Statistics 5014: Homework 3

Due Wednesday, September 12, 10 am

2018-09-12

Problem 4

Coding is akin to writing literature, i.e, using correct punctuation. It is important to be consistent and concise throughout the text, so that reviewing and debugging can be performed efficiently. I will attempt to implement the nomenclature scheme that the style guides suggests to be more consistent in naming and differentiating my variables and functions.

Problem 5: lint HW2

```r
lint(filename = "./02_data_munging_summarizing_R_git/HW2_Park_Stephen.Rmd")
```

As a result of linting HW2, the following issues were brought up:

- Excessively long lines ($>$ 80 characters),
- inconsistent spacing around infix operators and commas,
- use of single quotes over double-quotes, and
- variable and function names that are not all lowercase.

The general message from the linting process is that the code syntax needs to be consistent and arranged in a manner so that it is easy to review. The following changes were made to the existing code:

- Excessively long commands and/or comments were divided so that each line did not occupy more than 80 characters.
- Spaces were included before and after infix operators, and after commas.
- All single quotes were substituted with double quotes.
- Only lowercase was used for variable and function names.

Problem 6

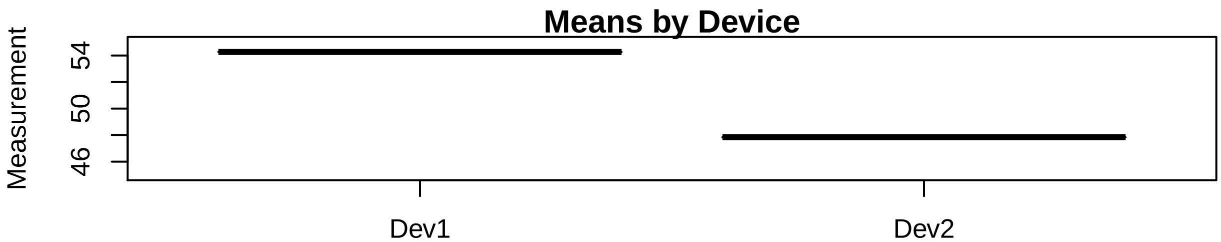
```r
dev.input <- readRDS("HW3_data.rds")   # Read in dataset from RDS file

# Define function to display descriptive statistics
DevStats <- function(input){
  obs.count <- input %>% count(Observer)  # Frequency of each Observer
  # Create empty output table with target variables
  output <- data.frame(id = obs.count[, 1],
                       mean.dev1 = numeric(nrow(obs.count)),
                       mean.dev2 = numeric(nrow(obs.count)),
                       sd.dev1 = numeric(nrow(obs.count)),
                       sd.dev2 = numeric(nrow(obs.count)),
                       corr = numeric(nrow(obs.count)))
# Group data by Observer and calculate descriptive statistics
  for (i in 1:nrow(obs.count)){
    sub.data <- filter(input, Observer == i)
    output$mean.dev1[i] <- mean(sub.data[, 2])
    output$mean.dev2[i] <- mean(sub.data[, 3])
    output$sd.dev1[i] <- sd(sub.data[, 2])
    output$sd.dev2[i] <- sd(sub.data[, 3])
    output$corr[i] <- cor(sub.data[, 2], sub.data[, 3])
  }
  return(output)
}
dev.output <- DevStats(dev.input)   # Run function with given data

# Create table for output data
kable(dev.output, digits = 3, format = "pandoc",
      caption = "Descriptive Statistic Summary of Device Measurements",
      col.names = c("Observer", "Dev1 Mean", "Dev2 Mean", "Dev1 SD",
                    "Dev2 SD", "Correlation"))
```

