# Hw4

*Jie Min*

*9/24/2019*

## Problem 4

After reading these two links I discovered a good coding style can help others quickly understand what I was doing, which is very important in teamwork. I will check my codes every time after coding and correct my bad coding style.

## Problem 5

```
library(lintr)
lint(filename = "C:\\courses\\rpackage\\STAT_5014_2019_jiem\\HW3_Min.Rmd")
```

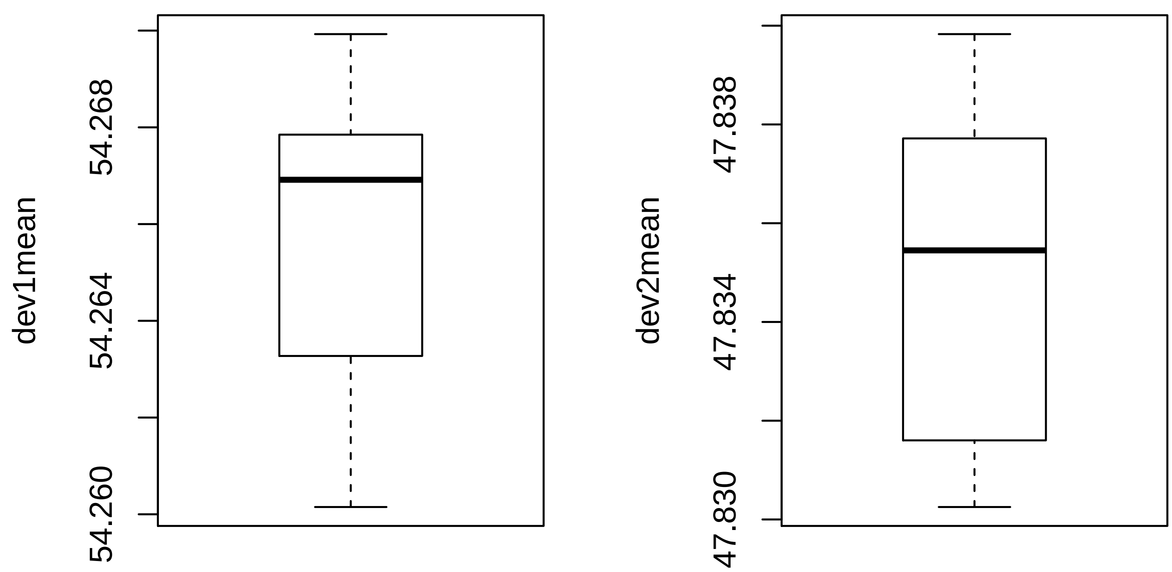
I need to change all single quotes to double quotes, put space after commas, put spaces around infix operators, and use '<-' instead of '=' in coding.

