Homework3

Peng Xu

2017/9/16

Problem 4

These two guides introduce basic aspects of codint style, including naming, syntax, and organization. A lot of deatails are introduced for R programming. With good coding style, the codes could be read, modified, and shared easily. As to me, the good coding style lies in the uniformity of code. For example, I will name the variable with the same format. If one third programmer open it, the contents could be understood easily. The best way to improve it is still practicing coding under these guides.

Problem 5

With this function, it is surprised of me to see these hundreds of suggestions. Most of them suggest adding spaces at certain positions to make the code clear. Some other tips, such as line length and double quote, are also mentioned.

Problem 6

The statistics of two devices are summarized as the table below.

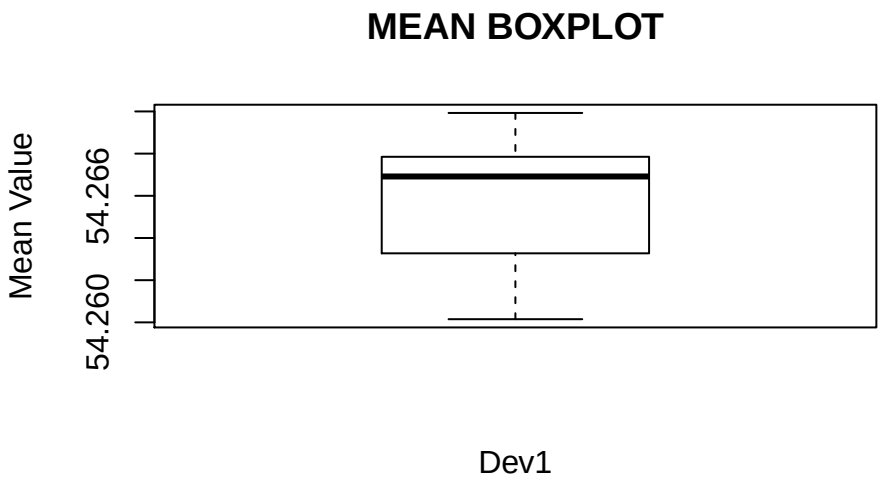
```
knitr::kable(FinalResult, caption="13 Observer Summary")
```

Table 1: 13 Observer Summary

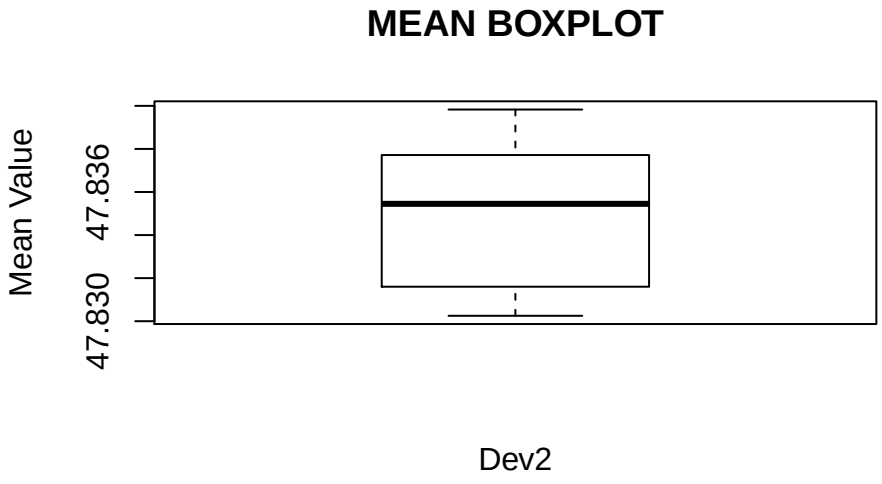
Observer	Mean1	Std1	Mean2	Std2	Cov
1	54.26610	16.76983	47.83472	26.93974	-0.0641284
2	54.26873	16.76924	47.83082	26.93573	-0.0685864
3	54.26732	16.76001	47.83772	26.93004	-0.0683434
4	54.26327	16.76514	47.83225	26.93540	-0.0644719
5	54.26030	16.76774	47.83983	26.93019	-0.0603414
6	54.26144	16.76590	47.83025	26.93988	-0.0617148
7	54.26881	16.76670	47.83545	26.94000	-0.0685042
8	54.26785	16.76676	47.83590	26.93610	-0.0689797
9	54.26588	16.76885	47.83150	26.93861	-0.0686092
10	54.26734	16.76896	47.83955	26.93027	-0.0629611
11	54.26993	16.76996	47.83699	26.93768	-0.0694456
12	54.26692	16.77000	47.83160	26.93790	-0.0665752
13	54.26015	16.76996	47.83972	26.93000	-0.0655833

