HW3_wliu3

Wei Liu

9/29/2020

Problem 1

Finished in the RStudio Cloud

Problem 2

Finished in the RStudio Cloud

Problem 3

The R style guide is very helpful for me. It gives some excellent examples to show how could we make the R code easier to read and share. This is also one part of the reproducible research. For my code works in future, I will pay attention to the naming conventions to make them easy to understand.

Problem 4

Problem 5

a. Summarize the dataset

```
#import the dataset
dat_import <- readRDS("HW3_data.rds")

# the function including summary statistics
summary_stat_caculator <- function(x=dat_import){
  group_by(x, Observer) %>%
  summarize("mean_dev1" = mean(dev1),
            "mean_dev2" = mean(dev2),
            "std_dev1" = sd(dev1),
            "std_dev2" = sd(dev2),
            "correlation_dev1_dev2"=cor(dev1,dev2))}

kable(summary_stat_caculator(dat_import),caption="Summary statistics from 13 Observers")
```

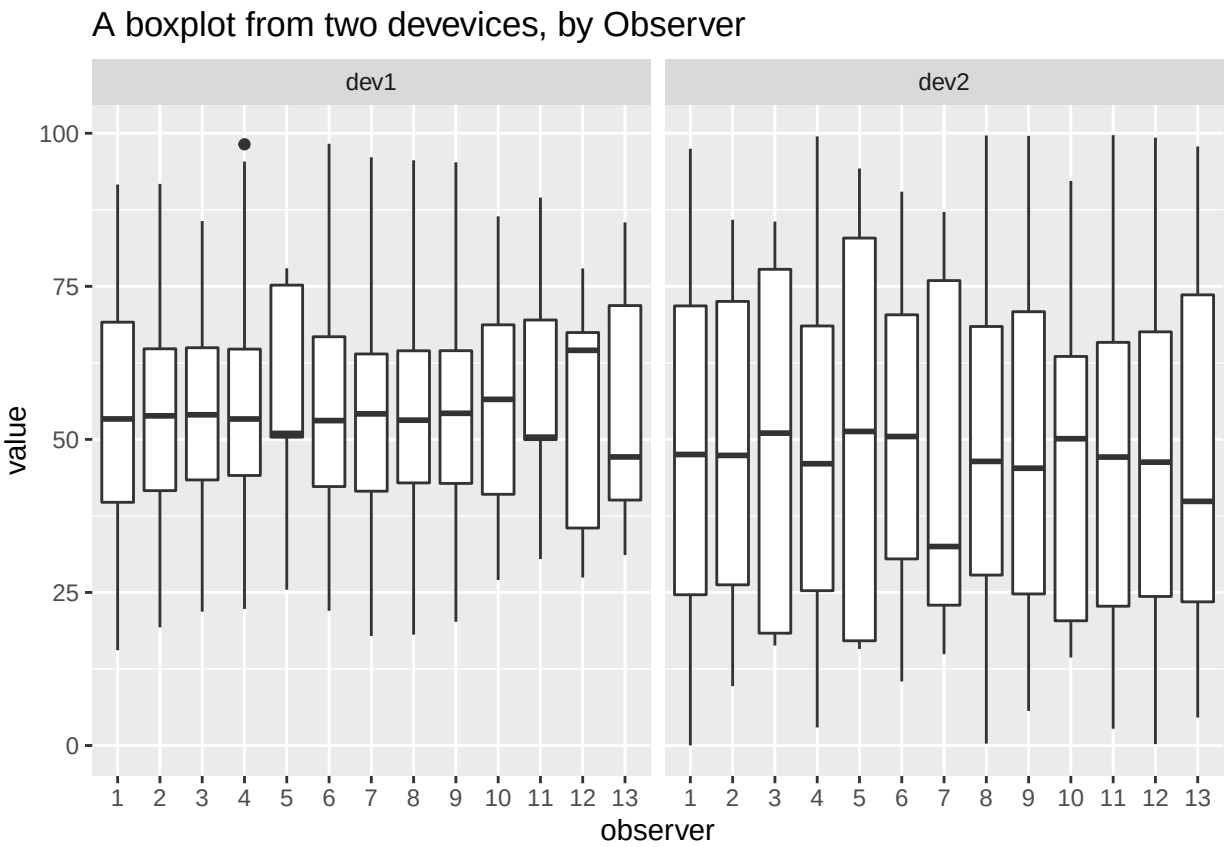
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

Table 1: Summary statistics from 13 Observers

Observer	mean_dev1	mean_dev2	std_dev1	std_dev2	correlation_dev1_dev2
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

