HW3-xliu96

Xueying Liu

9/27/2020

```
library(knitr)
library(ggplot2)
library(Deriv)
library(microbenchmark)
library(stats)
library(dplyr)
```

Problem 3

Although I understand the importance of programming style, I do not know what's rule to follow before. There are a few points I am inspired:

- strive for names that are concise and meaningful
- strive to limit code to 80 characters per line
- use explicit returns with *return*() function

Problem 5

a

```
HW3_data=readRDS("HW3_data.rds")
```

Create a function to get a summary statistics:

```
data_summary <- function(x){
  # calculate the mean of col1
  mean1 <- mean(x[,1])
  # calculate the mean of col2
  mean2 <- mean(x[,2])
  # statdard dev of col1
  sd1 <- sd(x[,1])
  # standard dev of col2
  sd2 <- sd(x[,2])
  # correlation between col1 and 2
  corr <- cor(x[,1],x[,2])
  return(c(mean1,mean2,sd1,sd2,corr))
}
```

Loop through the observers:

```
n=13
observer_summary=matrix(NA,nrow=n,ncol=5)
```

```
for(i in 1:n){
  observer=data_summary(HW3_data[HW3_data$Observer==i,2:3])
  observer_summary[i,] =observer
}
colnames(observer_summary)=c("mean of col1","mean of col2","sd of col1","sd of col2","correlation")
row.names(observer_summary) = c(1,2,3,4,5,6,7,8,9,0,11,12,13)
kable(observer_summary,caption = "Table of the means, sd, and correlation for each of the 13 Observers"
```

Table 1: Table of the means, sd, and correlation for each of the 13 Observers

	mean of col1	mean of col2	sd of col1	sd of col2	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
0	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

From the table, we can see that there is no significance difference in the mean, standard deviation of device 1 and device 2 and the correlation between device 1 and device 2 between observers. We can also verify this by the *group_by*() function in the *dplyr* package.

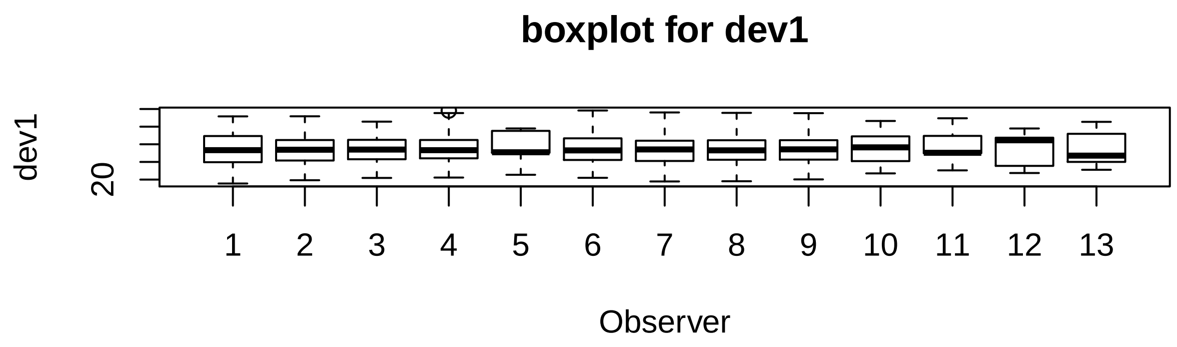
```
by_observer <- HW3_data %>% group_by(Observer)
by_observer %>% summarise(
  mean1 = mean(dev1),
  mean2 = mean(dev2),
  sd1 = sd(dev1),
  sd2 = sd(dev2),
  correlation = cor(dev1,dev2)
)
```

