Project_1

Byron Jones

January 19, 2017

Project dataset contains the following variables:

***steps*: number of steps taking in a 5-minute interval (missing values are coded as NA)**

***date*: the date on which the measurement was taken in YYYY-MM-DD format**

***interval*: identifier for the 5-minute interval in which measurement was taken**

The dataset is stored in a comma-separated-value (CSV) file and there are a total of 17,568 observations in this dataset

Required for the assignment:

1.Code for reading in the dataset and/or processing the data

2.Histogram of the total number of steps taken each day

3.Mean and median number of steps taken each day

4.Time series plot of the average number of steps taken

5.The 5-minute interval that, on average, contains the maximum number of steps

6.Code to describe and show a strategy for imputing missing data

7.Histogram of the total number of steps taken each day after missing values are imputed

8.Panel plot comparing the average number of steps taken per 5-minute interval across weekdays and weekends

9.All of the R code needed to reproduce the results (numbers, plots, etc.) in the report

Set working directory, read in the dataset and look at first few rows

```
# set working directory
setwd("//Phchbs-s3159/jonesby1$/data/Byron/R/Reproducible Research")

# read in the dataset and look at first few rows
activity<-read.csv("activity.csv",head=TRUE)
print(head(activity))
```

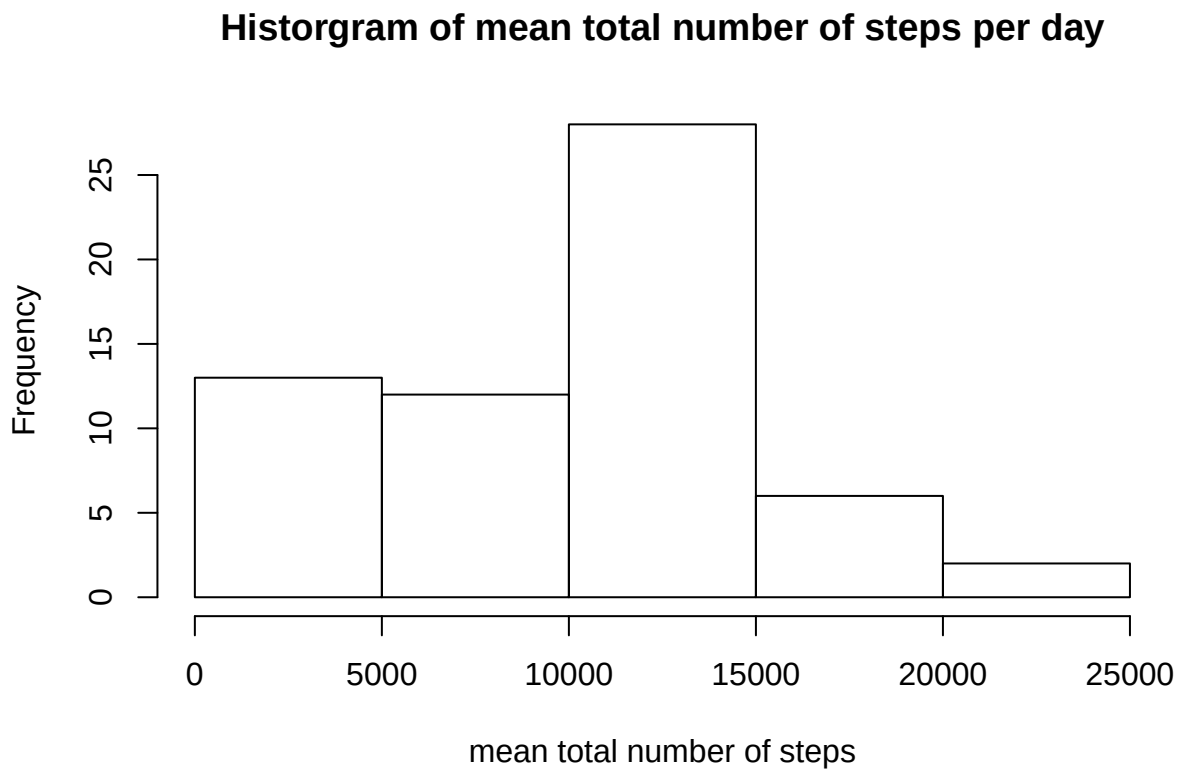
```
##   steps       date interval
## 1    NA 2012-10-01        0
## 2    NA 2012-10-01        5
## 3    NA 2012-10-01       10
## 4    NA 2012-10-01       15
## 5    NA 2012-10-01       20
## 6    NA 2012-10-01       25
```