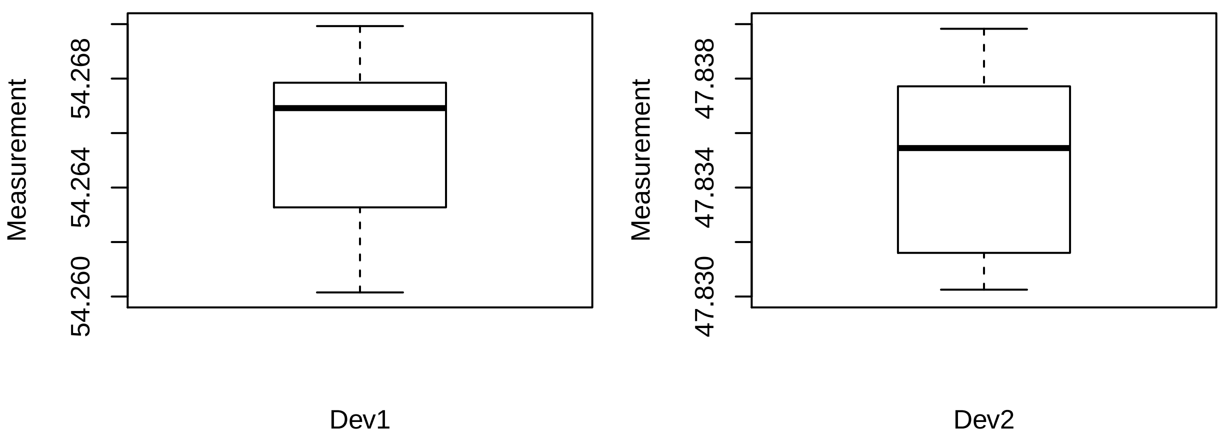
Table 1: Descriptive Statistic Summary of Device Measurements

Observer	Dev1 Mean	Dev2 Mean	Dev1 SD	Dev2 SD	Correlation
1	54.266	47.835	16.770	26.940	-0.064
2	54.269	47.831	16.769	26.936	-0.069
3	54.267	47.838	16.760	26.930	-0.068
4	54.263	47.832	16.765	26.935	-0.064
5	54.260	47.840	16.768	26.930	-0.060
6	54.261	47.830	16.766	26.940	-0.062
7	54.269	47.835	16.767	26.940	-0.069
8	54.268	47.836	16.767	26.936	-0.069
9	54.266	47.831	16.769	26.939	-0.069
10	54.267	47.840	16.769	26.930	-0.063
11	54.270	47.837	16.770	26.938	-0.069
12	54.267	47.832	16.770	26.938	-0.067
13	54.260	47.840	16.770	26.930	-0.066

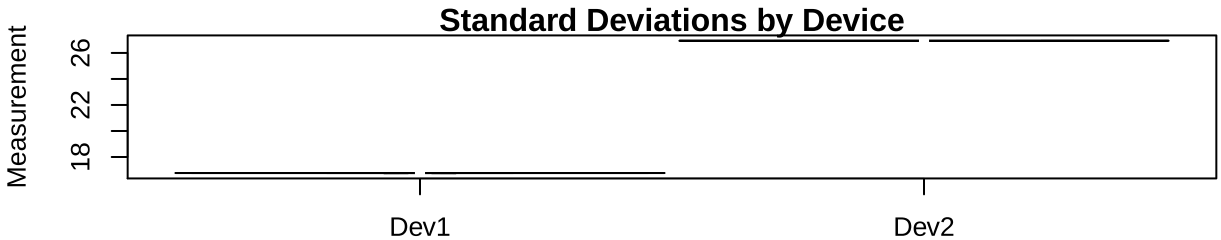
```r
# Configure boxplot layout
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE),
```