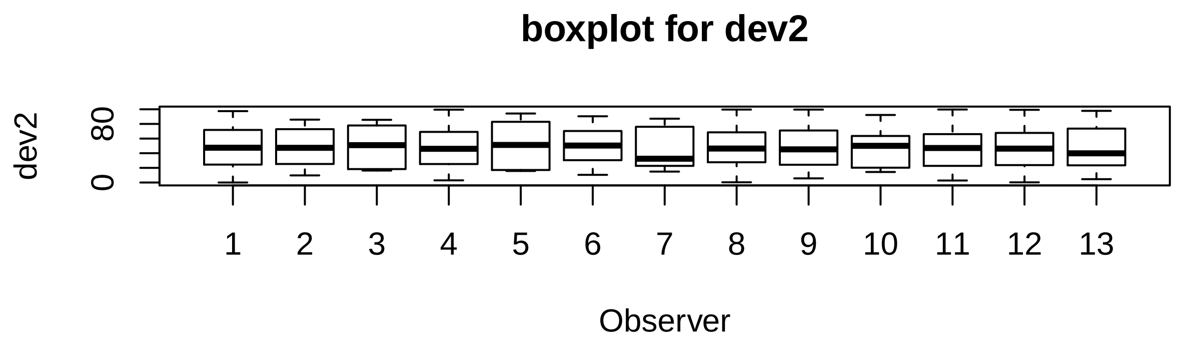
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # 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
```

b

Then we draw the boxplot of devices by observer:

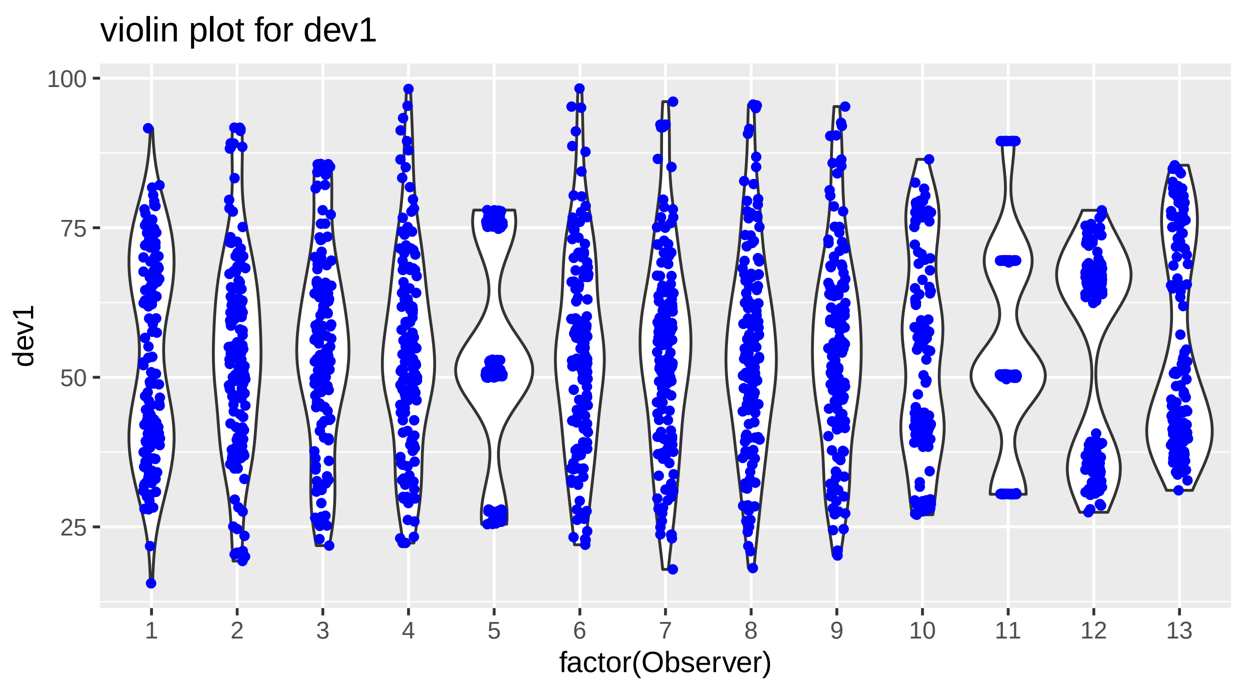


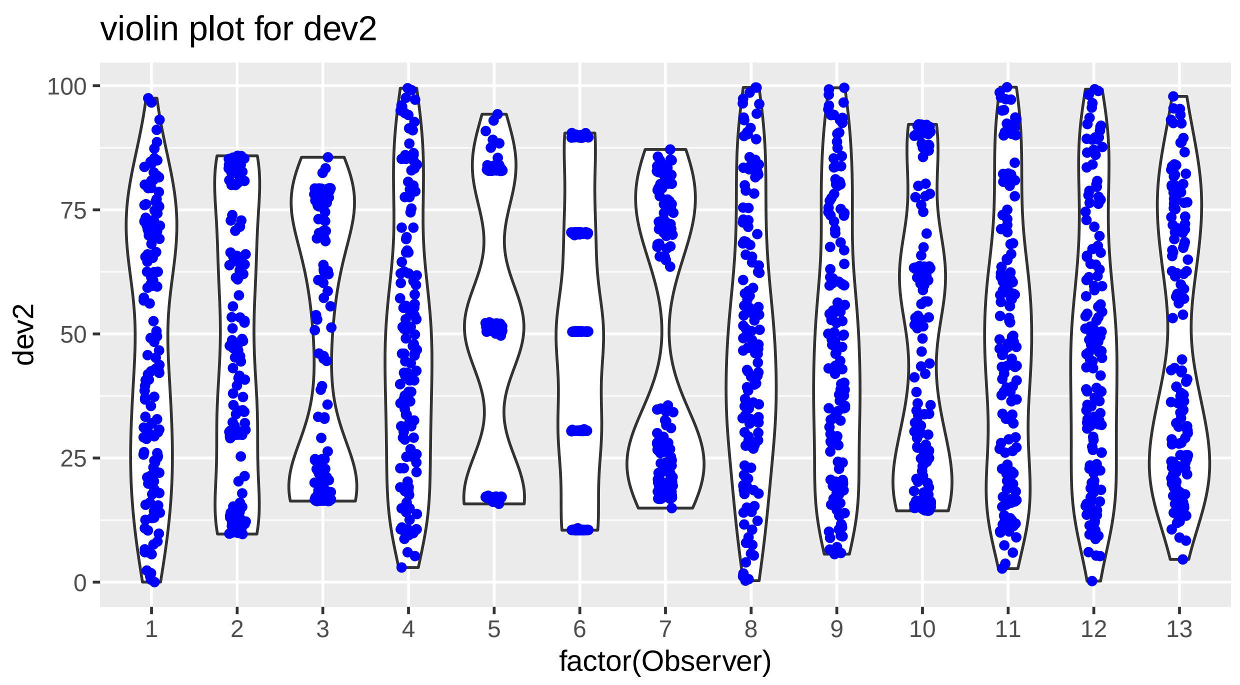


From these plot we have new finding that although the mean and standard variance among different observers is similar, the median and interquartile range are different. For observations in device 