Identify days of the week and add this new variable to existing activity list

```
week.days<-weekdays(as.Date(unlist(activity$date)))

activity[["day"]]<-week.days
names(activity)<-c("steps", "date","interval", "day")
```

Calculate mean total number of steps taken each day and mean total number of steps in each 5-minute interval using the aggregate function

```
day.mean<-as.numeric(unlist(aggregate(activity$steps, by=list(activity$date), FUN=sum, na.rm=TRUE)[2]))
hist(day.mean,xlab="mean total number of steps",main="Historgram of mean total number of steps per day")
```



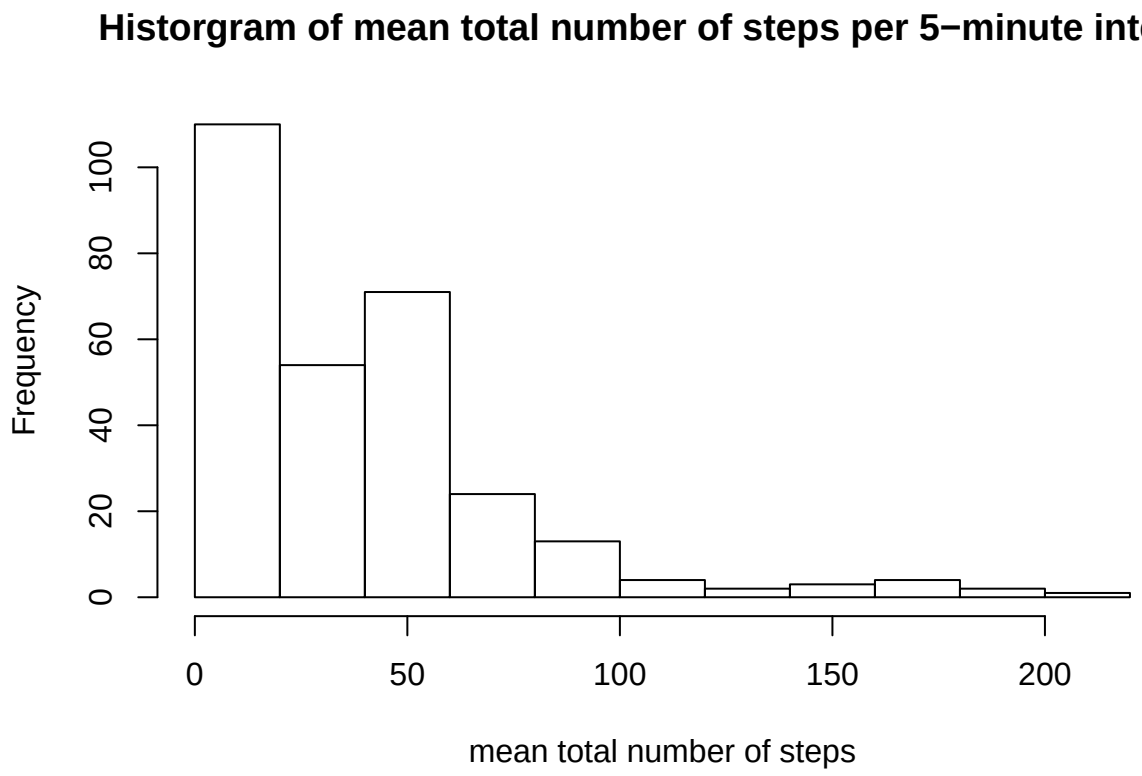

```
print(mean(day.mean))
```

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

```
print(median(day.mean))
```

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

```
## calculate mean for each 5-minute interval using the aggregate function
interval.mean<-as.numeric(unlist(aggregate(activity$steps, by=list(activity$interval), FUN=mean, na.rm='
hist(interval.mean,xlab="mean total number of steps",main="Historgram of mean total number of steps per
```




```
print(mean(interval.mean))
```

```
## [1] 37.3826
```

```
print(median(interval.mean))
```

