Homework 4

Due Wednesday 9am September 25, 2019

2019-09-21

For each assignment, turn in by the due date/time. Late assignments must be arranged prior to submission. In every case, assignments are to be typed neatly using proper English in Markdown.

This week, we spoke about R and version control, munging and 'tidying' data, good programming practice and finally some basic programming building blocs. To begin the homework, we will for the rest of the course, start by loading data and then creating tidy data sets.

Problem 1

Work through the "R Programming E" lesson parts 8 and 9.

From the R command prompt:

```r
library(swirl)
swirl()
```

Problem 2

Create a new R Markdown file (file–>new–>R Markdown–>save as.

The filename should be: HW4_lastname_firstname, i.e. for me it would be HW4_Settlage_Bob

You will use this new R Markdown file to solve problems 4-10.

Problem 4

In the lecture, there were two links to programming style guides. What is your takeaway from this and what specifically are *you* going to do to improve your coding style?

Answer Enssentially, there are at least two things for me to improve: 1. Name variables and functions: Use noun to name variable and verb to name function. Use lowercase. Use "_" to seperate words in a name. Be concise. 2. Indent: New line after curly braces, and closing curly brace in its own line. Use two space to indent code.

Problem 5

Good programming practices start with this homework. In the last homework, you imported, munged, cleaned and summarized datasets from Wu and Hamada's *Experiments: Planning, Design and Analysis*. In this problem, please using *lintr* to lint your last homework (if you didn't do it, perhaps the time is now ;)). In my case, the command looks like this (takes a few moments to run):

```r
library(lintr)
lint(filename = "./HW3_Chen_Han.Rmd")
```

Can you clean up your code to pass the major issues?? <— just a challenge, not part of the problem!!

Note that really all we have done is syntax checking and received a few stylistic suggestions. We COULD create a custom linter to check for indenting, etc. For now, I think it is enough to know there are a lot of opinions on what code should look like and working in teams requires you to code nicely!! So, clean up your code!!

From the messages, what are some things you need to change in your code?

Answer

1. Superfluous: a lot of space left unintentionally.
2. Infix space: space before or after infix like "="
3. Double quote: use double quote rather than single quote

Problem 6

A situation you may encounter is a data set where you need to create a summary statistic for each observation type. Sometimes, this type of redundancy is perfect for a function. Here, we need to create a single function to:

1. calculate the mean for dev1
2. calculate the mean for dev2
3. calculate the sd for dev1
4. calculate the sd for dev2
5. calculate the correlation between dev1 and dev2
6. return the above as a single data.frame

We will use this function to summarize a dataset which has multiple repeated measurements from two devices by thirteen Observers. In the current lecture directory, you will see a file "HW4_data.rds". Please load the file (?readRDS – really nice format for storing data objects), loop through the Observers collecting the summary statistics via your function for each Observer separately.

The output of this problem should be:

a. A single table of the means, sd, and correlation for each of the 13 Observers

b. A box plot of all the means to compare the spread of means from dev1 to dev2

c. A violin plot of all the sd to compare the spread of sd from dev1 to dev2

I will look at the code and comment on it, so make it NICE!!

Answer

```r
library(tidyverse)
df <- readRDS("HW4_data.rds")
df$Observer <- as.factor(df$Observer)

knitr::kable(head(df, n = 5),
             caption = "")
```