1, the observer 12 has much higher median than others, and for observations in the device 2, the observer 7 has much lower median than others.

c

We can also show these in the violin plot:

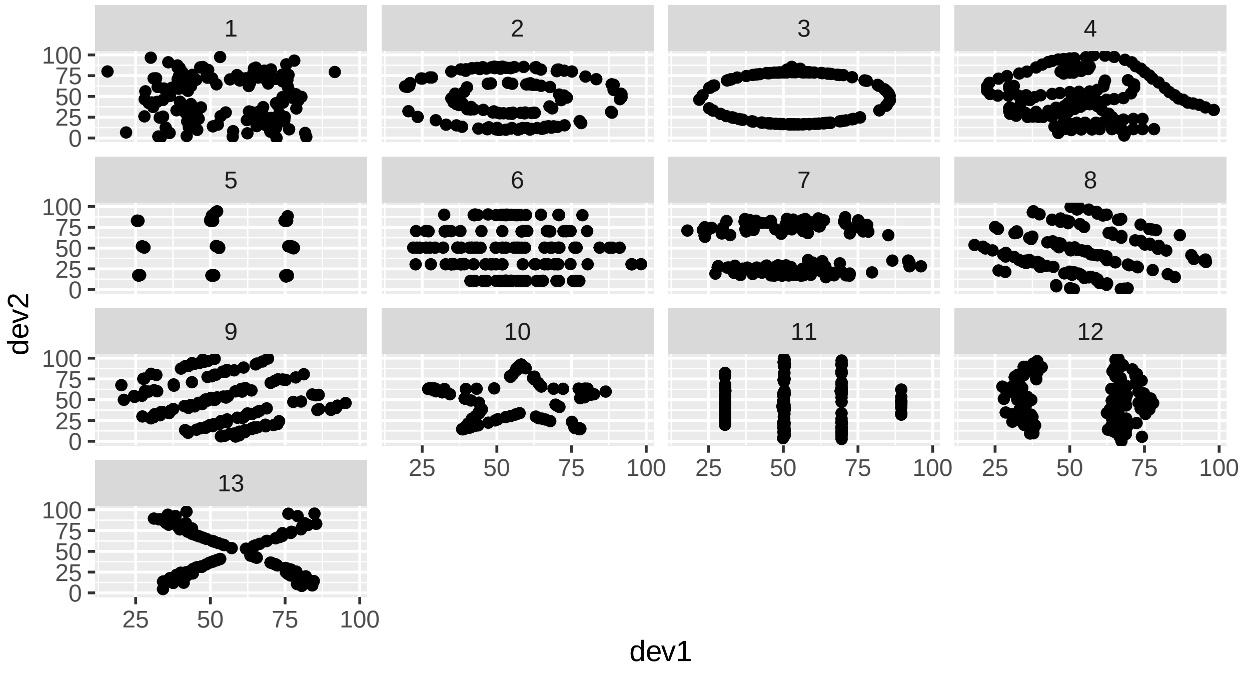




Compared with boxplot and summary statistics, we can conclude that mean, standard deviation and median cannot indicate all the characters of data, especially when the distribution of data matters. We need more plots and statistics to explore the data.

d

