# STAT 5014 Homework 3

Jingbin Xu

09/27/2020

## Problem 3

Here is my takeaway from the programming style guide:

- First, to optimize the experience of reviewing code and debugging, I will always leave a trace for the reader. Leave comments on the unusual functions.
- Avoid dangerous functions in my coding, such as exit, loop, break, etc. The infinite loop will cause the program to collapse and take a longer running time.
- I would also keep simplicity such as indenting. Millions of code lines without indention are challenging for my collaborators.

## Problem 5

### a. Table of means, sd, and correlation for each of the 13 Observers

According to the table, we noticed that the means, sd for each device across different observation are almost same. There exist weak negative correlation between two devices.

```
summary_dt <- function(dt) {
  ## Create empty dataframe for the iteration
  k <- cbind(seq(1,13),
             matrix(NA, nrow = 13, ncol = 5))
  ## rename the column names with obs, mean, sd and correlation
  colnames(k) <- c("obs", "mean1", "mean2", "sd1", "sd2", "correlation12")
  ## for loop to compute the mean, sd, cor for each obs
  for (i in 1:13) {
    k[i, 2] <- mean(dt5[dt5$obs == i, 2])
    k[i, 3] <- mean(dt5[dt5$obs == i, 3])
    k[i, 4] <- sd(dt5[dt5$obs == i, 2])
    k[i, 5] <- sd(dt5[dt5$obs == i, 3])
    k[i, 6] <- cor(dt5[dt5$obs == i, 2], dt5[dt5$obs == i, 3])

  }
  return(k)
}
# running the function of summary table
sum_table <- summary_dt(dt5)
# kable of the sum_table
summary_table <- knitr::kable(sum_table,
                              # align the columns at center
                              align = "cccccc",
                              # set the main title of the table
```

```
                              caption = "Summary Table",
                              # round the nums to 3 decimal
                              digits = 3)
summary_table
```

Table 1: Summary Table

| obs | mean1 | mean2 | sd1 | sd2 | correlation12 |
|---|---|---|---|---|---|
| 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 |

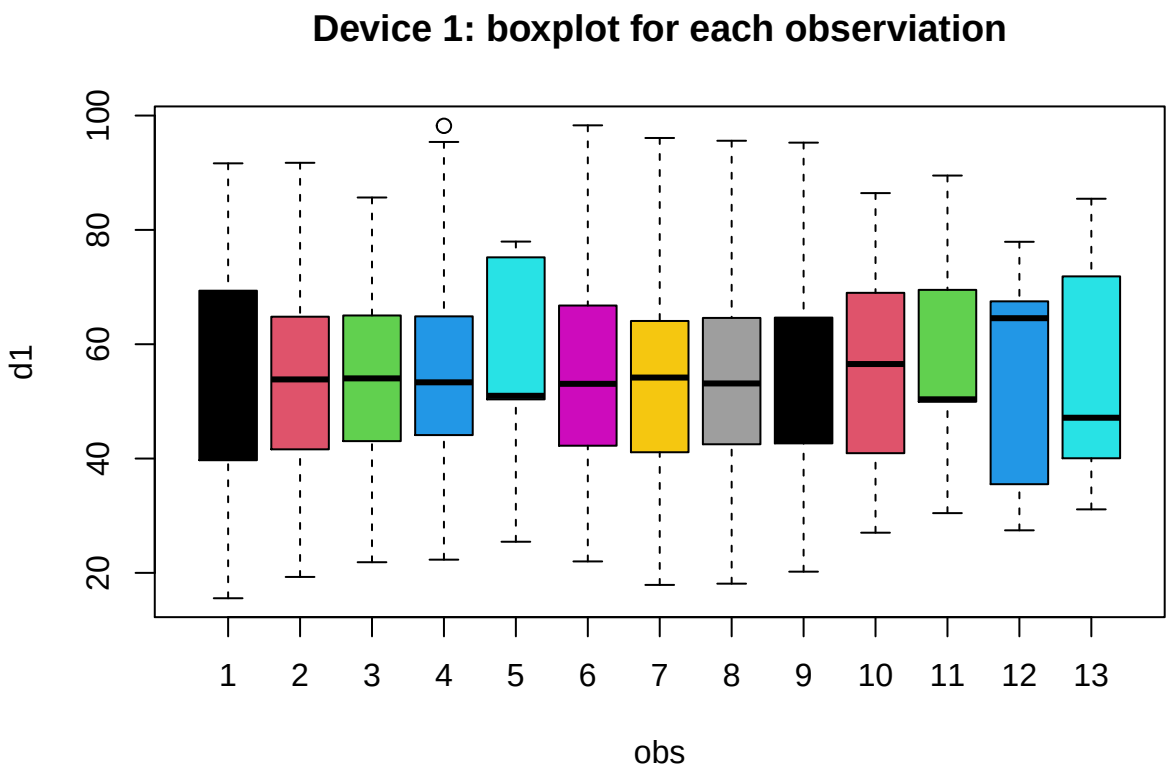
```
# Check the results using group_by and summarise
# dt5 %>% group_by(obs) %>%
# summarise(mean1 = mean(d1), mean2 = mean(d2), sd1 = sd(d1), sd2 = sd(d2), correlation = cor(d1, d2))
```

