# STATS506-Problem Set 6

Shenyi Tang

2024-12-03

### Shenyi Tang's GitHub Repo For STATS 506 FA 2024

https://github.com/shenyi-tang/stats506-computing-methods-and-tools.git

```r
library(DBI)
library(tidyverse)
library(rsample)
library(parallel)
library(future)
library(furrr)
library(microbenchmark)
```

### Stratified Bootstrapping

```r
lahman <- dbConnect(RSQLite::SQLite(), "lahman_1871-2022.sqlite")
```

**a. Calculate the average RF for each team in the Fielding table. Then, since we don't have a closed form for the standard deviation of this statistic, carry out a stratified bootstrap by team to estimate it. Do this out three ways:**

**1. Without any parallel processing**

**2. Using parallel processing with the 'parallel' package.**

**3. Using futures with the 'future' package.**

**Generate at least 1,000 bootstrapped samples for each approach.**

```r
df <- dbGetQuery(lahman,
                 "select * from fielding")


# generate the new field "RF"
new.df <- df %>% mutate(RF = 3*(PO+A)/InnOuts)

# filter out the null value of InnOuts to get the valid RF value
# generate the new field "RF"
```

```r
new.df.filter <- df %>%
  filter(!is.na(InnOuts)) %>%
  filter(InnOuts > 0) %>%
  mutate(RF = 3*(PO+A)/InnOuts)

# calculate the average RF for each team
avg.RF <- dbGetQuery(lahman,
                     "select
                        teamID
                        , avg(3*(PO+A)/NULLIF(InnOuts, 0)) as avg_rf
                      from fielding
                      where InnOuts > 0
                      group by teamID
                      order by avg_rf desc")

head(avg.RF, 10)
```

```
   teamID    avg_rf
1     MLU 0.2352941
2     BSU 0.2200000
3     LS1 0.1785714
4     RC1 0.1785714
5     SPU 0.1666667
6     PH3 0.1578947
7     BLA 0.1559633
8     KEO 0.1538462
9     BFP 0.1509434
10    ELI 0.1428571
```

```r
set.seed(506)

##' @title do the stratified bootstrap
##' @param data, the original dataset
##' @param strata, within the strata we do the bootstrap
##' @param N, integer, the sample size of the stratified bootstrap
##'
stratified_bootstrap <- function(data, strata = "teamID", N = 1000) {
  rsamples <- replicate(N, {
    # do the stratified bootstrap
    bootstrap_sample <- data %>%
      # to clarify that strata is an arg other than a column name
      group_by(!!sym(strata)) %>%
      sample_n(size = n(), replace = TRUE)

    # calculate the average RF of each team in the bootstrap sample
    bootstrap_sample %>%
      group_by(!!sym(strata)) %>%
      summarise(mean_RF = mean(RF, na.rm = TRUE))
  }, simplify = FALSE
  )
  return(rsamples)
}

# the result of 1000 times bootstrap
sample1 <- stratified_bootstrap(new.df.filter, strata = "teamID", N = 1000)
# bind the 1000 bootstrap results vertically
sample1.df <- as.data.frame(bind_rows(sample1))
# calculate the standard error of the RF of each team
sample1.se <- rownames_to_column(as.data.frame(tapply(sample1.df$mean_RF,
 ↪  sample1.df$teamID, sd, na.rm = TRUE)),
                                 var = "teamID")
colnames(sample1.se) <- c("teamID", "SD")
head(sample1.se)
```

```
  teamID          SD
1    ALT 0.051757912
2    ANA 0.015706299
3    ARI 0.008635939
4    ATL 0.006020428
5    BAL 0.005470144
6    BFN 0.021188571
```

```r
set.seed(506)

##' @title do the stratified bootstrap using parallel passage
##' @param data, data.frame, the original dataset
##' @param strata, string, within the strata we do
##' @param N, int, the size of bootstrap samples
##' @param n_cores, int, the number of cores
##'
parallel_stratified_bootstrap <- function(data, strata, N, n_cores = detectCores()) {
  cl <- makeCluster(n_cores/2)

  on.exit(stopCluster(cl))
  clusterEvalQ(cl, library(tidyverse))


  rsamples <- parLapply(cl, 1:N, function(i, data, strata){
    bootstrap_sample <- data %>%
      group_by_at(vars(all_of(strata))) %>%
      sample_n(size = n(), replace = TRUE)

    bootstrap_sample %>%
      group_by_at(vars(all_of(strata))) %>%
      summarise(mean_RF = mean(RF, na.rm = TRUE))
  }, data = data, strata = strata)


  rsamples <- as.data.frame(bind_rows(rsamples))
  return(rsamples)
}

# the result of 1000 times bootstrap
sample2 <- parallel_stratified_bootstrap(data = new.df.filter, strata = "teamID", N =
↪  1000)
# calculate the sd of RF of each team and rename the columns
sample2.se <- rownames_to_column(as.data.frame(tapply(sample2$mean_RF, sample2$teamID,
↪  sd)),
                                 var = "teamID")
colnames(sample2.se) <- c("teamID", "SD")
head(sample2.se)
```

```
  teamID          SD
1    ALT 0.050670983
2    ANA 0.015836941
3    ARI 0.008843929
4    ATL 0.005984448
5    BAL 0.005367701
6    BFN 0.021340922
```

```r
set.seed(506)

##' @title do the stratified bootstrap using parallel passage
##' @param data, data.frame, the original dataset
##' @param strata, string, within the strata we do
##' @param N, int, the size of bootstrap samples
##'
future_stratified_bootstrap <- function(data, strata, N) {
  # set multisession
  # avoid using all the resources
  plan(multisession, workers = availableCores() - 1)

  # mapping the result into the dataframe
  future_map_dfr(1:N, function(i, .data, .strata){
    bootstrap_sample <- .data %>%
      group_by(across(all_of(.strata))) %>%
      sample_n(size = n(), replace = TRUE) %>%
      summarize(mean_RF = mean(RF, na.rm = TRUE))
  }, .data = data, .strata = strata, .options = furrr_options(seed = TRUE))

}

sample3 <- future_stratified_bootstrap(data = new.df.filter, strata = "teamID", N =
↪  1000)
sample3.se <- rownames_to_column(as.data.frame(tapply(sample3$mean_RF, sample3$teamID,
↪  sd, na.rm = TRUE)),
                                 var = "teamID")
```

```r
colnames(sample3.se) <- c("teamID", "SD")
head(sample3.se)
```

```
  teamID          SD
1    ALT 0.049542224
2    ANA 0.015648703
3    ARI 0.008485279
4    ATL 0.006008002
5    BAL 0.005483009
6    BFN 0.020377663
```

