Problem Set #06 Solutions Statistics 506

Problem Set #06

Stratified Bootstrap

```
library(nycflights13)
library(parallel)
library(future)
data(flights)
flights <- as.data.frame(flights)
point_est <- aggregate(flights$air_time,
                       by = list(flights$origin),
                       FUN = mean, na.rm = TRUE)
point_est
```

```
  Group.1        x
1     EWR 153.3000
2     JFK 178.3490
3     LGA 117.8258
```

First, let's find a way to carry out the bootstrap resampling. There are of course *many* ways this can be done.

```
# Obtain samples of row numbers within each `dest`
a <- aggregate(row.names(flights),
               by = list(flights$dest),
               FUN = function(x) {
                 sample(x, replace = TRUE)
               })
# Get the resampled data
resamp <- flights[Reduce(c, a$x), ]
```

```r
# Make sure dimensions are the same
identical(dim(resamp), dim(flights))
```

```
[1] TRUE
```

```r
# Make sure sampling was done within `dest`
identical(sort(table(flights$dest)),
          sort(table(resamp$dest)))
```

```
[1] TRUE
```

Define the function:

```r
f <- function(dat) {
  a <- aggregate(row.names(dat),
                 by = list(dat$dest),
                 FUN = function(x) {
                   sample(x, replace = TRUE)
                 })
  resamp <- dat[Reduce(c, a$x), ]
  results <- aggregate(resamp$air_time,
                       by = list(resamp$origin),
                       FUN = mean, na.rm = TRUE)
  return(results)
}

f(flights)
```

```
  Group.1        x
1     EWR 153.2369
2     JFK 178.6646
3     LGA 117.6979
```

```r
system.time(f(flights))
```

```
   user  system elapsed
  0.420   0.020   0.441
```

This is fine, but slow. (Also, several students mentioned that their solution was significantly faster than mine!) Let's try **data.table**.

```
library(data.table)
flightsdt <- data.table(flights)
f2 <- function(data) {
  resamp <- data[, .SD[sample(x = .N, size = .N, replace = TRUE)], by = dest]
  return(resamp[, .(mean(air_time, na.rm = TRUE)), by = origin])
  return(results)
}

f2(flightsdt)
```

```
   origin       V1
1:    EWR 153.1375
2:    LGA 117.8573
3:    JFK 178.1271
```

```
system.time(f2(flightsdt))
```

```
   user  system elapsed
  0.058   0.003   0.061
```

Much better!

Carry out the simulation

```
reps <- 1000

system.time({
  res1 <- lapply(seq_len(reps),
                 function(x) f2(flightsdt))
})
```

```
   user  system elapsed
 75.090   3.096  78.417
```

Repeat with `mclapply`.

```
system.time({
  res2 <- mclapply(seq_len(reps),
                   function(x) f2(flightsdt),
                   mc.cores = 8)
})
```

```
   user  system elapsed
139.488   5.038  22.906
```

Repeat a third time with futures.

```
plan(multisession)
system.time({
  res3 <- lapply(seq_len(reps),
                 function(x) {
                   future(f2(flightsdt), seed = TRUE)
                 })
  res3 <- lapply(res3, value)
})
```

```
   user  system elapsed
 76.884  15.017 145.393
```

From here, compute the bootrap SEs. Note that using the bootstap SE to compute the confidence interval is one approach; another valid approach would be to estimate the CI directly with the empirical distribution.

```
produce_table <- function(res) {
  sds <- rbindlist(res)[, sd(V1), by=origin][, V1]

  results <- data.frame(est = point_est$x,
                        lb = point_est$x - 1.96*sds,
                        ub = point_est$x + 1.96*sds)
  rownames(results) <- point_est$Group.1
  results
}

# lapply
produce_table(res1)
```

```
         est       lb       ub
EWR 153.3000 152.8967 153.7033
JFK 178.3490 178.1157 178.5824
LGA 117.8258 117.3953 118.2564
```

```
# mclapply
produce_table(res2)
```

```
         est       lb       ub
EWR 153.3000 152.8927 153.7074
JFK 178.3490 178.1224 178.5757
LGA 117.8258 117.3740 118.2776
```

```
# futures
produce_table(res3)
```

```
         est       lb       ub
EWR 153.3000 153.0685 153.5315
JFK 178.3490 177.9260 178.7721
LGA 117.8258 117.3629 118.2887
```