**b. A box plot of dev, by Observer.**

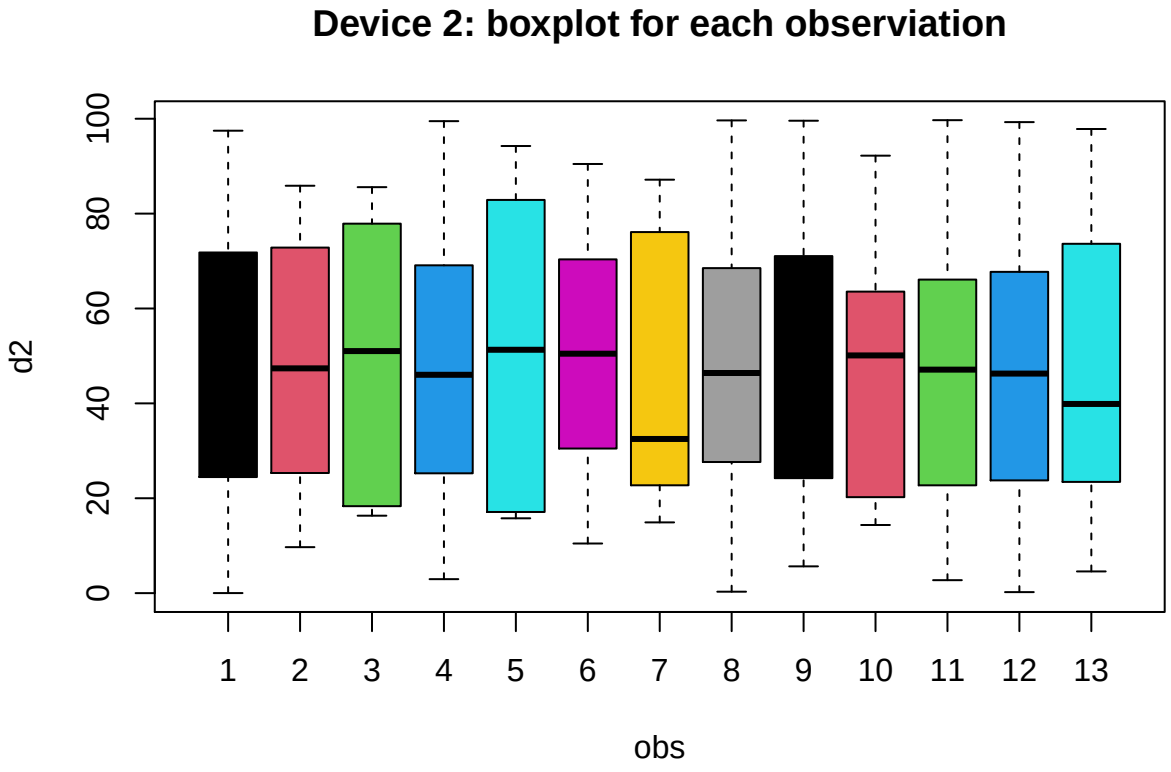
Device 1: * Potential outlier for observation 4. * More skewness for each observation compare to device 2