```r
ggplot(HW3_data, aes(x=dev1,y=dev2)) + geom_point() + facet_wrap(Observer~.)
```



The plots of dev2 v.s. dev1 by observer indicate that these observations may just be the casual paints of observers, which do not have specific statistical meaning. Therefore, we should pay attention to the raw data and should get more information about the data collected instead of analyse it without knowing the background.

Problem 6

In this problem, we create a function that uses Riemann sums to approximate the integral $f(x) = \int_0^1 e^{-\frac{x^2}{2}}$:

```r
f = function(x) exp(-(x)^2/2)
riemann_sum <- function(d){
  fsum <- 0
  x <- 0
  a <- 0
  b <- 1
  # number of rectangles under the curve
  n <- (b-a)/d
  for(i in 1:n){
    fsum <- fsum + f(x)*d
    x <- x + d
    i <- i +1
  }
  return(fsum)
}
```

Now use a looping construct (for or while) to loop through possible slice widths.

```r
dseq <- seq(0,0.01,0.00001)
#do not use dseq[1] since the width cannot be 0
i <- 1
value <- NULL
while(i < 0.01/0.00001-1){
  i <- i+1
  value <- c(value,riemann_sum(dseq[i]))
}
riemann_width <- data.frame(cbind(dseq[-1],value))
colnames(riemann_width) <- c("width","Riemann_Sum")
```

To obtain an answer within 1e-6 of the analytical solution:

```
# getting the solution using r function
integrate(f,0,1)
```

```
## 0.8556244 with absolute error < 9.5e-15
```

```
analytic_solution = 0.8556244

# obtain an answer within 1e-6 of the analytical solution
for(i in 1:nrow(riemann_width)){
  if(abs(riemann_width$Riemann_Sum[i] - analytic_solution) < 10^-6) {
    print(riemann_width[i,])
  }
}
```

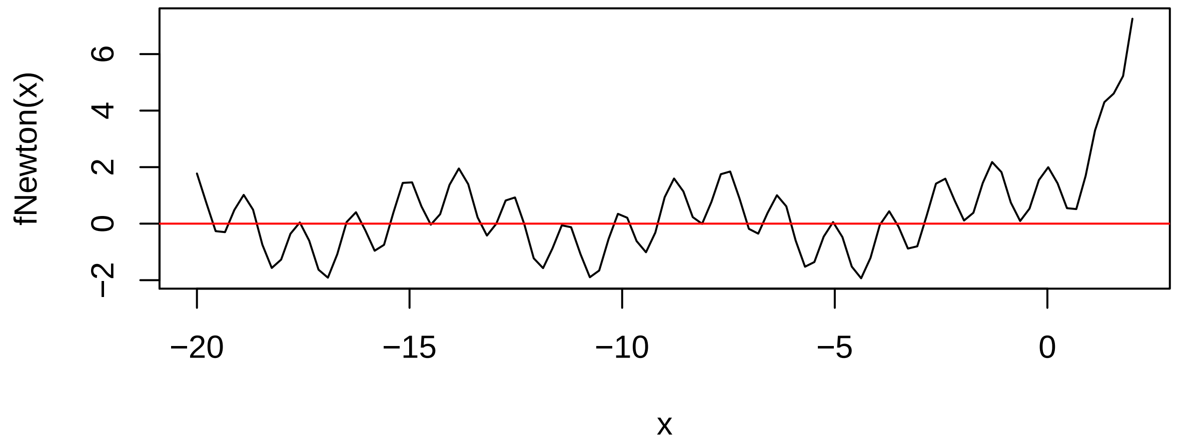
```
##   width Riemann_Sum
## 3 3e-05   0.8556242
##      width Riemann_Sum
## 12 0.00012   0.8556237
```

After we tried a sequence of width from 0 to 0.01 by 0.00001, we found that only when width equals 0.00003 and 0.00012 can we obtain an answer within 1e-6 of the analytical solution 0.8556244.

Problem 7

To find solutions to $f(x) = 3^x - sin(x) + cos(5x)$ using Newton's method, we first show the plot of this function.