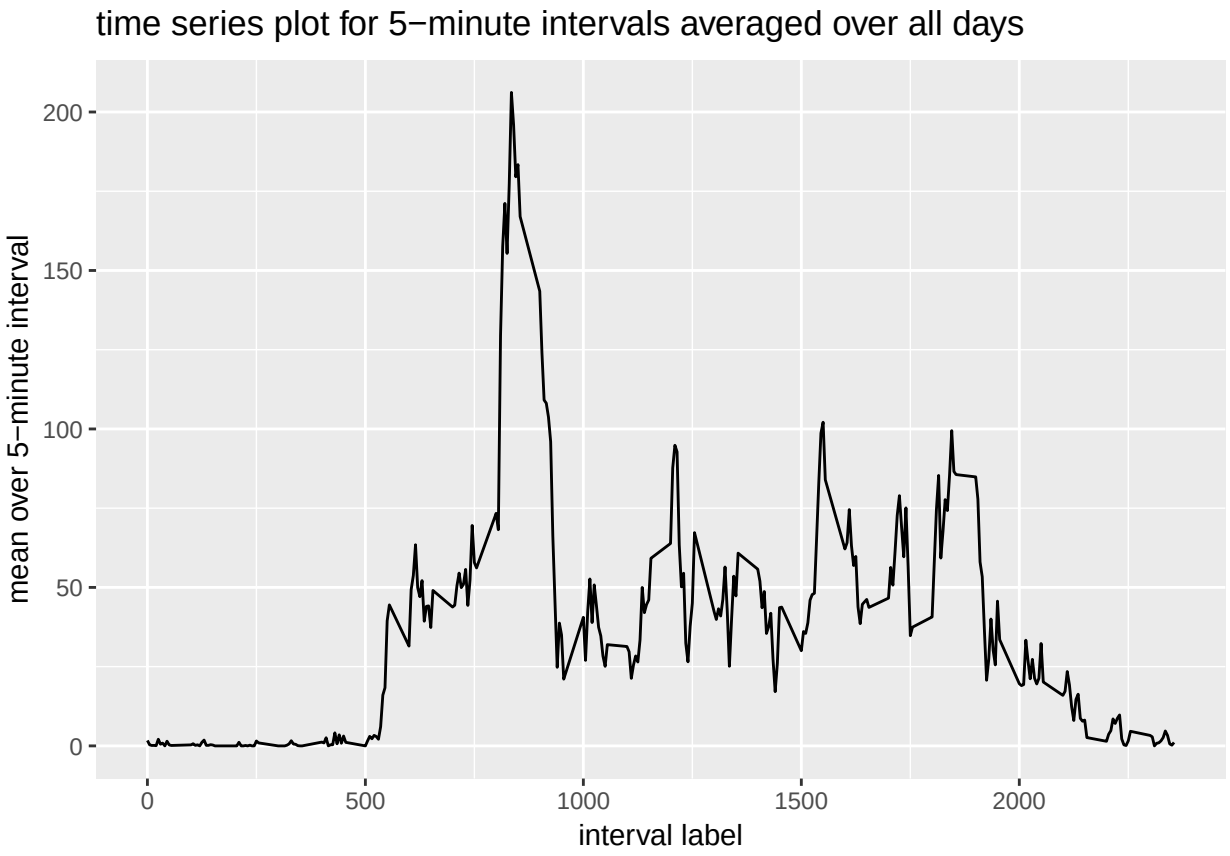
```
## [1] 34.11321
```

Time series plot

```
## plotting values for x-axis
interval.values<-unique(activity$interval)
## create a data frame for use in ggplot2
interval.df<-data.frame(cbind(interval.values,interval.mean))

## time series plot using ggplot2
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.2.5
```

```
ggplot(interval.df, aes(x=interval.values, y=interval.mean)) +
  geom_line() +
  ggtitle("time series plot for 5-minute intervals averaged over all days") +
  labs(x="interval label") +
  labs(y="mean over 5-minute interval")
```



Maximum mean number of days and the label of the interval that contains the maximum

```
## maximum mean number of steps
max.mean<-max(interval.mean)
print(max.mean)
```

```
## [1] 206.1698
```

```
## interval containing maximum mean
max.interval<-interval.values[interval.mean==max(interval.mean)]
print(max.interval)
```