Device 2: * No potential outliers. * Less skew for each observation compare to device 1

```
boxplot(d1 ~ obs, data = dt5, col = c(1:13), main = "Device 1: boxplot for each observiation")
```



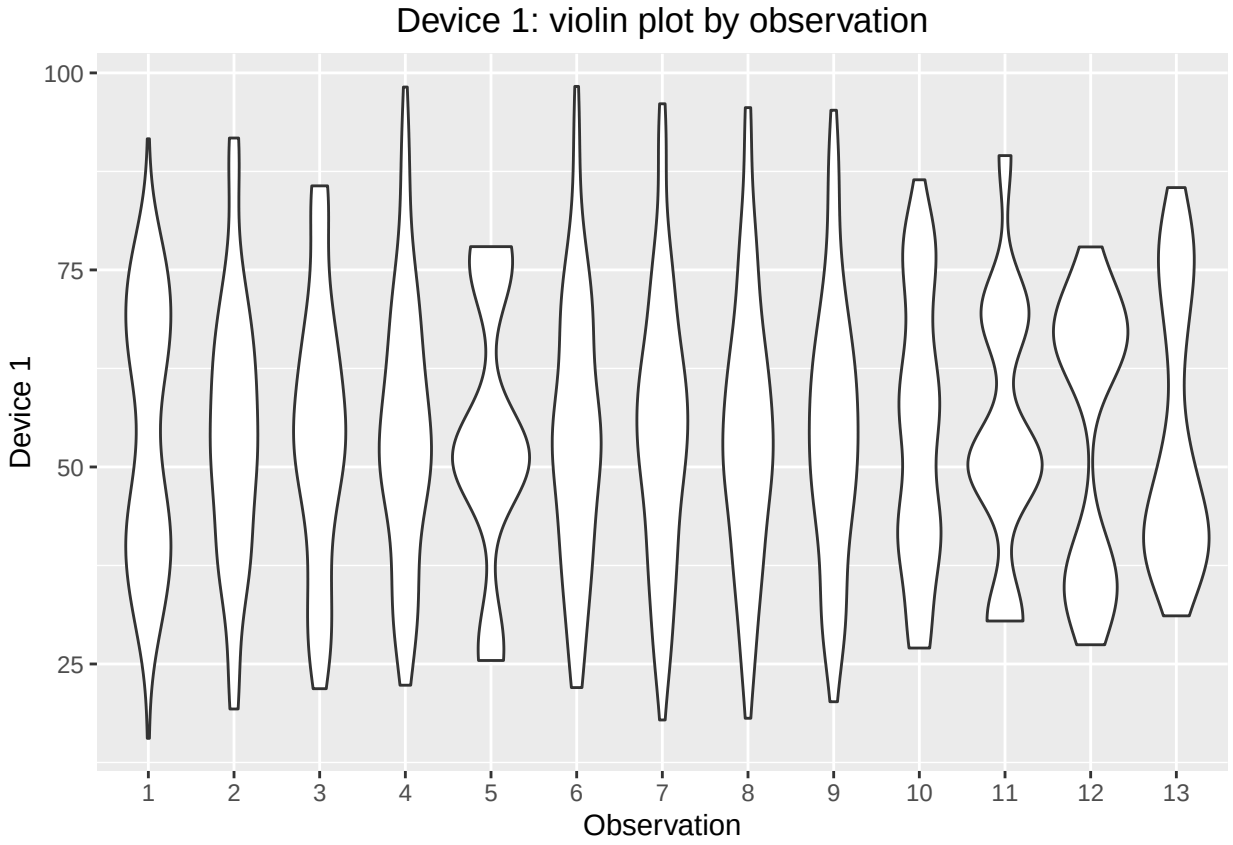

```
boxplot(d2 ~ obs, data = dt5, col = c(1:13), main = "Device 2: boxplot for each observiation")
```