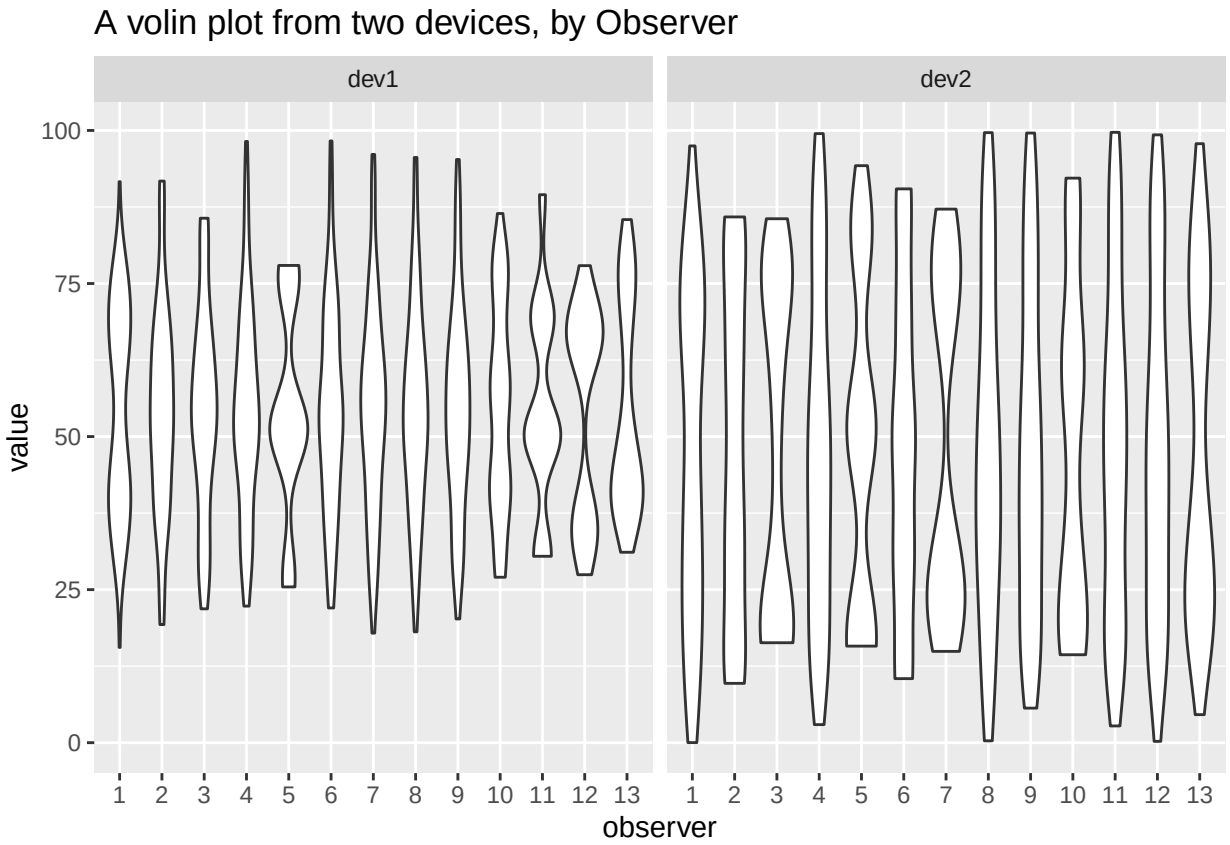
From table 1, it can be found that the means and standard deviations of each device from 13 observers are nearly same. In addition, the correlations between device 1 and 2 from 13 observers are nearly equal to each other.