The boxplot and violin plot are also drawn below.

```
    boxplot(FinalResult$Mean1,
            main= toupper("Mean Boxplot"),
            xlab="Dev1", ylab="Mean Value")
```



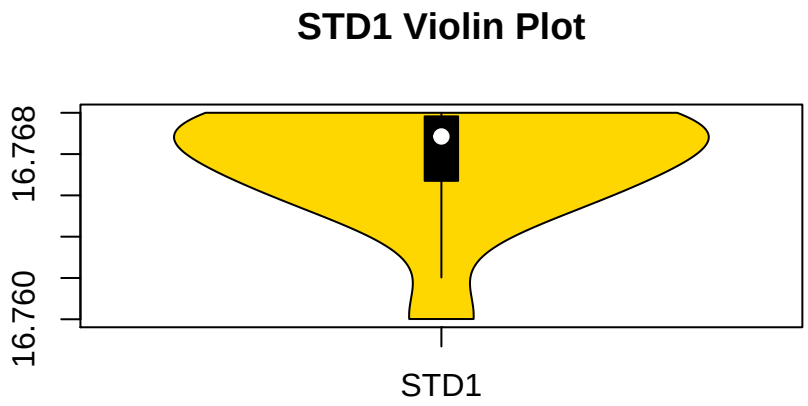

```
    boxplot(FinalResult$Mean2,
            main= toupper("Mean Boxplot"),
            xlab="Dev2", ylab="Mean Value")
```



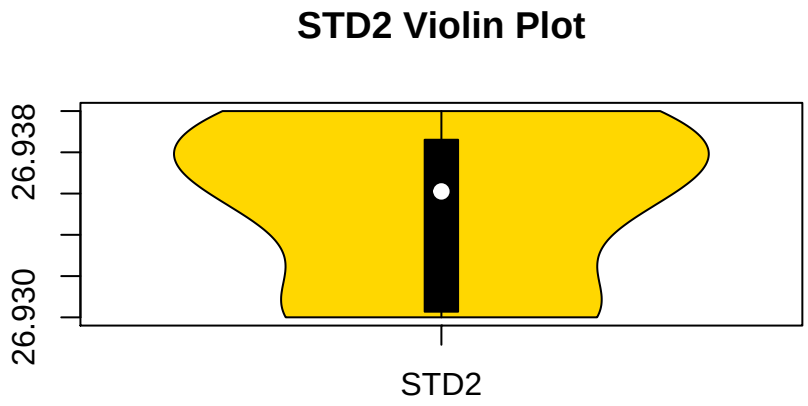

```
library(vioplot)
```

```
## Warning: package 'vioplot' was built under R version 3.3.3
## Loading required package: sm
## Warning: package 'sm' was built under R version 3.3.3
## Package 'sm', version 2.2-5.4: type help(sm) for summary information
```

```
library(sm)

x1 <- FinalResult$Std1

vioplot(x1, names=c("STD1"),
   col="gold")
title("STD1 Violin Plot")
```




```
library(vioplot)
library(sm)

x2 <- FinalResult$Std2

vioplot(x2, names=c("STD2"),
   col="gold")
title("STD2 Violin Plot")
```



