1] "73.5902777777778 15667"
## [1] "50.2708333333333 15668"
## [1] "41.0902777777778 15669"
## [1] "38.7569444444444 15670"
## [1] "47.3819444444444 15671"
## [1] "35.3576388888889 15672"
## [1] "24.46875 15673"
## [1] "NA 15674"
```

A valid solution would be to discard records with no valid entries for the day. However, discarding the bad reading dates would not accomplish the instructions from the assignment:

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

The best compromise solution would be to replace observation step NA values with 0 so as not to change the overall average:

```
naReplace <-function(steps){                ## Substitutes daily mean for NA
  if(is.na(steps)){                         ## values. Returns a vector of
    return(0)                               ## steps.
    }
  else{
    return(steps)
    }
  }

newActDat <-                                ## Create a new data frame with the
  data.frame(steps=as.numeric(              ## same sample labels.  Replace all
                      lapply(               ## NA values with the mean of all
                        actDat$steps,       ## days of data collections.
                        naReplace)),
             date=actDat$date,
             interval=actDat$interval)
dim(newActDat)                              ## show the size of the new data
```

```
## [1] 17568     3
```

```
head(newActDat)                             ## Show a sample of the new data
```

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

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 *(sic)* missing data on the estimates of the total daily number of steps?

Histogram below:

```
plot(
  summarize(
    group_by(
        newActDat, date),
          sum(steps)),
  type="h",
  ylab="Total Steps",
  xlab="Date")
```

```
## H-plot the total steps/day.
## Omits NaN and NA values, same as
## above.
```

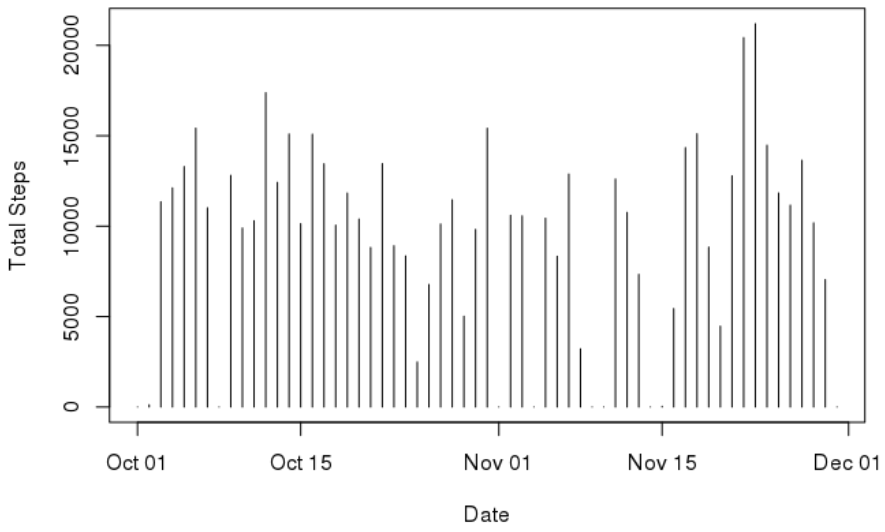

Figure 3: plot of chunk corrDatHist

```
dev.copy(png, file = "corrDatHist.png")
```

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

```
dev.off()
```

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

Mean and Median daily values

```
print(
  paste(
    "Average Steps Per Day",
    mean(                                   ## Computes the mean of the sums
      summarize(                            ## for each day.  Omits NaN and
        group_by(                           ## NA values.
            newActDat,
            date),
          sum(steps))$sum)))
```

```
## [1] "Average Steps Per Day 9354.22950819672"
```

```
print(
  paste(
    "Median Steps Per Day",
  median(                                 ## Computes the mid-point of the
    summarize(                            ## sums for each day.  Omits NaN
      group_by(                           ## and NA values.
          newActDat,
          date),
        sum(steps))$sum)))
```

```
## [1] "Median Steps Per Day 10395"
```

Since the NA data were replaced with zero, the daily sums for the affected days have not changed. The most recent histogram shows the same values as that of the original data, since the original plot omitted NA values

Are there differences in activity patterns between weekdays and weekends?

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.