b. A boxplot of dev, by Observer



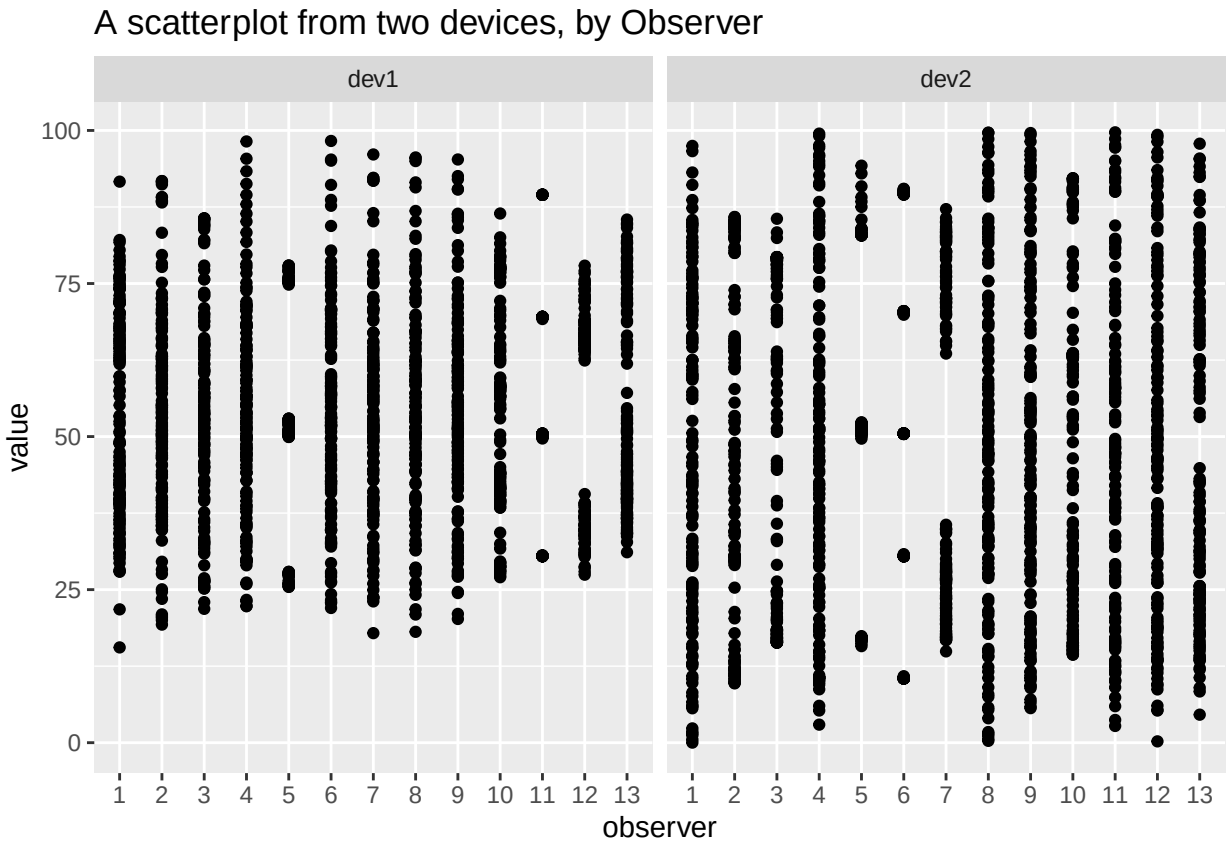
From the boxplots above, we found that the variations and means from device 1 and 2 are different. The mean of device 1 is higher than device 2. The variation of device 2 is higher than device 1. Both of them corresponds to the summary statistics from part a.

c. A volin of dev, by Observer



The violin plots above are similar to boxplots, but the violin plots show the kernel probability density of data based on different observations and device numbers. This gives us some insights about the distribution of data in each group. Such as for observation 12 of dev1, there are more data at the sides; however for observation 10 of dev1, there are more data at the center.

d. A scatter plot of the data



From the scatterplots above, they are consistent with summary statistics. However, there are more information displayed here than summary statistics. For example, for observation 5, 11, 12. The distributions of data are very different with other groups that we cannot find from the summary statistics. Therefore, when we analyze data, we should consider all aspects of data and shouldn't only depend on one or two plots.

Problem 6

```
# analytical solution obtained from the internet
analytical_sol<- 0.8556243918921488

# the function in problem 6
given_function<-function(x){
return(exp(-0.5*x^2))
}

# the integral using reimann sums
integral_function<-function(n){
i=1
n=i
# create x sequence
x=seq(0,1,1/n)
round(x, digits=9)
# midpoint obtained from 1/2(up point+ low point)
```

```
mid_point=0.5*(x[-1]+x[-length(x)])
# integral=summation of the length and width of each slice
integral=sum(1/n*given_function(mid_point))
y=integral
# the integral value should within 1e-6 of the analytical solution
while(abs(y-analytical_sol)>1e-6){
i=i+1
n=i
x=seq(0,1,1/n)
round(x, digits=9)
mid_point=0.5*(x[-1]+x[-length(x)])
integral=sum(1/n*given_function(mid_point))
y=integral
}
return(data.frame("neccessary_slice_width"=1/i,integral))
}
knitr::kable(integral_function(1))
```