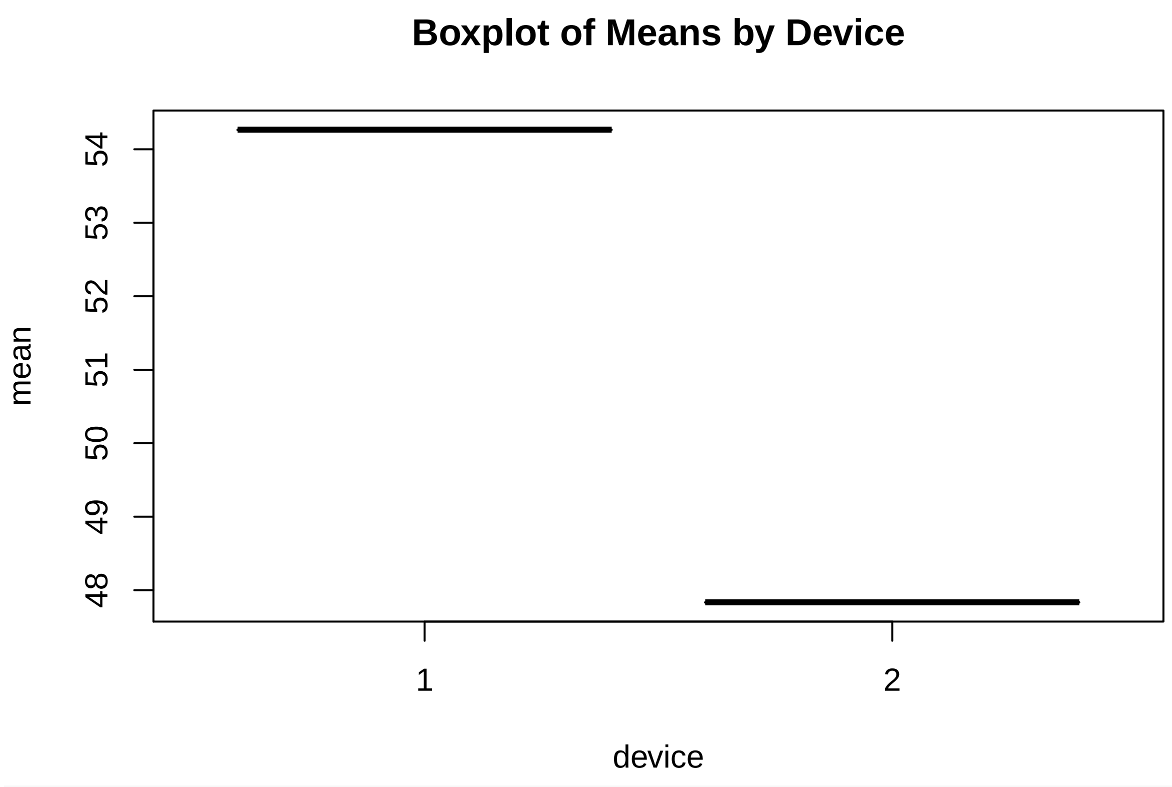
Observer	dev1	dev2
4	55.3846	97.1795
4	51.5385	96.0256
4	46.1538	94.4872
4	42.8205	91.4103
4	40.7692	88.3333

```r
df_stat <-
    df %>%
    group_by(Observer) %>%
    summarize(
     mean1 = mean(dev1),
     mean2 = mean(dev2),
     sd1 = sd(dev1),
     sd2 = sd(dev2),
     correlation = cor(dev1, dev2)
    )

knitr::kable(df_stat,
             caption = "Summary Statistics by Observer")
```