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

Imputation of missing data

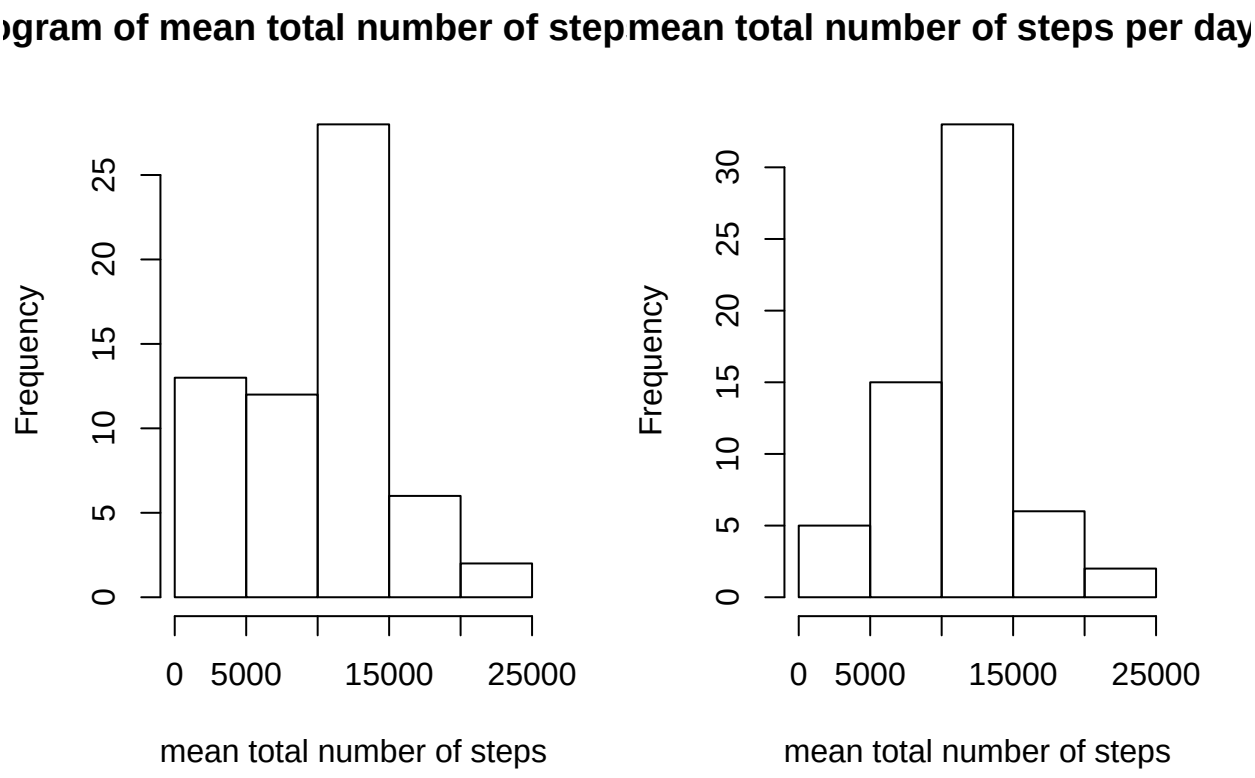
```
## total number of rows with missing data (NA)
total.NA<-sum(is.na(activity$steps))
print(total.NA)
```

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

```
## imputation rule: replace NA with the mean for that day

## calculate the mean for each day of the week
Saturday.mean<-mean(activity$steps[activity$day=="Saturday"],na.rm=TRUE)
Sunday.mean<-mean(activity$steps[activity$day=="Sunday"],na.rm=TRUE)
Monday.mean<-mean(activity$steps[activity$day=="Monday"],na.rm=TRUE)
Tuesday.mean<-mean(activity$steps[activity$day=="Tuesday"],na.rm=TRUE)
Wednesday.mean<-mean(activity$steps[activity$day=="Wednesday"],na.rm=TRUE)
Thursday.mean<-mean(activity$steps[activity$day=="Thursday"],na.rm=TRUE)
Friday.mean<-mean(activity$steps[activity$day=="Friday"],na.rm=TRUE)

## impute missing data
## copy original dataset
activity.imp<-activity
activity.imp$steps[is.na(activity.imp$steps) & activity.imp$day=="Saturday"]<-Saturday.mean
activity.imp$steps[is.na(activity.imp$steps) & activity.imp$day=="Sunday"]<-Sunday.mean
activity.imp$steps[is.na(activity.imp$steps) & activity.imp$day=="Monday"]<-Monday.mean
activity.imp$steps[is.na(activity.imp$steps) & activity.imp$day=="Tuesday"]<-Tuesday.mean
activity.imp$steps[is.na(activity.imp$steps) & activity.imp$day=="Wednesday"]<-Wednesday.mean
activity.imp$steps[is.na(activity.imp$steps) & activity.imp$day=="Thursday"]<-Thursday.mean
activity.imp$steps[is.na(activity.imp$steps) & activity.imp$day=="Friday"]<-Friday.mean
```

