Homework 3

Yueyao Wang

September 16, 2017

Problem 4

There are some programming skills in notations, syntax, functions and organisation to make code clean and clear:

- Using lowercase and underscore in the variable names
- Placing spaces around operators and before left parentheses, except in a function call.
- The opening curly brace should never go on its own line and should always followed by a new line. A closing curly brace should always go on its own line, unless it's followed by else.
- Each lines shouldn't exceed 80 characters.
- When function definition runs over multiple lines, indent the second line where the definition starts.
- Commenting code and use - or = to seperate code into readable chunks.

I will pay attention to improve the identation and comments.

Problem 5

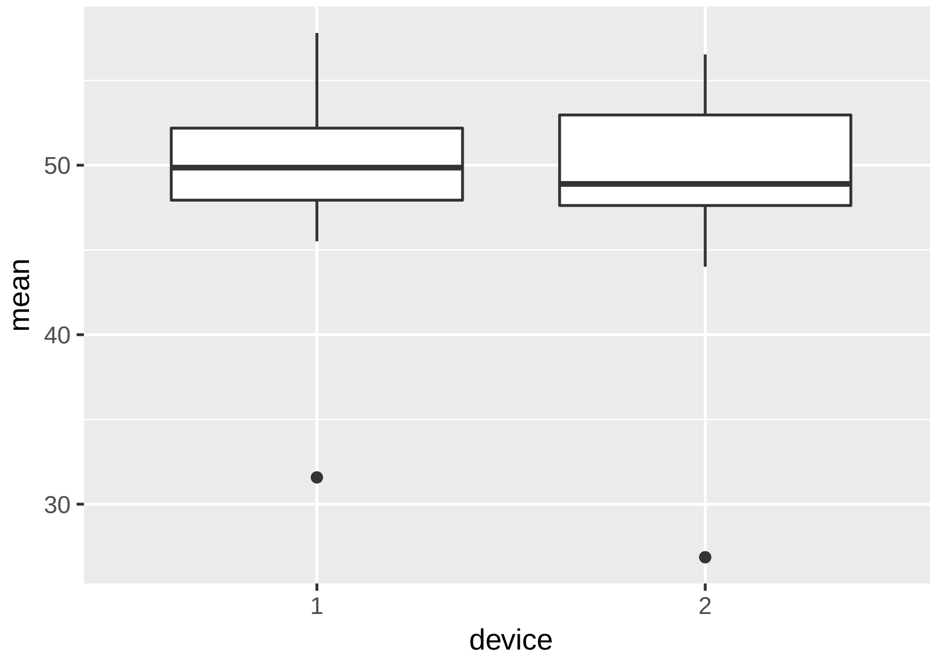
```r
library(lintr)
lint(filename = "../02_data_munging_summarizing_R_git/HW2_Wang_Yueyao.Rmd")
```

- Pay attention that lines shouldn't more than 80 characters.
- Variable and function names should be all lowercase.
- Only use double-quotes for text.
- Commas should always have a space after.

Problem 6

Table 1: Summary of data by Obersvers

	mean_dev1	mean_dev2	sd_dev1	sd_dev2	correlation
Observer1	31.5807	26.8694	30.2125	26.2174	-0.0641
Observer2	45.5146	48.8248	40.9673	42.6652	-0.0686
Observer3	46.0901	44.0151	39.0910	38.3621	-0.0683
Observer4	52.1880	50.5214	46.6719	46.0212	-0.0645
Observer5	49.0051	48.2065	42.9738	42.0988	-0.0603
Observer6	49.8582	49.6300	42.2135	42.1337	-0.0617
Observer7	49.5061	47.0308	39.0883	38.0867	-0.0685
Observer8	52.7384	53.5297	44.6718	43.0503	-0.0690
Observer9	50.6455	48.8953	43.1962	42.2014	-0.0686
Observer10	53.3652	53.4703	41.1557	41.3107	-0.0630
Observer11	51.5681	52.9641	41.1221	43.3672	-0.0694
Observer12	57.8013	56.5352	42.3666	40.7425	-0.0666
Observer13	47.9368	47.6229	40.5966	41.8410	-0.0656



Problem 7

The issue with this dataset is that:

- Column headers are values: the columns Devs and Docs in the original dataset should be the value of a variable that indicating the pressure read by devices or doctors.