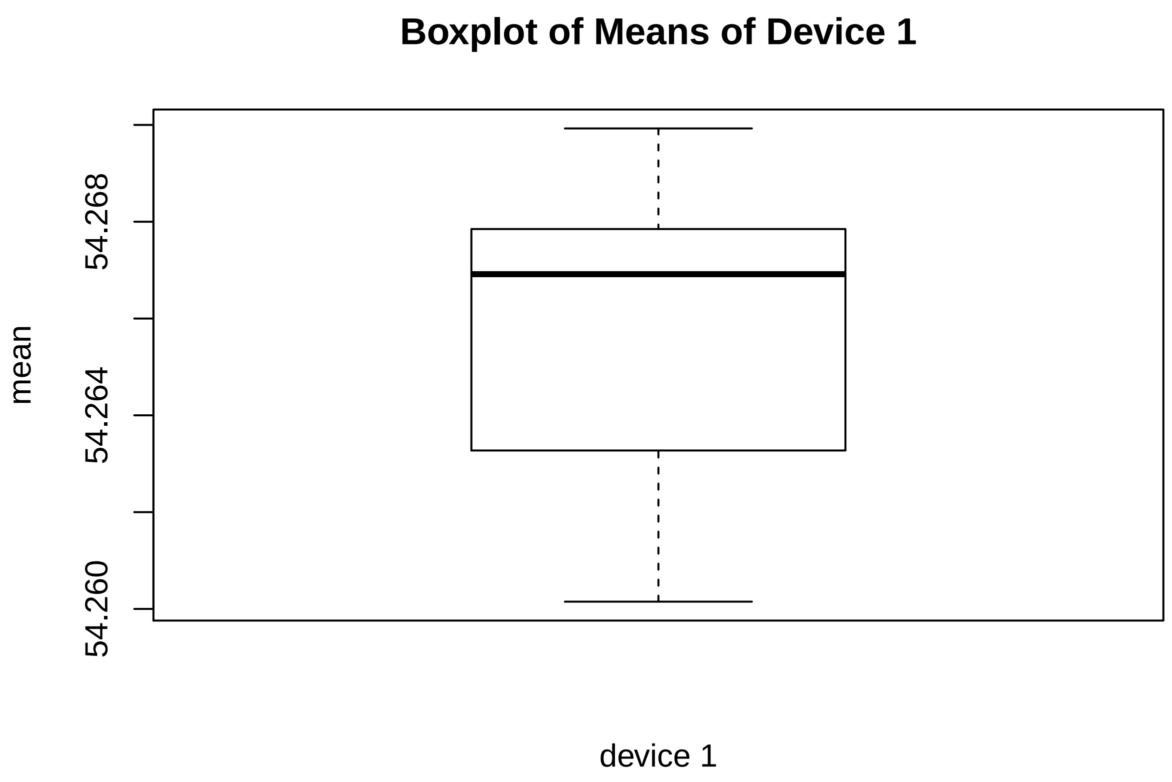
Table 2: Summary Statistics by Observer

Observer	mean1	mean2	sd1	sd2	correlation
1	54.26610	47.83472	16.76983	26.93974	-0.0641284
2	54.26873	47.83082	16.76924	26.93573	-0.0685864
3	54.26732	47.83772	16.76001	26.93004	-0.0683434
4	54.26327	47.83225	16.76514	26.93540	-0.0644719
5	54.26030	47.83983	16.76774	26.93019	-0.0603414
6	54.26144	47.83025	16.76590	26.93988	-0.0617148
7	54.26881	47.83545	16.76670	26.94000	-0.0685042
8	54.26785	47.83590	16.76676	26.93610	-0.0689797
9	54.26588	47.83150	16.76885	26.93861	-0.0686092
10	54.26734	47.83955	16.76896	26.93027	-0.0629611
11	54.26993	47.83699	16.76996	26.93768	-0.0694456
12	54.26692	47.83160	16.77000	26.93790	-0.0665752
13	54.26015	47.83972	16.76996	26.93000	-0.0655833