```
newActDat <-
  mutate(newActDat,                       ## Adds new variable for weekday
         weekday = !grepl("S[a-z]*day",   ## as a boolean. Uses regexp to
                          as.character(   ## detect day names beginning with
                            lapply(       ## "S" and ending in "day."
                              newActDat$date,
                              weekdays))))
dim(newActDat)
```

```
## [1] 17568     4
```

```
head(newActDat)
```

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

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).

```
par(mfrow = c(2,1),                              ## Sets up multipanel plots and
    pin = c(10, 10),                             ## other aesthetics.
    mar = c(.5,2,0,0),
    oma = c(1,1,0,0),
    lab = c(10, 6, 5))
plot(                                            ## L-plot the average steps/interval
  summarize(                                     ## for weekdays. Omits NaN and NA
    group_by(                                    ## values.
      newActDat[newActDat$weekday==TRUE,],
      interval),
        mean(steps)),
  type = "l",
  xaxt = "n",                                    ## Omits x axis for the top graph
  ylim = c(0,210),
  xlim = c(0,2355),                              ## but sets x axis interval to match
  cex.axis = .7)
text(1150, 210, "Weekday",                       ## Labels plot data
     pos = 1, cex = .8)
par(mar = c(2,2,0,0))                            ## Modifies the margin for plot 2

plot(                                            ## L-plot the average steps/interval
  summarize(                                     ## for weekend data. Omits NaN and
    group_by(                                    ## NA values.
      newActDat[newActDat$weekday==FALSE,],
      interval),
        mean(steps)),
  type="l",
  ylim = c(0,210),
  xlim = c(0,2355),
  cex.axis = .7 )
text(1150, 210, "Weekend",                        ## Labels the plot
```

```
     pos = 1, cex = .8)
mtext("Number of Steps",                    ## Labels the shared axes
      side = 2,
      outer = TRUE)
mtext("Interval",
      side = 1,
      outer = TRUE)
```

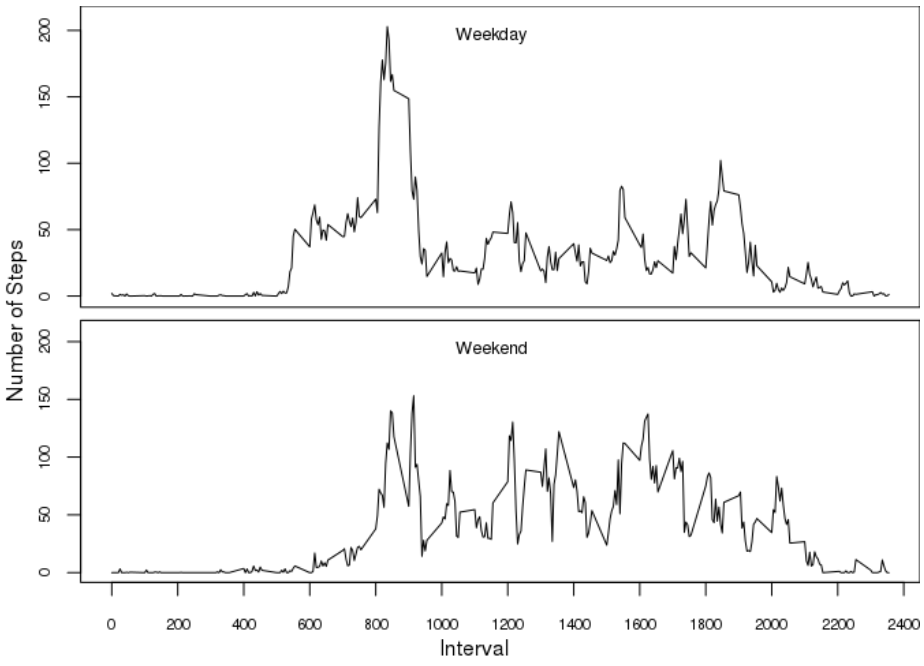

Figure 4: plot of chunk panelPlot

```
dev.copy(png, file = "panelPlot.png")

## png
##   3

dev.off()

## pdf
##   2

rm(newActDat)                               ## Cleanup temp data frame
```