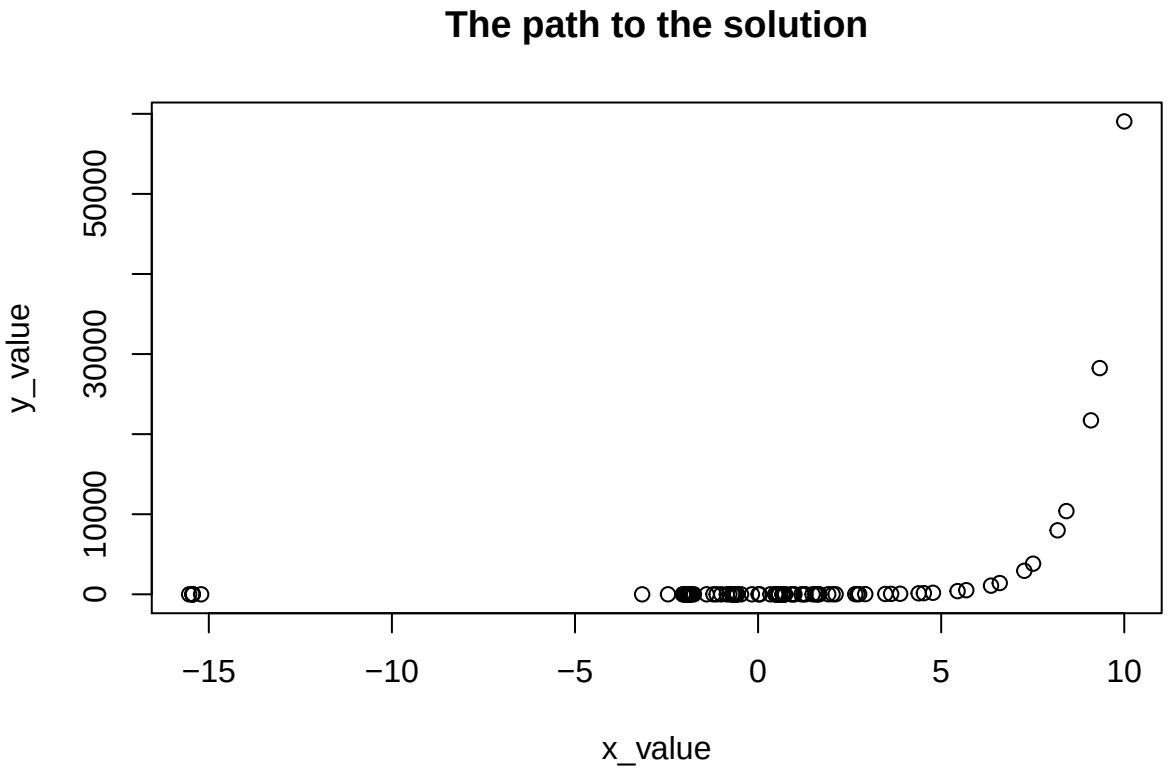
neccessary_slice_width	integral
0.0062893	0.8556254

This table shows the slice width necessary 0.0062893 to obtain an answer within 1e-6 of the analytical solution. The sum calculated is 0.8556254.

Problem 7

```
# set function as seen in problem7
f_p7<-function(x){
  return (3^x - sin(x) + cos(5*x))
}

# set differentiated function as seen in problem7
df_p7 <- function(x){
  return (3^x*log(3)- cos(x) -5*sin(5*x) )
}

# get the points path based on newton's method
newton_m_p7<-function(x_0,tolerance=1e-9){
# x0 is starting x point, y1 is starting y point
y1<- f_p7(x_0)
y_result <- f_p7(x_0)
x1<- x_0
i<-1
# set tolerance for y_result
while(abs(y_result)>tolerance){
# loop i using newton's method
x_begin <- x1[i]
x_next <- x_begin - f_p7(x_begin)/df_p7(x_begin)
i <- i+1
x1[i]<- x_next
```

```
y1[i] <- f_p7(x_next)
y_result <- y1[i]
}
# return data frame of x and y value
return(data.frame(x1,y1))
}

# assign the x_value and y_value
x_value<-newton_m_p7(10)[,1]
y_value<-newton_m_p7(10)[,2]

# draw the points path
plot(x_value,y_value,main = "The path to the solution")
```



The tolerance used to terminate the loop is 1e-9, the interval used is abs(f(x))>1e-9.

Problem 8

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

# get mean of y
y_m<-mean(y)
sst_y<-rep(0,100)
```

```r
# accumulating values in a for loop
loop_sst=function(x){
for (i in 1:100){
sst_y[i]<-(y[i]-y_m)^2
}
return(sum(sst_y))
}
```

```r
matrix_sst<-t(y-y_m) %*% (y-y_m)
```

Therefore, the results from the loop method is 2.0957587×10^4, and the matrix method is 2.0957587×10^4.The timing of the methods are shown as follows:

```
## Unit: milliseconds
##                                 expr      min       lq       mean    median
##              result1 <- loop_sst(y) 0.014937 0.015327 0.01561704 0.0154525
##  result2 <- t(y - y_m) %*% (y - y_m) 0.003392 0.003568 0.00405772 0.0037145
##        uq      max neval
##  0.015615 0.029517   100
##  0.003980 0.020358   100
```