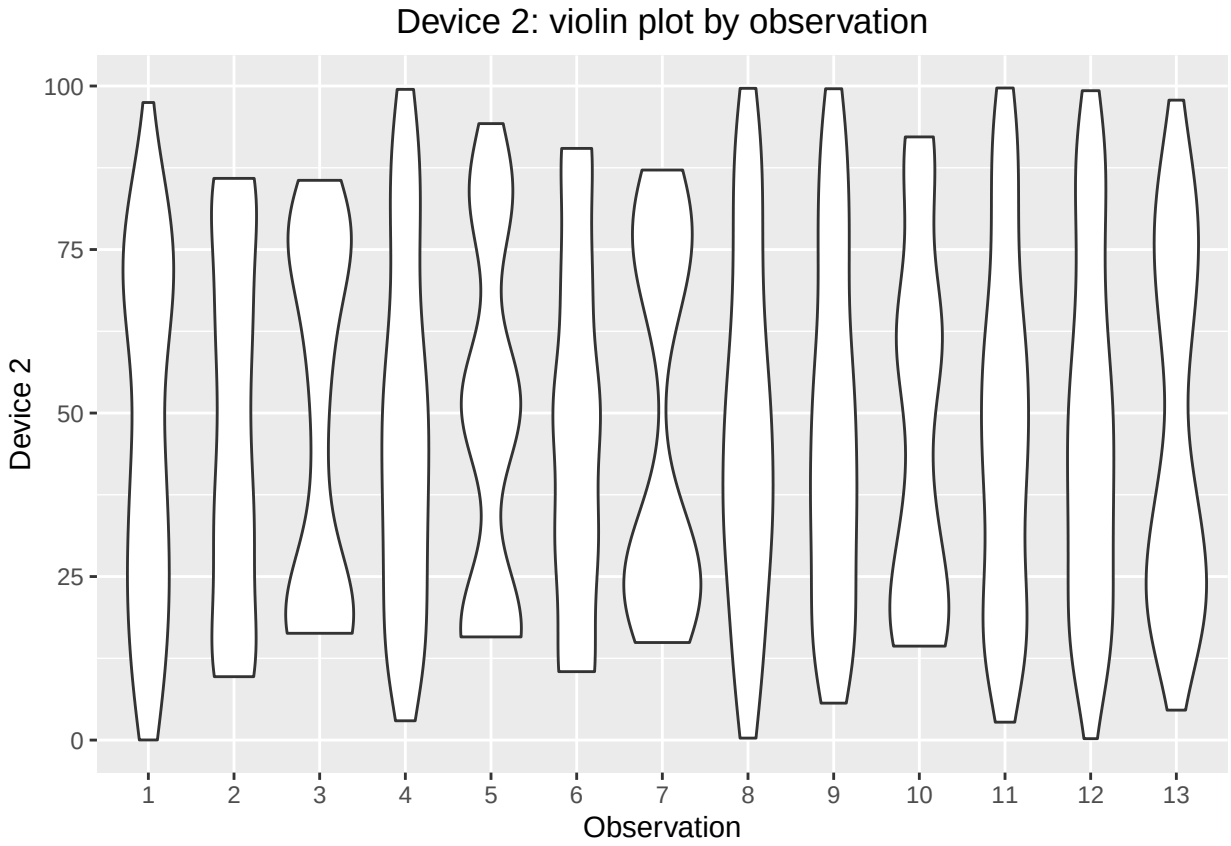
### c. A violin plot of dev by Observer

We could see that the violin plots reveal more underlying data mechanisms compare to the summary table and boxplots. We could notice that the distributions the device have several types, some are normal and some are not.

```
dt5 %>% ggplot() +
  geom_violin(mapping = aes(factor(obs), d1)) +
  ggtitle("Device 1: violin plot by observation") +
  theme(plot.title = element_text(hjust=0.5)) +
  xlab("Observation") +
  ylab("Device 1")
```