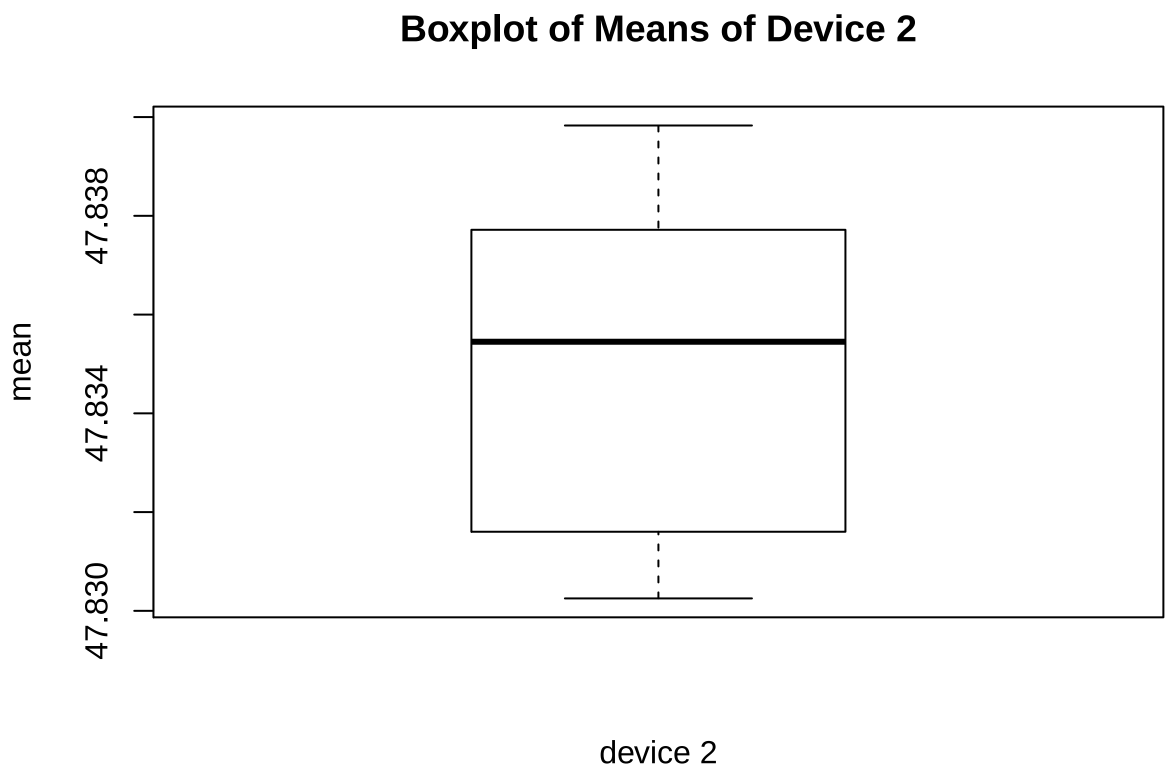
```r
data.frame(mean = c(df_stat$mean1, df_stat$mean2),
           device = as.factor(c(rep(1, 13), rep(2,13)))) %>%
boxplot(mean ~ device, .,
        xlab = "device",
        ylab = "mean",
        main = "Boxplot of Means by Device")
```



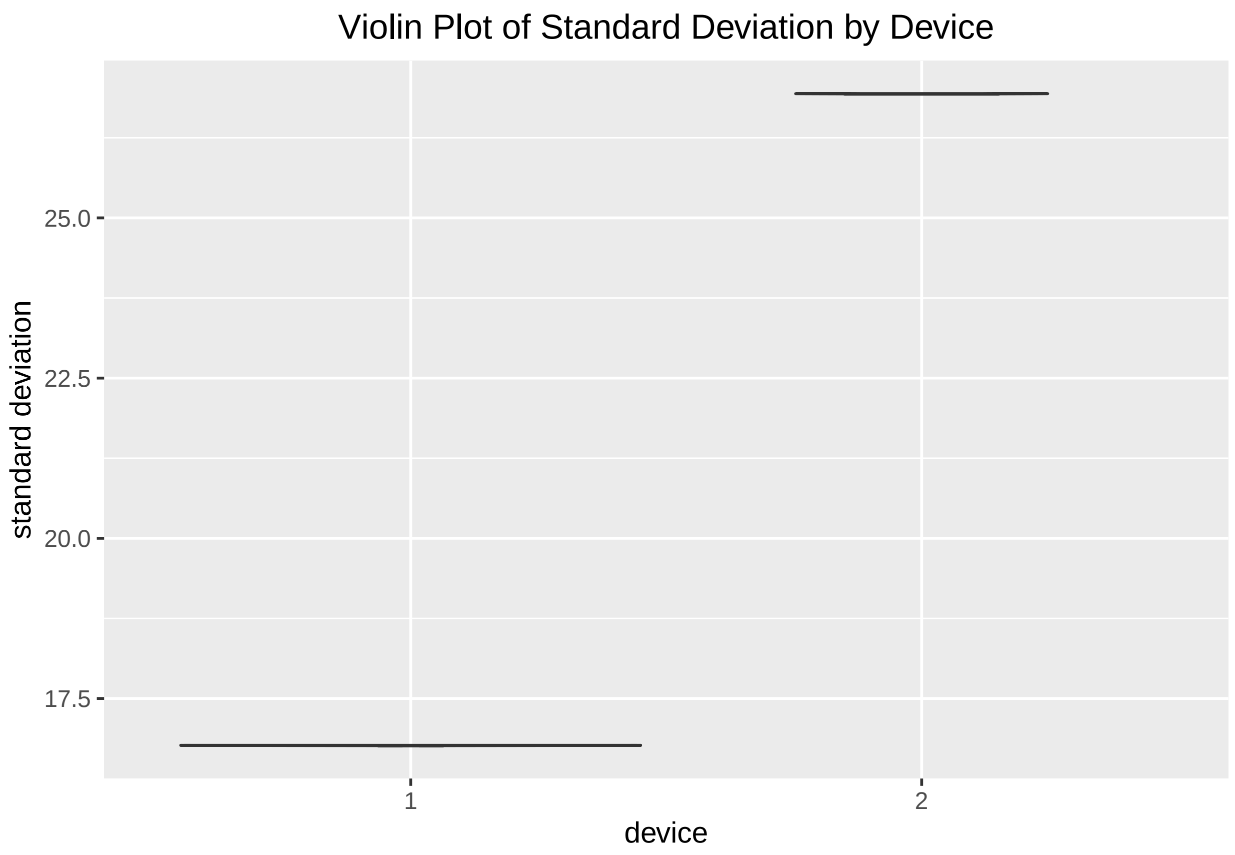
```
boxplot(df_stat$mean1,
        xlab = "device 1",
        ylab = "mean",
        main = "Boxplot of Means of Device 1")
```



```r
boxplot(df_stat$mean2,
        xlab = "device 2",
        ylab = "mean",
        main = "Boxplot of Means of Device 2")
```