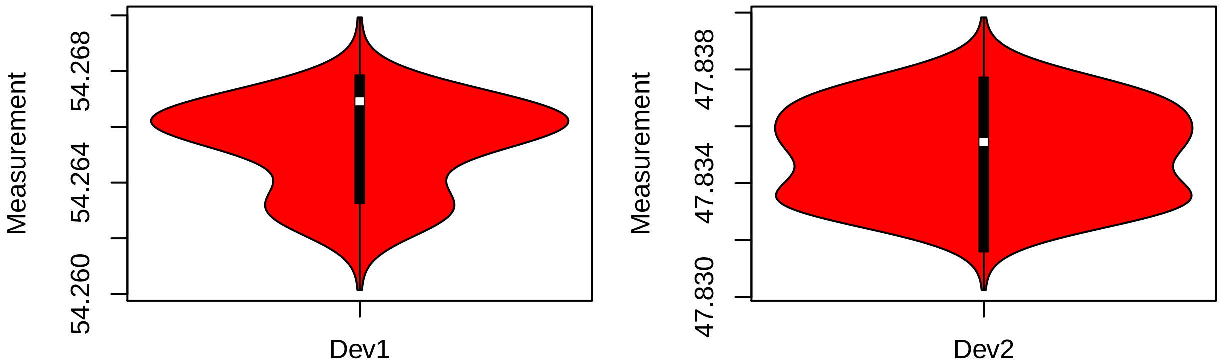
```
   widths=c(2,2), heights=lcm(c(4,6)))
par(mar=c(4, 4, 1, 1))
# Side-by-side boxplot to compare means
boxplot(dev.output$mean.dev1, dev.output$mean.dev2, main="Means by Device",
        ylab="Measurement", ylim = c(45, 55), names = c("Dev1", "Dev2"))
# Detailed boxplots to view spreads
boxplot(dev.output$mean.dev1, xlab = "Dev1", ylab = "Measurement",
        ylim = c(54.26, 54.27))
boxplot(dev.output$mean.dev2, xlab = "Dev2", ylab="Measurement",
        ylim = c(47.83, 47.84))
```






```
# Configure violin plot layout
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE),
   widths=c(2,2), heights=lcm(c(4,6)))
par(mar=c(4, 4, 1, 1))
# Side-by-side violin plot to compare means
violin_plot(data.frame(dev.output$sd.dev1, dev.output$sd.dev2), main = "Standard Deviations by Device",
# Detailed boxplots to view spreads
violin_plot(dev.output$mean.dev1, x_axis_labels = "Dev1", ylab = "Measurement", main = NA)
violin_plot(dev.output$mean.dev2, x_axis_labels = "Dev2", ylab = "Measurement", main = NA)
```





Problem 7

```
# Store URL for blood pressure data
url <- "http://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/BloodPressure.dat"
# Import data; use 2nd row as header
data_mess <- fread(url, skip = 1, header = T, fill = T, sep = " ")
# Preserve messy data and manipulate using a new variable
data_tidy <- data_mess %>% gather(key = Reader, value = Blood.Pressure) %>%
    filter(Reader != "Day") %>% arrange(Day)
head(data_tidy, 7)  # Preview data set
```

```
##   Day Reader Blood.Pressure
## 1   1   Dev1         133.34
## 2   1   Dev2         133.36
## 3   1   Dev3         133.45
## 4   1   Doc1         126.54
## 5   1   Doc2         127.36
## 6   1   Doc3         131.88
## 7   2   Dev1         110.94
```

```
summary(data_tidy)  # Summarize data
```

```
##       Day         Reader          Blood.Pressure
##  Min.   : 1   Length:90          Min.   :110.8
##  1st Qu.: 4   Class :character   1st Qu.:125.5
##  Median : 8   Mode  :character   Median :130.4
##  Mean   : 8                      Mean   :129.0
##  3rd Qu.:12                      3rd Qu.:134.3
##  Max.   :15                      Max.   :139.6
```

```r
str(data_tidy)  # Display blood pressure data structure
```

```
## 'data.frame':    90 obs. of  3 variables:
##  $ Day           : int  1 1 1 1 1 1 2 2 2 2 ...
##  $ Reader        : chr  "Dev1" "Dev2" "Dev3" "Doc1" ...
##  $ Blood.Pressure: num  133 133 133 127 127 ...
```