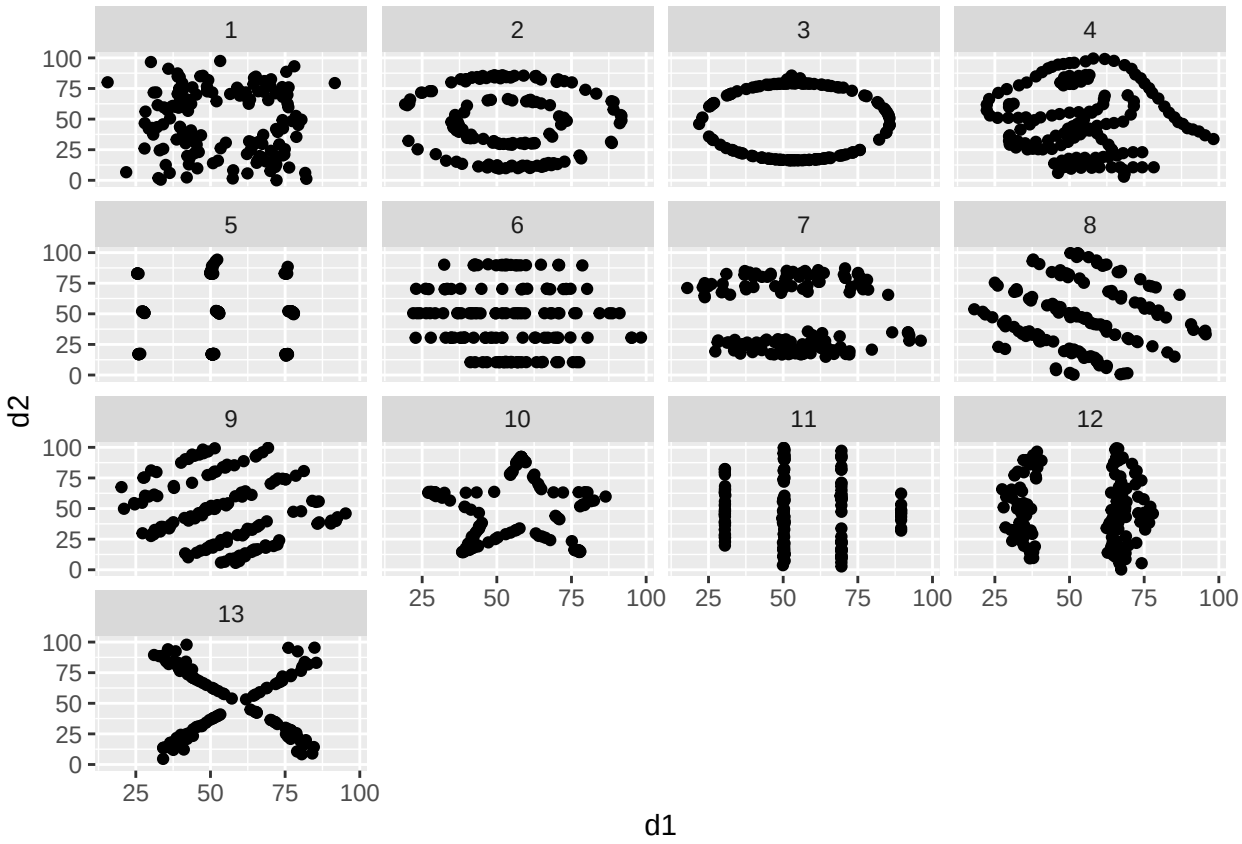

```
dt5 %>% ggplot() +
  geom_violin(mapping = aes(factor(obs), d2)) +
  ggtitle("Device 2: violin plot by observation") +
  theme(plot.title = element_text(hjust=0.5)) +
  xlab("Observation") +
  ylab("Device 2")
```




