HW3_Sauer_Annie

Annie Sauer

9/12/2018

Problem 4

I understand the importance of writing readable code. As a student I will have professors grading my coding assignments, and I will have to make sure they can easily understand my work. I personally will adjust my variable names, add proper spacing, and add frequent comments to my code as I move forward. Adding comments may be difficult to remember, but it is also one of the most important aspects of making my code readable. Taking these steps now will help me when I enter the job market in several years.

Problem 5

```
# lint(filename = "HW2_Sauer_Annie.Rmd")
```

The lint function recommended I change the following in my code:

- Use double quotes instead of single quotes
- Keep my lines at 80 characters or fewer
- Put a space after every comma
- Use only lowercase letters in my variable names

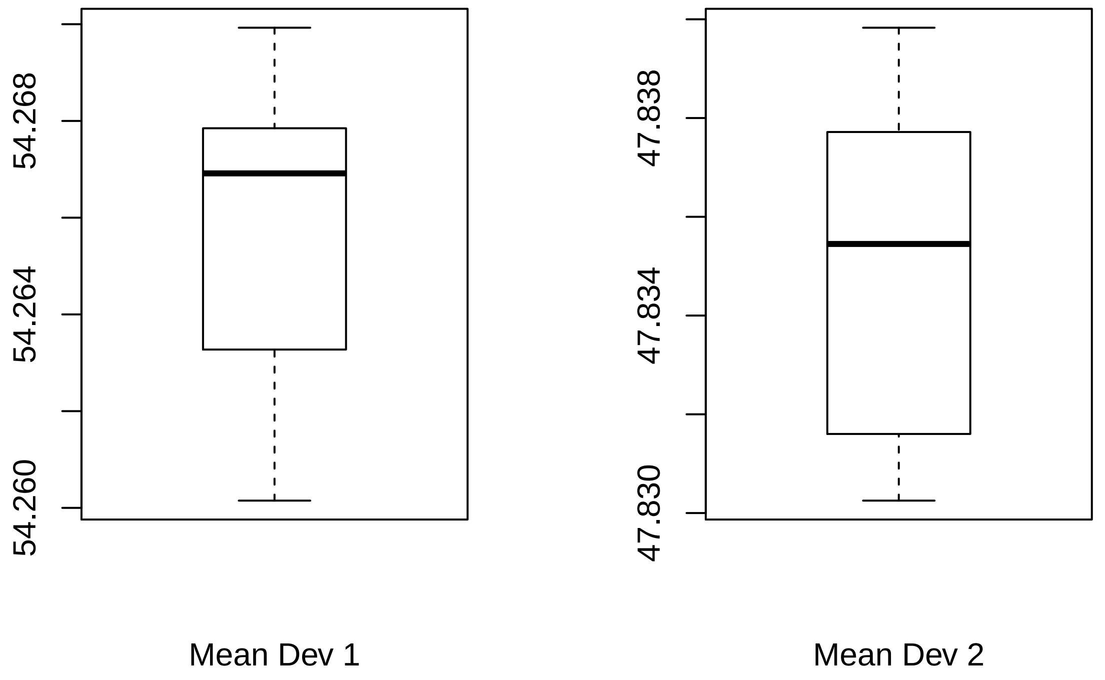
I will implement the second and third changes, but I prefer single quotes and mixing uppercase and lowercase letters in variable names. Single quotes are more convenient, and uppercase letters can help clarify variable names in some situations.

Problem 6

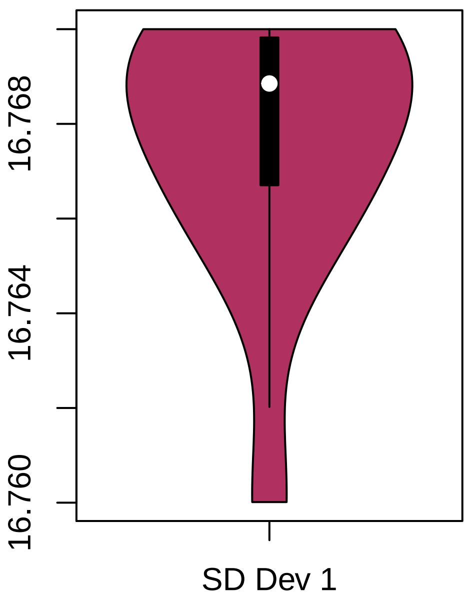
```
# Read data
data <- readRDS('03_good_programming_R_functions/HW3_data.rds')

# Create a function to output mean by observer, sd by observer, and correlation
# Notice: we could run this code outside of the function and get the same output

my_function <- function(input){
  input %>%
    group_by(Observer) %>%
    summarise(mean1 = mean(dev1), mean2 = mean(dev2), sd1 = sd(dev1), sd2 = sd(dev2),
              correlation = cor(dev1, dev2))
}

# Run my_function on given data set
new_df <- my_function(data)
new_df
```