```
fNewton <- function(x) 3^x - sin(x) + cos(5*x)
curve(fNewton,from = -20, to = 2)
abline(h=0,col="red")
```



From the plot we can see that the function is approximately periodic when x<0, therefore, we could only consider the solution between x=-5 and x=0.

```r
# starting from x=-2.5
x <- -2.5
iter <- 1
itervalue <- x
while(iter < 20){
  x = x - fNewton(x)/Deriv(fNewton)(x)
  iter <- iter + 1
  itervalue <- c(itervalue,x)
}
x
```

```
## [1] -3.930114
```

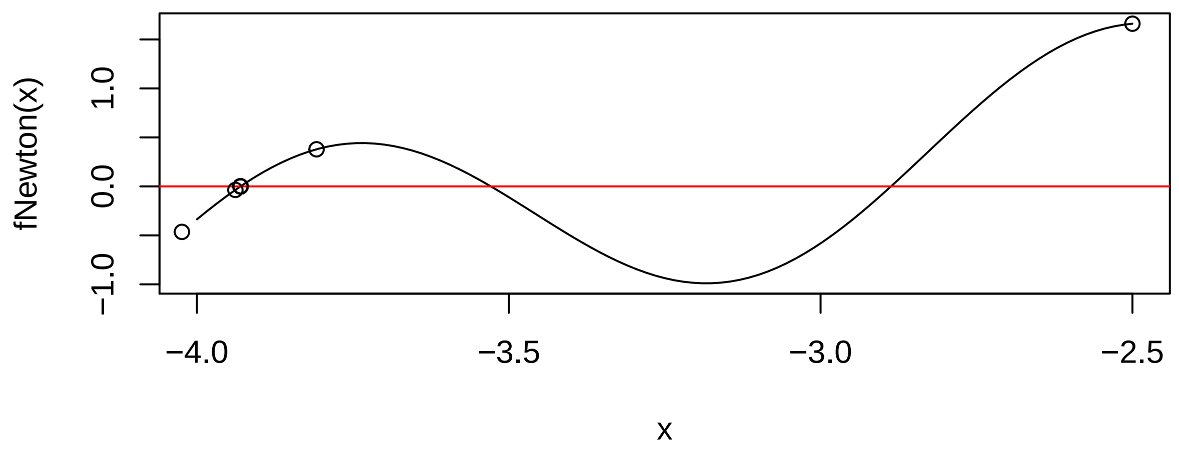
```r
fNewton(x)
```

```
## [1] -1.44329e-15
```

```r
itervalue
```

```
##  [1] -2.500000 -5.574795 -4.257943 -3.808206 -4.024055 -3.938399 -3.930238
##  [8] -3.930114 -3.930114 -3.930114 -3.930114 -3.930114 -3.930114 -3.930114
## [15] -3.930114 -3.930114 -3.930114 -3.930114 -3.930114 -3.930114
```

```r
curve(fNewton,from = -2.5, to = -4)
abline(h=0,col="red")
points(itervalue,fNewton(itervalue))
```



```
# starting from x=-3
x <- -3
iter <- 1
itervalue <- x
while(iter < 20){
  x = x - fNewton(x)/Deriv(fNewton)(x)
  iter <- iter + 1
  itervalue <- c(itervalue,x)
}
x
```