**d. a scatter plot of the data using ggplot, geom_points, and add facet_wrap on Observer.**

The scatterplot reveals more information and the 13 plots show different patterns. Exploratory data analysis plays an important role for the further statistical inference and hypothesis testing. We should try multiple different techniques to examine the underlying data.

```
ggplot(dt5, aes(x=d1,y=d2)) + geom_point() + facet_wrap(~obs)
```




## Problem 6

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

The slice width smaller than $1e^{-6}$ necessary to obtain an answer with small errors.

```
# Compute the area for the integral using for loop
compute_integral <- function(width) {
  x <- seq(0, 1, width)
  s <- c()
  for (i in 1:length(x)){
    s[i] <- exp(-x[i]^2/2)*width
  }
  result <- sum(s)
  return(result)
}

# Compare the result with the analytical solution
wid <- c(0.2,0.02,0.01,10^-3,10^-4,10^-5,10^-6,10^-7)
for (i in 1:length(wid)){
  errors <- abs(sqrt(2*pi)*(pnorm(1)-pnorm(0)) - compute_integral(wid[i]))
  output <- cat("Width: ", wid[i], "Intergral: ", compute_integral(wid[i]), "Error: ", errors, "\n")
}
```

```
## Width:  0.2 Intergral:  1.014253 Error:  0.1586286
## Width:  0.02 Intergral:  0.8716695 Error:  0.01604509
```

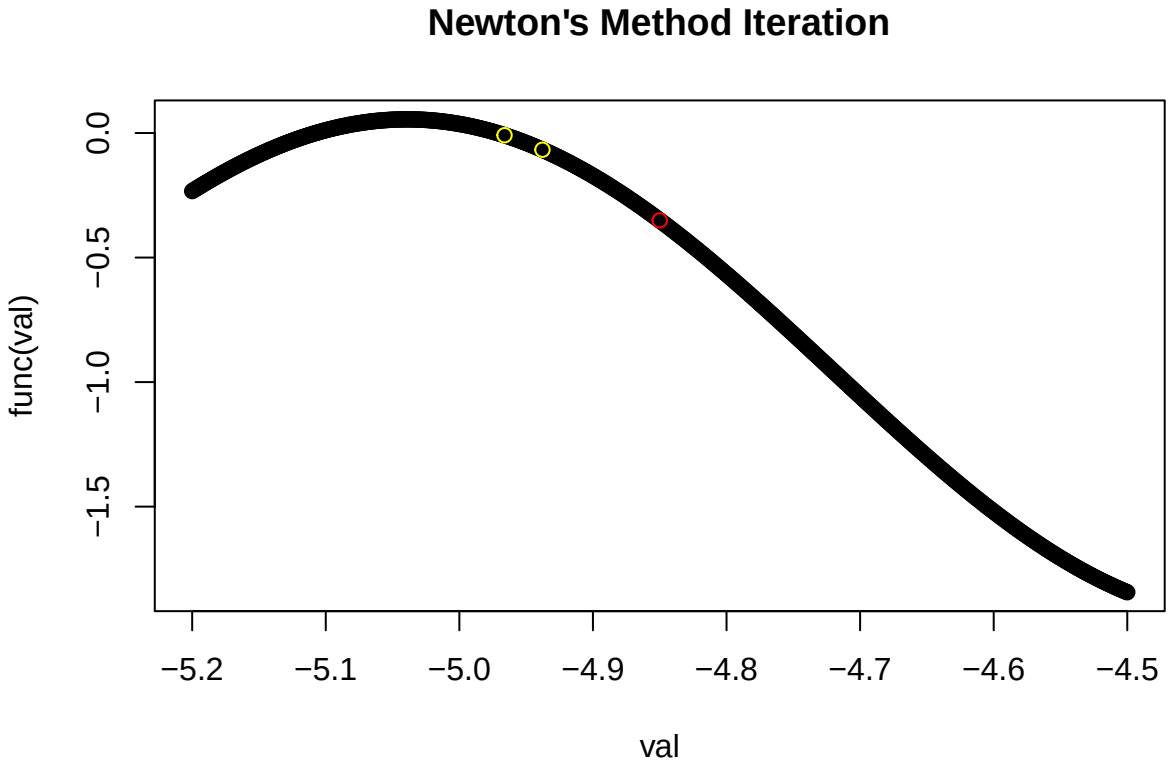
```
## Width:  0.01 Intergral:  0.863652 Error:  0.008027599
## Width:  0.001 Intergral:  0.8564276 Error:  0.0008032148
## Width:  1e-04 Intergral:  0.8557047 Error:  8.032603e-05
## Width:  1e-05 Intergral:  0.8556324 Error:  8.032648e-06
## Width:  1e-06 Intergral:  0.8556252 Error:  8.032653e-07
## Width:  1e-07 Intergral:  0.8556245 Error:  8.032653e-08
```