Recalculate using imputed data set and compare with results obtained when missing data were not imputed

```
## calculate total number of steps taken each day using the aggregate function
day.mean.imp<-as.numeric(unlist(aggregate(activity.imp$steps, by=list(activity.imp$date), FUN=sum, na.r

## compare with unimputed results
## using histograms
par(mfrow=c(1,2))
hist(day.mean, xlab="mean total number of steps",main="Histogram of mean total number of steps per day"
hist(day.mean.imp, xlab="mean total number of steps",main="Histogram of mean total number of steps per
```




```
## and using mean and median
print(c("mean without imputation",round(mean(day.mean),2)))
```

```
## [1] "mean without imputation" "9354.23"
```

```
print(c("mean with imputation",   round(mean(day.mean.imp),2)))
```

```
## [1] "mean with imputation" "10821.21"
```

```
print(c("median without imputation",round(median(day.mean),2)))
```

```
## [1] "median without imputation" "10395"
```

```
print(c("median with imputation",round(median(day.mean.imp),2)))
```

```
## [1] "median with imputation" "11015"
```

Identify weekends and add this identifier to the existing activity list

```
weekend<-(activity.imp$day=="Saturday" | activity.imp$day=="Sunday")

## add this to existing activity list
activity.imp[["daytype"]]<-weekend
activity.imp$daytype[activity.imp$daytype]<-"weekend"
activity.imp$daytype[activity.imp$daytype==FALSE]<-"weekday"
```

Split data into week days and weekend days (separate datasets)

```
activity.weekday<-activity.imp[activity.imp$daytype=="weekday",]
activity.weekend<-activity.imp[activity.imp$daytype=="weekend",]

# x-axis values for plotting each type of day
interval.values.weekday<-unique(activity.weekday$interval)
interval.values.weekend<-unique(activity.weekend$interval)

## calculate mean for each 5-minute interval using the aggregate function
## for weekdays
interval.mean.weekday<-as.numeric(unlist(aggregate(activity.weekday$steps, by=list(activity.weekday$inte
## for weekends
interval.mean.weekend<-as.numeric(unlist(aggregate(activity.weekend$steps, by=list(activity.weekend$inte

## make data frames for ggplot2
interval.weekday.df<-data.frame(cbind(interval.values.weekday,interval.mean.weekday))
interval.weekend.df<-data.frame(cbind(interval.values.weekend,interval.mean.weekend))
```

Panel plot

```
## weekdays pane
p1<-ggplot(interval.weekday.df, aes(x=interval.values.weekday, y=interval.mean.weekday)) +
  geom_line() +
  ggtitle("time series plot for week days") +
  labs(x="interval label") +
  labs(y="mean over 5-minute interval")
## weekends panel
p2<-ggplot(interval.weekend.df, aes(x=interval.values.weekend, y=interval.mean.weekend)) +
  geom_line() +
  ggtitle("time series plot for weekend days") +
  labs(x="interval label") +
  labs(y="mean over 5-minute interval")
```

Function multiplot

```
############################
# Multiple plot function
############################
#
# ggplot objects can be passed in ..., or to plotlist (as a list of ggplot objects)
```

```
# - cols:   Number of columns in layout
# - layout: A matrix specifying the layout. If present, 'cols' is ignored.
#
# If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE),
# then plot 1 will go in the upper left, 2 will go in the upper right, and
# 3 will go all the way across the bottom.
#
multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
  library(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }

  if (numPlots==1) {
    print(plots[[1]])

  } else {

    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                      layout.pos.col = matchidx$col))
    }
  }
}
```

Panel plot via function multiplot

```
multiplot(p1, p2, cols=1)
```