**b. Generate a table showing the estimated RF and associated standard errors for the teams with the 10 highest RF from the three approaches.**

```r
top10_sample1_team <- sample1.df %>%
  group_by(teamID) %>%
  summarize(avg_RF = mean(mean_RF, na.rm = TRUE)) %>%
  slice_max(order_by = avg_RF, n = 10) %>%
  pull(teamID)

top10_sample2_team <- sample2 %>%
  group_by(teamID) %>%
  summarize(avg_RF = mean(mean_RF, na.rm = TRUE)) %>%
  slice_max(order_by = avg_RF, n = 10) %>%
  pull(teamID)

top10_sample3_team <- sample3 %>%
  group_by(teamID) %>%
  summarize(avg_RF = mean(mean_RF, na.rm = TRUE)) %>%
  slice_max(order_by = avg_RF, n = 10) %>%
  pull(teamID)

all_equal <- setequal(top10_sample1_team, top10_sample2_team) &&
↪  setequal(top10_sample2_team, top10_sample3_team)
cat("TOP 10 teams from 3 approaches are the same:", all_equal)
```

```
TOP 10 teams from 3 approaches are the same: TRUE
```

```r
top10_sample1_val <- sample1.df %>%
  filter(teamID %in% top10_sample1_team) %>%
  group_by(teamID) %>%
  summarize(avg_RF_base = mean(mean_RF, na.rm = TRUE),
            sd_RF_base = sd(mean_RF, na.rm = TRUE)) %>%
  arrange(desc(avg_RF_base))

top10_sample2_val <- sample2 %>%
  filter(teamID %in% top10_sample2_team) %>%
  group_by(teamID) %>%
  summarize(avg_RF_parallel = mean(mean_RF, na.rm = TRUE),
            sd_RF_parallel = sd(mean_RF, na.rm = TRUE))

top10_sample3_val <- sample3 %>%
  filter(teamID %in% top10_sample3_team) %>%
  group_by(teamID) %>%
  summarize(avg_RF_future = mean(mean_RF, na.rm = TRUE),
            sd_RF_future = sd(mean_RF, na.rm = TRUE))

final_tibble <- top10_sample1_val %>%
  inner_join(., top10_sample2_val, by = "teamID") %>%
  inner_join(., top10_sample3_val, by = "teamID")

final_tibble
```

```
# A tibble: 10 x 7
   teamID avg_RF_base sd_RF_base avg_RF_parallel sd_RF_parallel avg_RF_future
   <chr>        <dbl>      <dbl>           <dbl>          <dbl>         <dbl>
 1 RC1          0.573     0.0829           0.576         0.0829         0.575
 2 LS1          0.530     0.0541           0.533         0.0567         0.532
 3 ELI          0.526     0.0641           0.529         0.0645         0.529
 4 MLU          0.516     0.126            0.518         0.131          0.506
 5 KEO          0.516     0.110            0.519         0.112          0.515
 6 RIC          0.506     0.0660           0.510         0.0666         0.510
 7 BLA          0.495     0.0313           0.494         0.0317         0.495
 8 LS3          0.489     0.0157           0.489         0.0158         0.489
 9 TRN          0.480     0.0296           0.480         0.0285         0.480
10 PHU          0.479     0.0471           0.478         0.0464         0.481
# i 1 more variable: sd_RF_future <dbl>
```

**Report and discuss the performance difference between the versions.**

```r
set.seed(506)

time1 <- system.time({
  stratified_bootstrap(new.df.filter, strata = "teamID", N = 1000)
})

time2 <- system.time({
  parallel_stratified_bootstrap(data = new.df.filter, strata = "teamID", N = 1000)
})

time3 <- system.time({
  future_stratified_bootstrap(data = new.df.filter, strata = "teamID", N = 1000)
})


##' @title to create a dataframe presenting the time consuming result of each approach
##' @param approach, string, the name of each approach
##' @param time_result, the output of system.time()
```

```r
create_time_df <- function(approach, time_result) {
  data.frame(
    Approach = approach,
    User = time_result["user.self"],
    System = time_result["sys.self"],
    Elapsed = time_result["elapsed"],
    row.names = NULL
  )
}

time_consumption <- rbind(
  create_time_df("Non-parallel", time1),
  create_time_df("Parallel", time2),
  create_time_df("Future", time3)
)

time_consumption
```

```
      Approach   User System Elapsed
1 Non-parallel 19.614  0.888  20.670
2     Parallel  0.180  0.091   9.687
3       Future  0.738  0.185   7.162
```

- The outputs of three approaches have slightly differences but are similar in general. The Usage of parallel processing reduces the running time of bootstrap sharply, among which the parallel package performs the best.