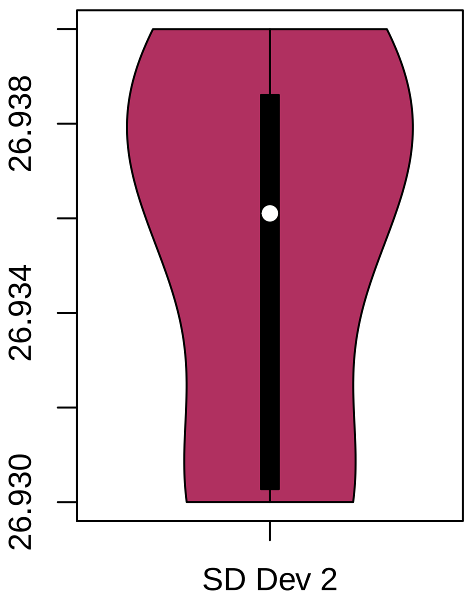
```
## # A tibble: 13 x 6
##    Observer mean1 mean2   sd1   sd2 correlation
##       <dbl> <dbl> <dbl> <dbl> <dbl>       <dbl>
##  1        1  54.3  47.8  16.8  26.9     -0.0641
## 2         2  54.3  47.8  16.8  26.9      -0.0686
## 3         3  54.3  47.8  16.8  26.9      -0.0683
## 4         4  54.3  47.8  16.8  26.9      -0.0645
## 5         5  54.3  47.8  16.8  26.9      -0.0603
## 6         6  54.3  47.8  16.8  26.9      -0.0617
## 7         7  54.3  47.8  16.8  26.9      -0.0685
## 8         8  54.3  47.8  16.8  26.9      -0.0690
## 9         9  54.3  47.8  16.8  26.9      -0.0686
## 10       10  54.3  47.8  16.8  26.9      -0.0630
## 11       11  54.3  47.8  16.8  26.9      -0.0694
## 12       12  54.3  47.8  16.8  26.9      -0.0666
## 13       13  54.3  47.8  16.8  26.9      -0.0656
```

```
# Produce box plots of mean values
par(mfrow=c(1,2))
boxplot(new_df$mean1, xlab = 'Mean Dev 1')
boxplot(new_df$mean2, xlab = 'Mean Dev 2')
```



```
# Produce violin plots of standard deviation values
par(mfrow=c(1,2))
vioplot(new_df$sd1, col='maroon', names = 'SD Dev 1')
vioplot(new_df$sd2, col='maroon', names = 'SD Dev 2')
```





Problem 7

```r
# Read data
data <- fread('https://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/BloodPressure.dat',
              sep = ' ')

# Gather day variable into a single column
data_new <- gather(data, Day)
# Rename columns
names(data_new) <- c('day','doctor_device','value')
head(data_new)
```

```
##   day doctor_device  value
## 1   1          Dev1 133.34
## 2   2          Dev1 110.94
## 3   3          Dev1 118.54
## 4   4          Dev1 137.94
## 5   5          Dev1 139.52
## 6   6          Dev1 139.23
```

```r
summary(data_new)
```

```
##       day     doctor_device          value
##  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
```