```
## [1] -2.887058
```

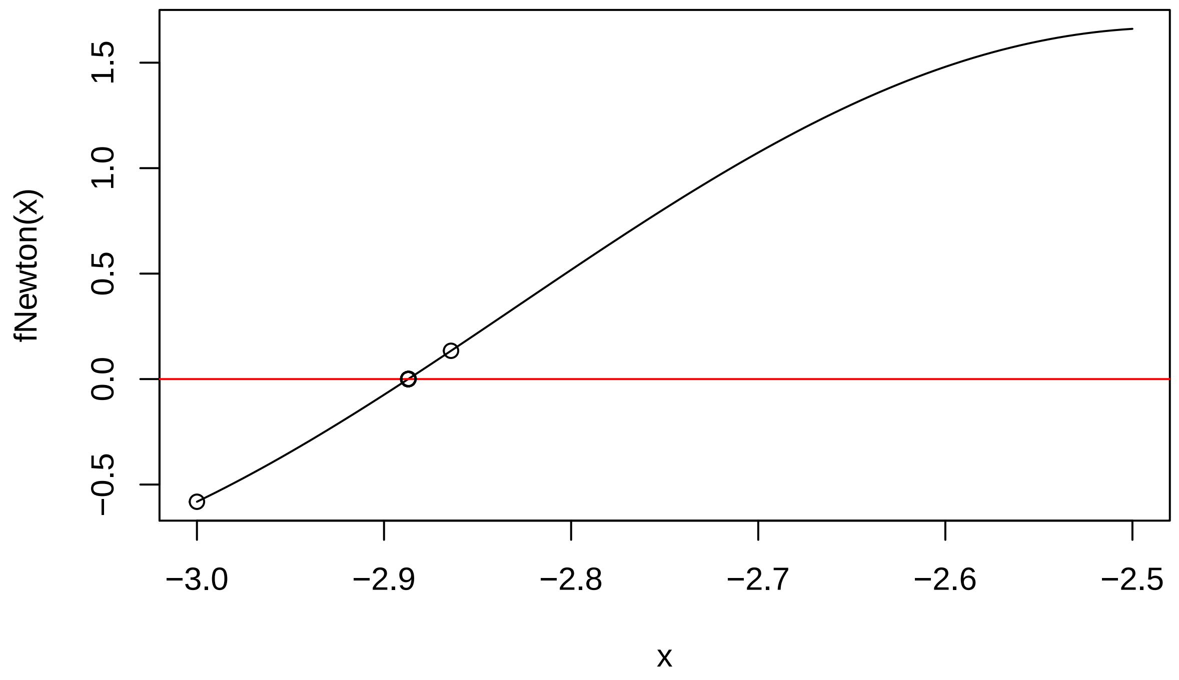
```
fNewton(x)
```

```
## [1] 9.436896e-16
```

```
itervalue
```

```
##  [1] -3.000000 -2.864196 -2.886825 -2.887058 -2.887058 -2.887058 -2.887058
##  [8] -2.887058 -2.887058 -2.887058 -2.887058 -2.887058 -2.887058 -2.887058
## [15] -2.887058 -2.887058 -2.887058 -2.887058 -2.887058 -2.887058
```

```
curve(fNewton,from = -3, to = -2.5)
abline(h=0,col="red")
points(itervalue,fNewton(itervalue))
```



```
# starting from x=-3.5
x <- -3.5
iter <- 1
itervalue <- x
while(iter < 20){
  x = x - fNewton(x)/Deriv(fNewton)(x)
  iter <- iter + 1
  itervalue <- c(itervalue,x)
}
x
```