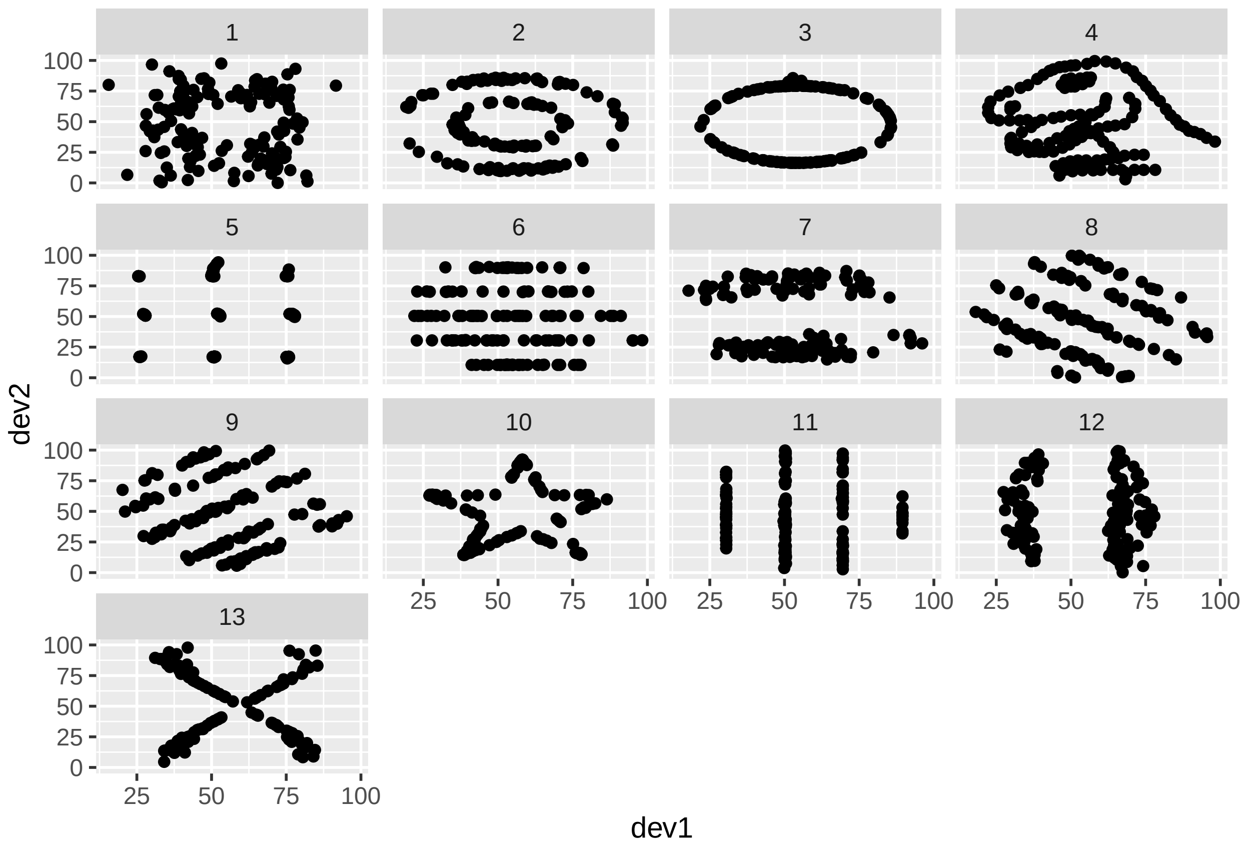


```r
data.frame(sd = c(df_stat$sd1, df_stat$sd2),
           device = as.factor(c(rep(1, 13), rep(2, 13)))) %>%
ggplot(., aes(device, sd)) +
  geom_violin() +
  ylab("standard deviation") +
  ggtitle("Violin Plot of Standard Deviation by Device") +
  theme(plot.title = element_text(hjust = 0.5))
```