- Multiple variables are stored in one column: Also we should have a variable of ID number indicating which device or doctors read this value, not combining them togehter.

So I would like to clean and munge the data into 4 variables:

- Day: the day recording those values

- Type: indicating the value is read by doctor or device
- ID: indicating which device or doctor read this value
- Pressure: the corresponding blood pressure

The tidying process:

- Use the gather function to convert those columns header Devs and Docs into a variable read_by_type
- Separate this characters and numbers in read_by_type variable into two columns Type and ID via mutate and gsub function
- Select the columns that we want to keep and arrange the rows by the ascending order of Day

So the first 10 rows of the tidy data is following:

Table 2: 10 Rows of Blood Pressure Data after Tidying

Day	type	ID	pressure
1	Dev	1	133.34
1	Dev	2	133.36
1	Dev	3	133.45
1	Doc	1	126.54
1	Doc	2	127.36
1	Doc	3	131.88
2	Dev	1	110.94
2	Dev	2	110.85
2	Dev	3	110.92
2	Doc	1	124.69

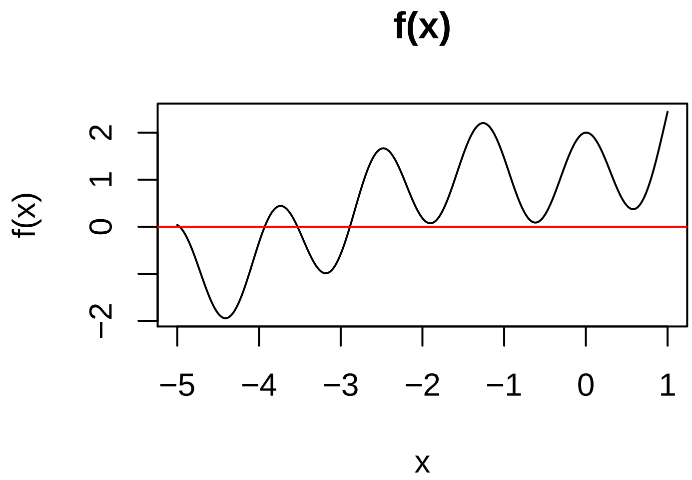
Problem 8

We need to calculate the derivative function of $f(x)$ in the Newton's method which is:

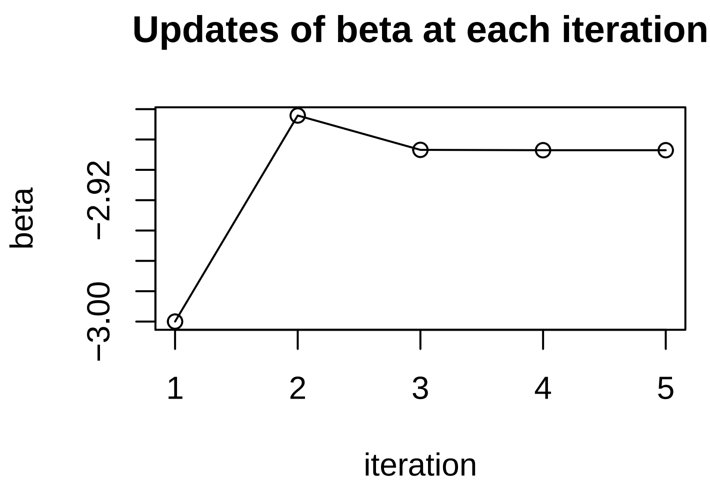
$$f^{'}(x) = 3^x log(3) - cos(x) - 5sin(5x)$$

####The procedure Newton's method is following:
set initial value of x denoted as x_0 and tolerance ϵ
repeat:
  update $x_{new} = x_0 - f(x_0)/f^{'}(x_0)$
until $|x_0 - x_{new}| \leq \epsilon$

Before directly applying this method, we can plot this function to choose an initial value. Because a bad initial point is easy to cause unconvergency.