```
## [1] -3.528723
```

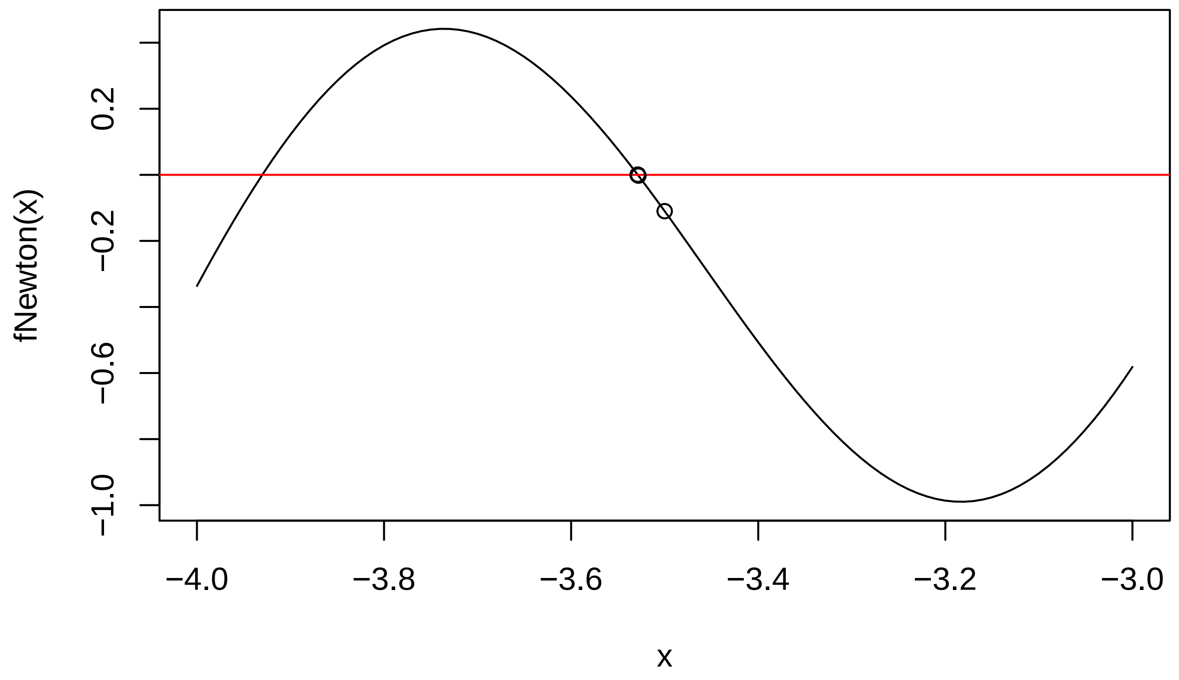
```
fNewton(x)
```

```
## [1] 1.44329e-15
```

```
itervalue
```

```
##  [1] -3.500000 -3.528064 -3.528722 -3.528723 -3.528723 -3.528723 -3.528723
##  [8] -3.528723 -3.528723 -3.528723 -3.528723 -3.528723 -3.528723 -3.528723
## [15] -3.528723 -3.528723 -3.528723 -3.528723 -3.528723 -3.528723
```

```
curve(fNewton,from = -3, to = -4)
abline(h=0,col="red")
points(itervalue,fNewton(itervalue))
```



Therefore, we can find 3 solutions between -5 and 0, which are -3.930114, -2.887058 and -3.528723.

Problem 8

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

a

Calculating the SST using a for loop:

```
SSTloop <- function(y){
SST <- 0
for(i in 1:100){
  SST <- SST + (y[i]-ave)^2
}
return(SST)
}
SSTloop(y)
```

```
## [1] 20722.82
```

```
loop_time <- microbenchmark(SSTloop(y),times = 100, unit = "ms",control=list(warmup=0))
loop_time
```