Interesting Scatterplot

```
ggplot(data = df, aes(x = dev1, y = dev2)) +
  geom_point() +
  facet_wrap(vars(Observer))
```



Problem 7

Some numerical methods are perfect candidates for funtions. Create a function that uses Reimann sums to approximate the integral:

$$f(x) = \int_0^1 e^{-\frac{x^2}{2}} dx$$

The function should include as an argument the width of the slices used. Now use a looping construct (for or while) to loop through possible slice widths. Report the various slice widths used, the sum calculated, and the slice width necessary to obtain an answer within $1e^{-6}$ of the analytical solution.

Note: use good programming practices.

Answer

As we can see from the approximation results table below, when the slice width is less or equal than $1e^{-6}$, the error is within within $1e^{-6}$ of the analytical solution.

```r
reimann_sum <- function(width = 0.01){
  x <- seq(from = 0, to = 1, by = width)
  sum(exp(- x^2 / 2) * width)
}

# analytical solution using standard normal distribution
exact_value <- sqrt(2 * pi) * (pnorm(1) - pnorm(0))
```

```
# approximation
approx_results <- matrix(
                    nrow = 7, ncol = 3,
                    dimnames = list(c(), c("Slice Width", "Summation", "Error"))
                    )

for (i in 1:7){
  approx_results[i, 1] <- 10^(-i)
  approx_results[i, 2] <- reimann_sum(approx_results[i, 1])
  approx_results[i, 3] <- exact_value - approx_results[i, 2]
}

knitr::kable(approx_results)
```

Slice Width	Summation	Error
1e-01	0.9354453	-0.0798209
1e-02	0.8636520	-0.0080276
1e-03	0.8564276	-0.0008032
1e-04	0.8557047	-0.0000803
1e-05	0.8556324	-0.0000080
1e-06	0.8556252	-0.0000008
1e-07	0.8556245	-0.0000001