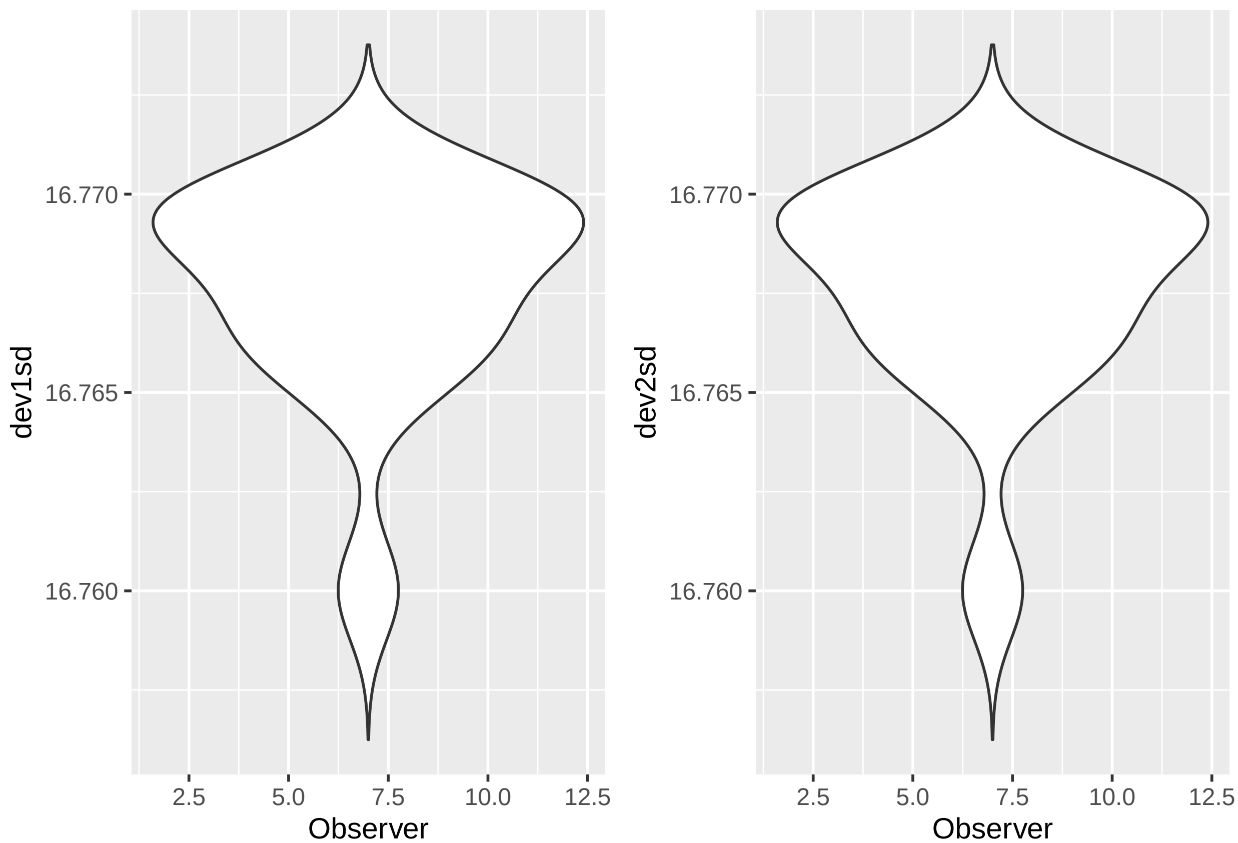
## Problem 6

```
.summary_dev <- function(data){
  re <- NULL
  obs <- unique(data[, 1])
  for(i in 1:max(obs)){

  idx <- which(data[, 1]==i)
  n <- length(idx)
  mean1 <- mean(data[idx, 2])
  mean2 <- mean(data[idx, 3])
  sd1 <- sd(data[idx, 2])
  sd2 <- sd(data[idx, 2])
  corr <- cor(data[idx, 2],data[idx, 3])
  r <- cbind(i, mean1, mean2, sd1, sd2, corr)
  re <- rbind(re, r)

  }
  colnames(re) <- cbind(c("Observer", "dev1mean", "dev2mean", "dev1sd", "dev2sd", "correlation"))
  resut <- as.data.frame(re)
  return(resut)
  }
```

```
##    Observer dev1mean dev2mean   dev1sd   dev2sd correlation
## 1         1 54.26610 47.83472 16.76982 16.76982 -0.06412835
## 2         2 54.26873 47.83082 16.76924 16.76924 -0.06858639
## 3         3 54.26732 47.83772 16.76001 16.76001 -0.06834336
## 4         4 54.26327 47.83225 16.76514 16.76514 -0.06447185
```

```
## 5         5 54.26030 47.83983 16.76774 16.76774 -0.06034144
## 6         6 54.26144 47.83025 16.76590 16.76590 -0.06171484
## 7         7 54.26881 47.83545 16.76670 16.76670 -0.06850422
## 8         8 54.26785 47.83590 16.76676 16.76676 -0.06897974
## 9         9 54.26588 47.83150 16.76885 16.76885 -0.06860921
## 10       10 54.26734 47.83955 16.76896 16.76896 -0.06296110
## 11       11 54.26993 47.83699 16.76996 16.76996 -0.06944557
## 12       12 54.26692 47.83160 16.77000 16.77000 -0.06657523
## 13       13 54.26015 47.83972 16.76996 16.76996 -0.06558334
```






I created side-by-side box plots and violin plots because the absolute value of mean and sd are very different between dev1 and dev2 so if I put two box plots(violin plots) together with same ylim, each box would be too small.