## Problem 7

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

Tolerance used: 0.001 Interval used: (-5.2, -4.5) Final Answer: -4.966185 Function Value:-0.008897002

```r
# Function 3^x - sin(x) + cos(5*x)
func <- function(x){
  fc <- 3^x - sin(x) + cos(5*x)
  return(fc)
}
# Derivative of the function
deri <- function(x){
  dv <- 3^x*log(3) -cos(x) - 5*sin(5*x)
  return(dv)
}
# Newton method
newton_method <- function(interval, tollerence){
  min <- min(interval)
  max <- max(interval)
  start_point <- mean(interval)
  val <- seq(min, max, 0.001)
  plot(val, func(val), main = "Newton's Method Iteration")
  points(start_point, func(start_point), col = "red")
  start <- start_point
  i <- 1
  while(abs(func(start)) > tollerence){
    new <- start - func(start)/deri(start)
    if (new < min){
      new <- min
    }
    if (new > max){
      new <- max
    }
    start <- new
    points(start, func(start), col = "yellow")
    i =+ 1
  }
  s <- cat("Point: ", start, "Value: ", func(start))
  return(s)
}
# Test newton method
newton_method(c(-5.2, -4.5), 0.01)
```




```
## Point:  -4.966185 Value:  -0.008897002
```

```
## NULL
```

### Problem 8

```
X <- cbind(rep(1,100),rep.int(1:10,time=10))
beta <- c(4,5)
y <- X%*%beta + rnorm(100)
```

```
library("microbenchmark")
# Set seed for reproducible study
set.seed(20202020)
X <- cbind(rep(1,100),rep.int(1:10,time=10))
beta <- c(4,5)
y <- X%*%beta + rnorm(100)
# Compute the mean and 1 matrix
mean_y <- mean(y)
matrix_one <- rep(1, 100)
# accumulating values in a for loop
loop_sum <- function(){
  SST <- 0
  for (i in 1:length(y)){
    SST =+ (mean_y[i] - mean_y)^2
  }
  return(SST)
}

# using matrix to calculate sum
```

```r
matrix_sum <- function() {
  SST <- (t(y) - mean_y*matrix_one) %*% (y - mean_y*matrix_one)
  return(SST)
}

microbenchmark(loop_sum, times = 50)
```

```
## Unit: nanoseconds
##      expr min lq  mean median uq  max neval
##  loop_sum  27 27 87.22     28 28 2957    50
```

```r
microbenchmark(matrix_sum, times = 50)
```

```
## Unit: nanoseconds
##        expr min lq  mean median uq  max neval
##  matrix_sum  27 27 96.68     28 29 3384    50
```

## Problem 9

Finish this homework by pushing your changes to your repo.

**Only submit the .Rmd and .pdf solution files. Names should be formatted HW3_pid.Rmd and HW3_pid.pdf**