Problem 7

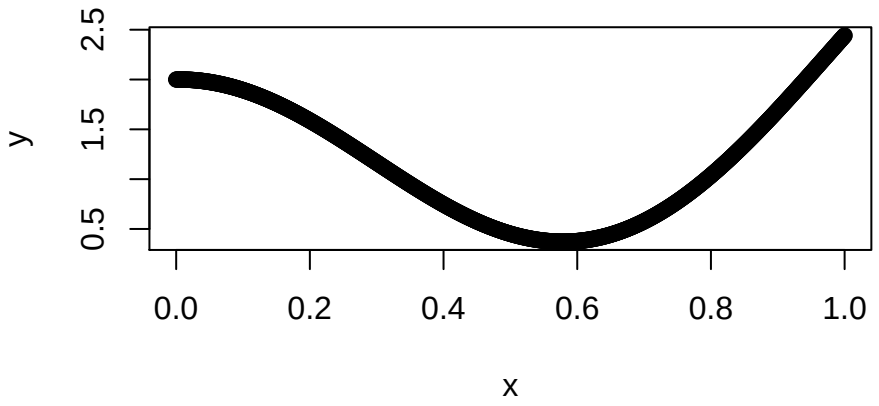
```
knitr::kable(summary(BloodPressure_Tidy), caption="Blood Pressure summary")
```

Table 2: Blood Pressure summary

Day	Approach	Value
Min. : 1	Length:90	Min. :110.8
1st Qu.: 4	Class :character	1st Qu.:125.5
Median : 8	Mode :character	Median :130.4
Mean : 8	NA	Mean :129.0
3rd Qu.:12	NA	3rd Qu.:134.3
Max. :15	NA	Max. :139.6

Problem 8

The function is plotted first for observation.



Using Newton's method, the iteration process could be realized through the while loop.Here the support is set as [0,1] while the precision is defined as 0.001. Then the result of minimum value could be calculated as 0.5169415.

```
solve_func <- function(Min,Max,Tolerance){
    set.seed(1)
    InitValue <- runif(1,min=Min,max=Max)
    tolerance <- Tolerance
    error <- 1
    X1 <- InitValue
    while (error > tolerance){
        X2 <- X1 - f_x(X1)/f_x_deri(X1)/100
        error <- abs(X2-X1)
        X1 <- X2
    }
```

```
    return(X2)
}
```

```
    solve_func(0,1,0.001)
```

```
## [1] 0.5169415
```

Problem 9

Part d

Through the summary of data in 2017, there are 354 makes and 22275 models.

```
#### Summary of Make and Model
total_2017 <- read.table('total_2017')
length(levels(factor(total_2017$Make)))
```

```
## [1] 354
```

```
length(levels(factor(total_2017$Model)))
```

```
## [1] 22275
```

Part e

```
#### Summary of Top 5 Defects and Make/Models
Defect_list <- sort(table(total_2017$Defect_Code),decreasing = TRUE)
Top5_defect <- data.frame(Defect_list[1:5])

Defect_list <- NULL
Target_MakeModel_List <- NULL
for (i in 1:5){
    CertainDefect <- as.character(Top5_defect[i,1])
    Defect_list <- c(Defect_list,CertainDefect)
    temp <- filter(total_2017, Defect_Code_char == CertainDefect)
    MakeModel_List_CertainDefect <- sort(table(temp$Defect_Code),decreasing = TRUE)

    temp_summary_2017 <- temp %>% group_by(Make_Model) %>%
        mutate(makemodel_defect = paste(Make_Model,Defect_Code,sep="_"))

    makemodel_defect_summary <- sort(tapply(temp_summary_2017$Number_of_Defects,
                                            temp_summary_2017$makemodel_defect,sum),
                                            decreasing = TRUE)
    Top5_makemodel_defect <- makemodel_defect_summary[1:5]
    Target_MakeModel <- row.names(Top5_makemodel_defect)[1]
    Target_MakeModel_List <- c(Target_MakeModel_List,Target_MakeModel)
}

Relation <- cbind(Defect_list,Target_MakeModel_List)
colnames(Relation) <- c('Defect','Make/Model')
knitr::kable(Relation, caption="TOP Make/Models of Five Common Defects")
```

Table 3: TOP Make/Models of Five Common Defects

Defect	Make/Model
K04	VOLKSWAGEN/POLO_K04
AC1	VOLKSWAGEN/POLO_AC1
G05	VOLKSWAGEN/POLO_G05
RA2	PEUGEOT/206; 1.4 3DRS_RA2
K05	VOLKSWAGEN/POLO_K05

Part f & g

The lm and aov functions have been attempted to explore the relationship between number of defects and make/models. However, as the value of make/model is not numeric, the software could not give the results.

Part h

This workflow considers all the possible records at first, which lead to high volume of computing capacity. So if the contraint could be applied at first, such as the year 2017, the computation capacity could be reduced a lot.