## Problem 7

```
.Reimann_int <- function(width){
    grid <- seq(0, 1, width)
    y <- exp(-(grid^2)/2)
    lefts <- sum(width*y[1:(length(y)-1)])
    rights <- sum(width*y[2:length(y)])
    re <- (lefts+rights)/2
    return(re)
}

width <- c(0.5, 0.1, 0.05, 0.01, 0.005, 0.001)
re <- NULL
for(i in 1:length(width)){
  r <- .Reimann_int(width[i])
  re <- cbind(re, r)
}

true <- 0.855624
err <- 10^(-6)
```

```
er <- abs(true-re)
re <- rbind(re,er)
colnames(re) <- c('0.5', '0.1', '0.05', '0.01', '0.005', '0.001')
rownames(re) <- c('Sum', 'error')
print(re)
```

```
##              0.5          0.1         0.05         0.01        0.005
## Sum   0.84288112 0.8551187811 0.8554980208 8.556193e-01 8.556231e-01
## error 0.01274288 0.0005052189 0.0001259792 4.662547e-06 8.717144e-07
##              0.001
## Sum   8.556243e-01
## error 3.413479e-07
```

```
id <- width[which(er<=err)]
```

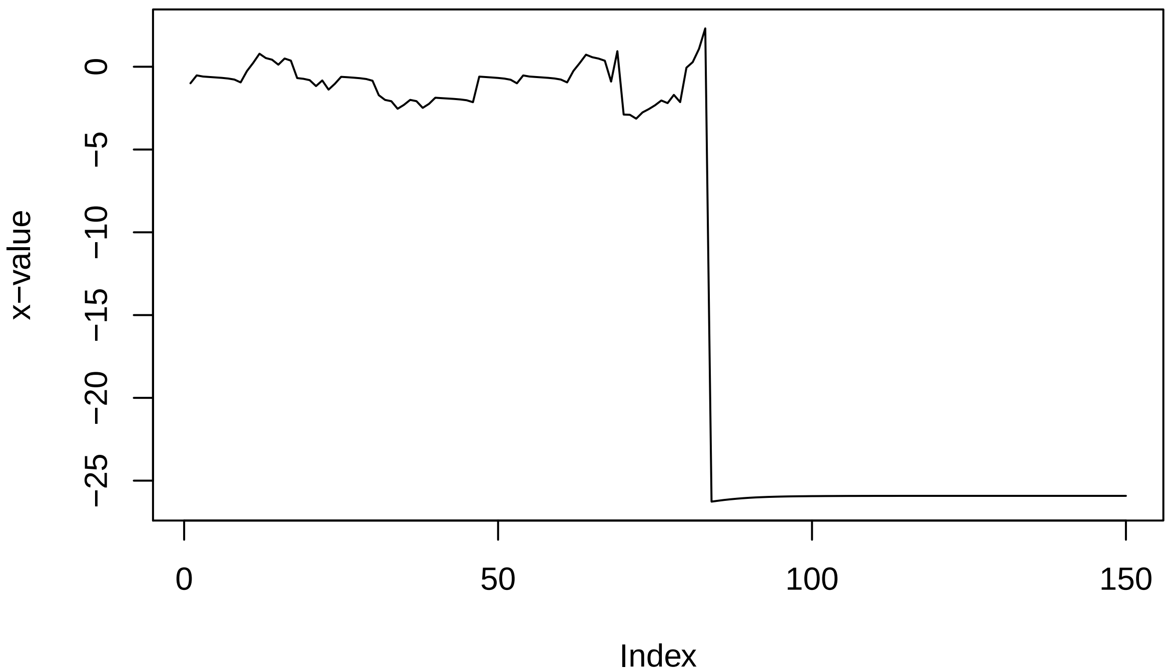
Among my choices, the width that small enough to obtain an answer within 1eˆ-6 of the analytic solution is 0.005 and 0.001.

## Problem 8

$f(x) = 3^x - sin(x) + cos(5x)$

$f'(x) = xlog3 - cox(x) - 5sin(5x)$

```
.find_root <- function(x1, tolerance=10^(-5)){

  error <- 1
  xx <- x1
  while(error>=tolerance){
  x <- x1
  f <- 3^x-sin(x)+cos(5*x)
  fd <- x*log(3)-cos(x)-5*cos(5*x)
  x1 <- x-f/fd
  f1 <- 3^x1-sin(x1)+cos(5*x1)
  error <- abs(f1-0)
  xx <- c(xx,x1)
  }

  return(xx)
}


fx <- .find_root(x1=-1)
```




| final x value | x interval | tolerance |
|---|---|---|
| -25.9181413994994 | [-26.27,2.32] | 1e-05 |

Starting from x=-1, with tolerance = 1e-5, the final x-value and interval is listed in the table above.