STATS 506 HW 6

ZHAO Shengchun

Github URL:

https://github.com/ZHAOShengchun67734538/STAT-506-HW-6

Question 1

```
library(DBI)
library(parallel)
library(future)
```

```
Warning: package 'future' was built under R version 4.4.2
```

```
library(data.table)
```

```
Warning: package 'data.table' was built under R version 4.4.2
```

```
# Import the database of the Lahman data
lahman = dbConnect(RSQLite::SQLite(),
                   "C:/Users/z1883/Desktop/lahman_1871-2022.sqlite")
data = dbGetQuery(lahman, "SELECT teamID, PO, A, InnOuts FROM Fielding")
dim(data)
```

```
[1] 149365      4
```

```
class(data)
```

```
[1] "data.frame"
```

```r
head(data)
```

```
  teamID PO A InnOuts
1    SFN  0 0      32
2    CHN  1 5     159
3    CHA  2 4      97
4    BOS  3 6     146
5    SEA  2 5     214
6    SEA  2 3     149
```

```r
# Check and delete the NA values of PO, A, and INNOUT
sum(is.na(data$PO))
```

```
[1] 0
```

```r
sum(is.na(data$A))
```

```
[1] 0
```

```r
sum(is.na(data$InnOuts))
```

```
[1] 29932
```

```r
# Delete the NA values
data = data[!is.na(data$InnOuts), ]

# Check and delete the 0 values of INNOUT
nrow(data[which(data$InnOuts == 0),])
```

```
[1] 230
```

```r
data = data[-which(data$InnOuts == 0),]
dim(data)
```

```
[1] 119203      4
```

(a)

Calculate the average RF for each team in the Fielding table.

```r
# Calculate the average RF for each team in the Fielding table.
data$RF = 3*(data$PO+data$A)/data$InnOuts

### Calculate the estimate RF value ###
mean_RF = aggregate(data$RF,
                    by = list(data$teamID),
                    FUN = mean, na.rm = TRUE)
# Show the first ten highest RF
head(mean_RF[order(mean_RF$x, decreasing = TRUE),], n=10L)
```

```
    Group.1         x
103     RC1 0.5740314
67      LS1 0.5301629
50      ELI 0.5265842
79      MLU 0.5133325
64      KEO 0.5121290
105     RIC 0.5089137
12      BLA 0.4948384
69      LS3 0.4891679
126     TRN 0.4808805
98      PHU 0.4805772
```

```r
# Use data.table to do the bootstrap
rfdata = data.table(data)

#' stratify bootstrap function
#'
#' @param dat the data which have the RF values and team ID
#'
#' @return a stratified bootstrap sample
stra_boostrap = function(dat)
{
  resamp = dat[, .SD[sample(x = .N, size = .N, replace = TRUE)],
                 by = teamID]
  result = resamp[, .(mean(RF, na.rm = TRUE)), by = teamID]
  return(result)
}

# Do a test to verify
# Make sure sampling was done within team ID
d = stra_boostrap(rfdata)
```

```
d = as.data.frame(d)
identical((sort(d$teamID)), (sort(mean_RF$Group.1)))
```

```
[1] TRUE
```

```
# Determine the sample size
size = 1000
```

Because the bootstrap sample is too large, so, in the quarto file, we only show the system time instead of the whole sample!

1, Without using Parallel

```
# Without any parallel processing
system.time({
  s1 = lapply(seq_len(size),
                function(x) stra_boostrap(rfdata))
})
```

```
   user  system elapsed 
  29.87   10.84   34.39 
```

2, Using parLapply

```
# Use parLapply
system.time({
  cl = makeCluster(6)
  clusterEvalQ(cl, library(data.table))
  clusterExport(cl, varlist = c("stra_boostrap", "rfdata"))
  s2 = parLapply(cl, seq_len(size),
                function(x){stra_boostrap(rfdata)})
})
```

```
   user  system elapsed 
   0.11    0.02    8.44 
```

```
stopCluster(cl)
```

3, Using future

```
# Using futures
plan(multisession)
system.time({
  s3 = lapply(seq_len(size),
                function(x) {
                  future(stra_boostrap(rfdata), seed = TRUE)
                })
  s3 = lapply(s3, value)
})
```

```
   user  system elapsed
 102.99    3.66  133.81
```

(b)

```
#' Showing the estimated statistics by decreasing order
#'
#' @param d is the bootstrap sample
#'
#' @return a table showing the estimated RF and
#' associated standard errors
#' for the teams with the 10 highest RF.
stat_table = function(d) {
  sd = rbindlist(d)[, sd(V1), by=teamID][, V1]
  team = rbindlist(d)[, sd(V1), by=teamID][, teamID]
  sd_table = data.frame(Group.1 = team, estimated_sd = sd)
  result = merge(mean_RF,sd_table, by = "Group.1")
  colnames(result) = c("teamID","estimated_mean_RF","estimated_sd_RF")
  result = result[order(result$estimated_mean, decreasing = TRUE),]
  # Only renturn the top 10 RF values
  return(result[1:10,])
}
```

```
# without parallel
stat_table(s1)
```

```
    teamID estimated_mean_RF estimated_sd_RF
103    RC1         0.5740314      0.08217064
67     LS1         0.5301629      0.05619994
50     ELI         0.5265842      0.06162636
79     MLU         0.5133325      0.12589728
```

```
64     KEO          0.5121290      0.11054401
105    RIC          0.5089137      0.06849612
12     BLA          0.4948384      0.03172253
69     LS3          0.4891679      0.01628139
126    TRN          0.4808805      0.02885032
98     PHU          0.4805772      0.04725321
```

```
# parLapply
stat_table(s2)
```

```
    teamID estimated_mean_RF estimated_sd_RF
103    RC1          0.5740314      0.08135337
67     LS1          0.5301629      0.05642742
50     ELI          0.5265842      0.06317419
79     MLU          0.5133325      0.13319222
64     KEO          0.5121290      0.11489213
105    RIC          0.5089137      0.06476120
12     BLA          0.4948384      0.03256799
69     LS3          0.4891679      0.01542090
126    TRN          0.4808805      0.02980728
98     PHU          0.4805772      0.04583875
```

```
# futures
stat_table(s3)
```

```
    teamID estimated_mean_RF estimated_sd_RF
103    RC1          0.5740314      0.08202314
67     LS1          0.5301629      0.05607685
50     ELI          0.5265842      0.06475752
79     MLU          0.5133325      0.12798910
64     KEO          0.5121290      0.11039589
105    RIC          0.5089137      0.06851563
12     BLA          0.4948384      0.03194291
69     LS3          0.4891679      0.01526937
126    TRN          0.4808805      0.02946040
98     PHU          0.4805772      0.04594472
```

(c)

From the system time, we can find the parLapply using the smallest time, which is much much faster than other two ways; the second is the approach "without using parallel", the method "future" consumes the longest time, which may need to some adjust later. All in all, use the parLapply could save you a lot of time when you are dealing with very large data set.