From the above plot we could see that there isn't unique root for this function because the trigonometric function. We can just find the biggest negative root in this problem which is between -3 and -2 from the plot.



So from we could know, when the tolerance is 1e-5, the biggest negative root of this function is -2.8870577

Appendix: R code

```
####### Problem6: Construct summarise function ############
data <- readRDS("./HW3_data.rds")
#--------------------Construct functions---------------------------------------
# input: x is a dataframe for all samples from 1
# Observer return:a dataframe of descriptive statstics
descrip_stats <- function(x) {
    mean_dev1_dev2 <- apply(x, MARGIN = 1, FUN = mean)
    sd_dev1_dev2 <- apply(x, MARGIN = 1, FUN = sd)
    correlation <- cor(x[, 2], x[, 3])
    d <- data.frame(mean_dev1 = mean_dev1_dev2[1], mean_dev2 = mean_dev1_dev2[2],
        sd_dev1 = sd_dev1_dev2[1], sd_dev2 = sd_dev1_dev2[2],
        correlation = correlation)
    return(d)
}

#----------loop through the Observers via the descrip_stats function-----------
```

```r
obs_1 <- data[data$Observer == 1, ]
com_df <- descrip_stats(obs_1)
for (i in 2:13) {
    obs_i <- data[data$Observer == i, ]
    des_df <- descrip_stats(obs_i)
    com_df <- rbind(com_df, des_df)
}
rownames(com_df) <- paste("Observer", 1:13, sep = "")
kable(com_df, caption = "Summary of data by Obersvers",
    digits = 4) %>% kable_styling(full_width = T)

#-------------create boxplot for dev1 and dev2 mean-------------------------------
device_mean <- c(com_df[, 1], com_df[, 2])
device_index <- rep(c(1, 2), each = 13)
mean_summary <- data.frame(mean = device_mean, device = as.factor(device_index))
ggplot(data = mean_summary, aes(x = device, y = mean), main = "boxplot of the device mean") +
    geom_boxplot()

#----------------create violin plot fro dev1 and dev2 sd-------------------------------
device_sd <- c(com_df[, 3], com_df[, 4])
sd_summary <- data.frame(sd = device_sd, device = as.factor(device_index))
ggplot(data = sd_summary, aes(x = device, y = sd), main = "violin plot for standard deviation") +
    geom_violin() + stat_summary(fun.y = mean, geom = "point",
    color = "red", size = 2)
######## Problem 8:Plot the function to choose initial value
######## #########
f <- function(x) {
    value <- 3^x - sin(x) + cos(5 * x)
    return(value)
}
x <- seq(-5, 1, by = 0.01)
plot(x, f(x), main = "f(x)", type = "l")
abline(h = 0, col = "red")
######## Problem 8: Construct the function ########
derv_f <- function(x) {
    value <- 3^x * log(3) - cos(x) - 5 * sin(5 * x)
    return(value)
}
## input: initial value: set it as -3 default tolerance:
## 1e-5 as the default output: a vector of the updates of
## beta at each iteration
newton_method <- function(epsilon = 1e-05, beta_0 = -3) {
    beta <- c(beta_0)
    while (TRUE) {
        beta_new <- beta_0 - f(beta_0)/derv_f(beta_0)
        beta <- c(beta, beta_new)
        if (abs(beta_new - beta_0) < epsilon) {
            beta_est <- beta_new
            break
        } else {
            beta_0 <- beta_new
        }
    }
```

```r
    return(beta)
}
beta <- newton_method()
plot(1:length(beta), beta, xlab = "iteration", main = "Updates of beta at each iteration")
lines(1:length(beta), beta)
```