Problem 8: Newton's Method

```r
# Pre-define function
func <- function(x) {
  3^x - sin(x) + cos(5*x)
}
# Define Newton's Method function with the following input:
# f = given funtion
# x_0 = initial x
# tol = tolerance
# n = max number of iterations
newton <- function(f, x_0, tol = 1e-5, n = 1000) {
  require(numDeriv)
  # If x_0 is zero, end function and output x_0
  if (f(x_0) == 0.0) {
    return(x_0)
  }
  x <- x_0
  for (i in 1:n) {
    x[i+1] <- x[i] - f(x[i])/genD(func = f, x = x[i])$D[1]
    if (x[i+1] - x[i] < tol) {
      print("Convergence achieved")
      break
    }
    if(i == n) {
      print("Method did not converge")
    }
  }
  print(paste0("The tolerance for convergence is ", tol))
  plot(x[1:length(x) - 1], f(x[1:length(x) - 1]), col = "blue", ylab = "f(x)", xlab = "x", xlim = c(-ma
  points(x[length(x)], f(x[length(x)]), col = "red")
  abline(h=0)
  abline(v=0)
  x.data <- tibble::rowid_to_column(data.frame(x), "Iterations")
  kable(x.data, digits = 3, format = "pandoc",
      caption = "Iterative Estimations in Newton's Method")
}
# Test the method with given function
newton(func, -10, tol = 1e-6)
```

```
## Loading required package: numDeriv

## [1] "Convergence achieved"
## [1] "The tolerance for convergence is 1e-06"
```

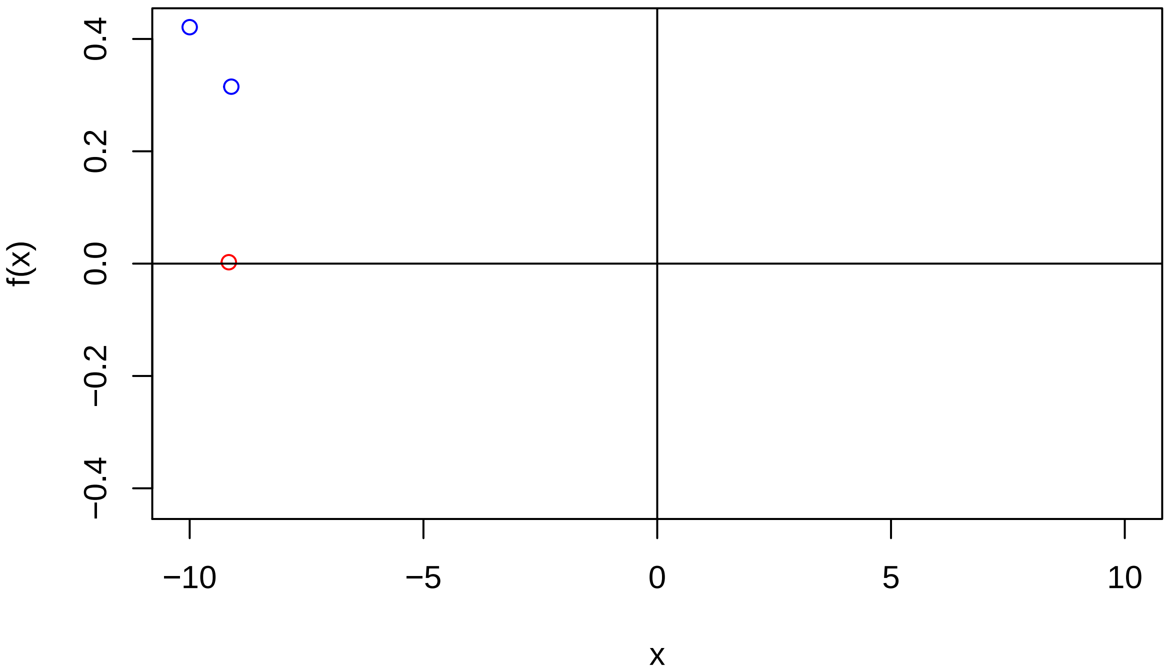


Table 2: Iterative Estimations in Newton's Method

Iterations	x
1	-10.000
2	-9.110
3	-9.163