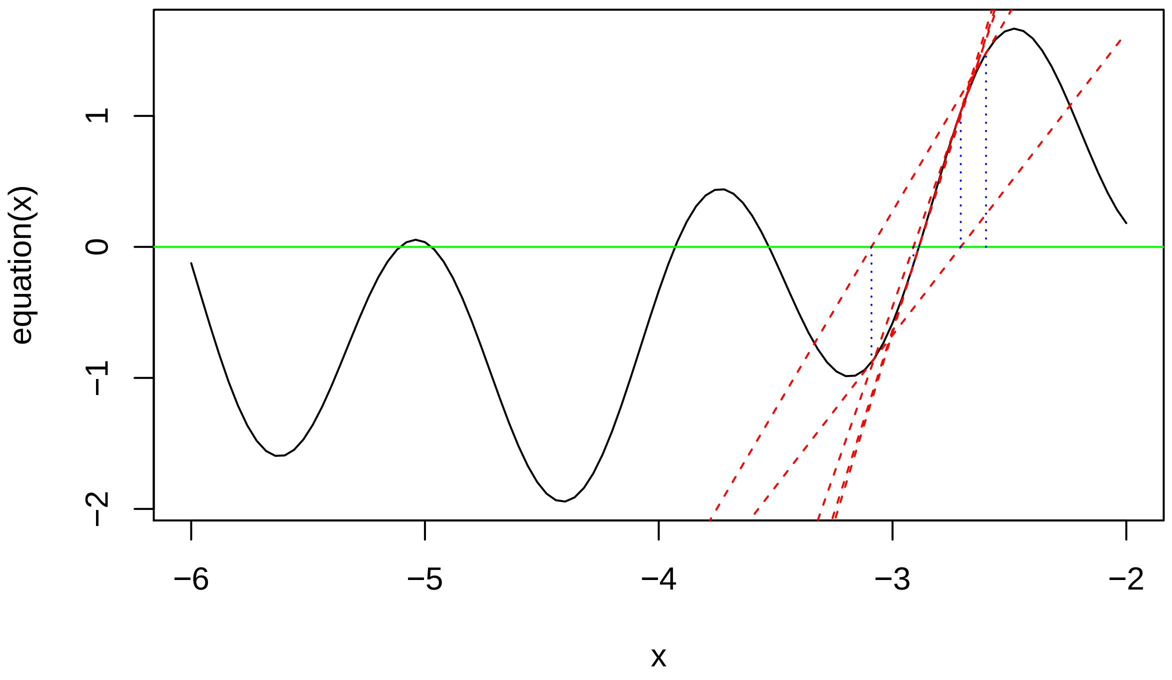
Problem 8

Create a function to find solutions to (1) using Newton's method. The answer should include the solutions with tolerance used to terminate the loop, the interval used, and a plot showing the iterations on the path to the solution.

$$f(x) = 3^x - sin(x) + cos(5x) \tag{1}$$

Answer

```
equation <- function(x){
  3^x - sin(x) + cos(5*x)
}

derivative <- function(x){
  3^x * log(3) - cos(x) - 5 * sin(5*x)
}

newton_method <- function(tolerance = 0.0001, start_point = -2.6){

  it_x <- c(start_point)
  error <- c(equation(it_x))
  i <- 1

  curve(equation, from = -6, to = -2)
  abline(h = 0, col = "green")

  while(abs(error[i]) > tolerance){
    segments(x0 = it_x[i], y0 = 0,
             x1 = it_x[i], y1 = equation(it_x[i]),
             col = "blue", lty = 3)

    curve(derivative(it_x[i]) * (x - it_x[i]) + equation(it_x[i]),
          add = TRUE, col = "red", lty = 2)

    i = i + 1

    it_x <- c(it_x, it_x[i-1] - equation(it_x[i-1]) / derivative(it_x[i-1]))

    error <- c(error, equation(it_x[i]))
  }

  print(paste("One of the solution to the equation is", it_x[i],
              ". The tolerance to terminate the loop is", tolerance,
              "and the interval used is from", min(it_x), "to", max(it_x)))
  knitr::kable(tbl_df(data.frame("x value" = it_x, error)))
}

newton_method()
```



```
## [1] "One of the solution to the equation is -2.88705759329088 . The tolerance to terminate the loop i
```

x.value	error
-2.600000	1.4804239
-3.090066	-0.8819437
-2.708189	1.0325183
-2.910607	-0.1341870
-2.886611	0.0025872
-2.887058	0.0000007

Problem 9

Finish this homework by pushing your changes to your repo.

Only submit the .Rmd and .pdf solution files. Names should be formatted HW4_lastname_firstname.Rmd and HW4_lastname_firstname.pdf

Optional preperation for next class:

Next week we will talk about Exploratory Data Analysis and graphing. Swirl will be a bit part of this. Check out "Exploratory_Data_Analysis" Swirl lessons 1-10.