```
## Unit: milliseconds
##        expr    min     lq       mean median       uq    max neval
##  SSTloop(y) 0.0219 0.0222 0.02272705 0.0223 0.022499 0.0356   100
```

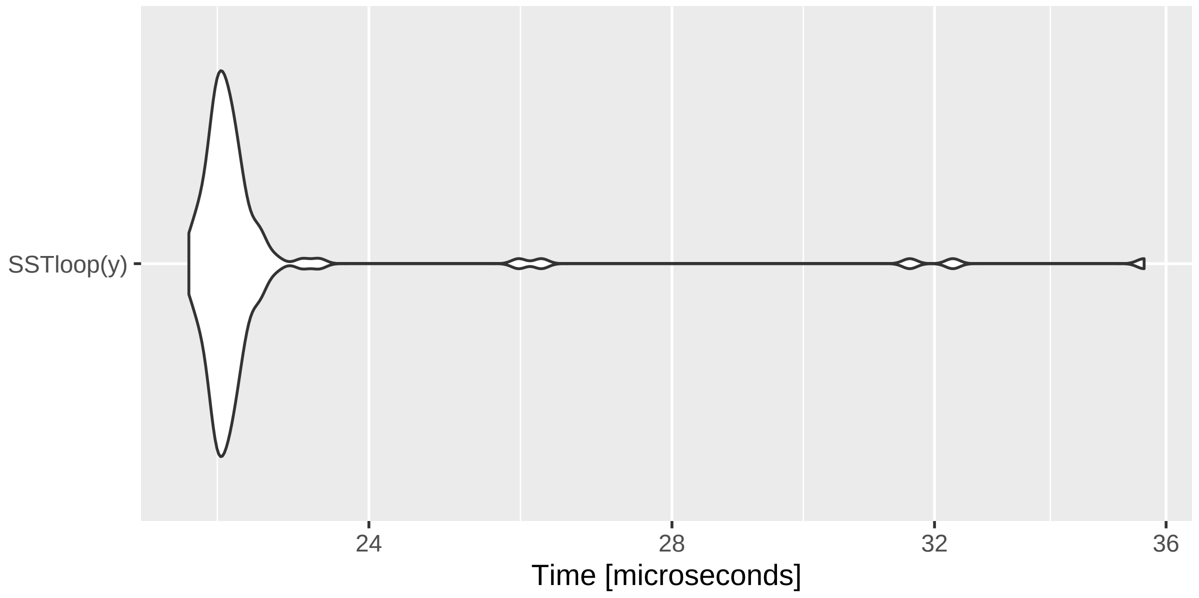
b

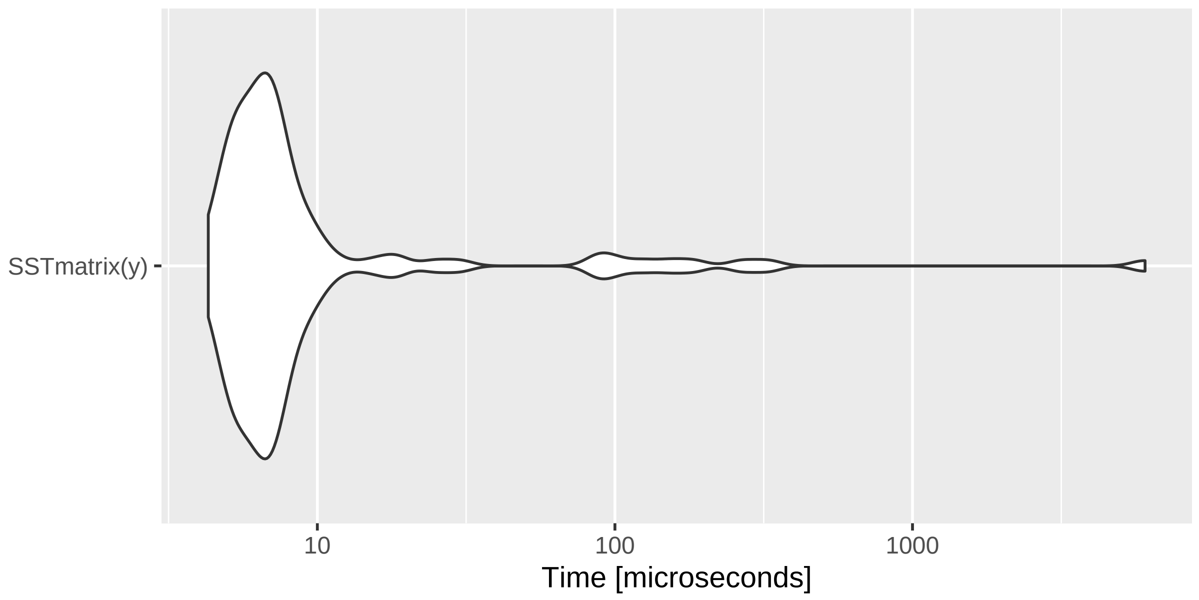
```
SSTmatrix <- function(y){
  return(sum((y-ave)^2))
}
SSTmatrix(y)
```

```
## [1] 20722.82
```

```
matrix_time <- microbenchmark(SSTmatrix(y),times = 100, unit = "ms",control=list(warmup=0))
matrix_time
```

```
## Unit: milliseconds
##          expr    min        lq       mean median        uq    max neval
##  SSTmatrix(y) 0.0043 0.0055505 0.08057395 0.0068 0.0082495 6.0468   100
```




SSTmatrix(y)
10
100
1000
Time [microseconds]