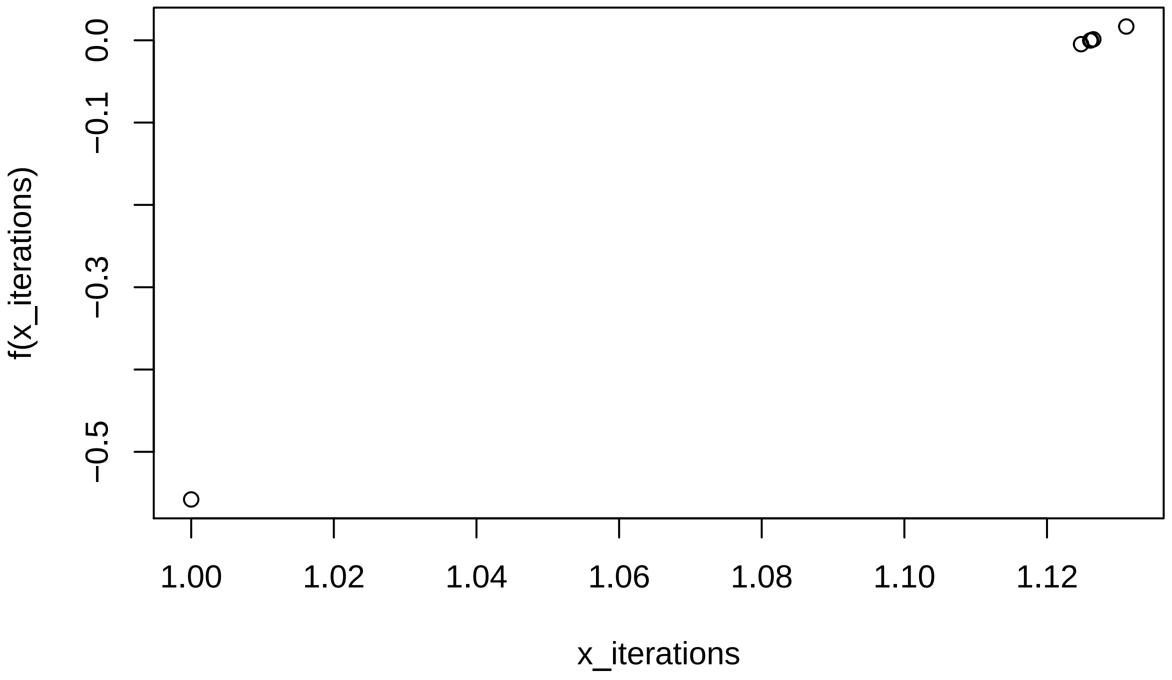
In this tidy data set we have one variable "doctor_device" that has six possible values. Ideally we would split this into two variables, but we have no way of pairing the device and doctor values from the original data set.

Problem 8

```
# Create Newton Method function
newton_method <- function(x0,t,n,f,g){
  # INPUTS
  # x0: initial guess
  # t: tolerance
  # n: maximum number of iterations
  # f: function f(x)
  # g: derivative of f(x)
  # OUTPUTS
  # Estimated x value such that f(x) = 0
  # Tolerance (t)
  # Scatter plot of iteration values

  # Initialize vector of iterations and count
  x_iterations <- c(x0)
  count <- 0
  while ( f(x0) > t | f(x0) < -t){
    # Update estimate
    x0 <- x0 - f(x0)/g(x0)
    # Add one to count
    count <- count + 1
    # Add updated estimate to iteration vector
    x_iterations <- append(x_iterations, x0,
                           after = length(x_iterations))
    # Break loop if maximum number of iterations is reached
    if (count >= n){
      cat('Maximum Iterations Reached \n')
      break
    }
  }
# Print x, t, and plot of iterations
cat('Estimated zero at x = ', x0,'\n')
cat('Tolerance = ', t,'\n')
cat('Number of Iterations = ', count,'\n')
plot(x_iterations, f(x_iterations))
}

# Original function f(x)
f <- function(x){
  logb(x,3) - sin(x) + cos(5*x)
}

# Derivative function g(x)
g <- function(x){
  log(3)*logb(x,3) - cos(x) - 5*sin(5*x)
}

# Initial guess
x <- 1

# Tolerance
```

```r
t <- 0.0001

# Maximum number of iterations
n <- 1000

# Call function
newton_method(x,t,n,f,g)
```

```
## Estimated zero at x =  1.126176
## Tolerance =  1e-04
## Number of Iterations =  5
```


f(x_iterations)
x_iterations