Assignment 1: Comments for the Best Practice

Kohei Kawaguchi

2019/2/21

- You did very well. Most of you got the right result.
- Some of you did not realize that `optim` is a minimizer, but it is okay. You will be carefull next time.
- Here are some comments for the best practice.
- Accumulating the knowledge about the best practice is extremely important to speed up the programming and delivering correct results.
- I cite some students' codes in the following, but please do not feel offended.

Always try to find the best practice

- If you expect to do the same thing more than one time, don't hesitate to pay a fixed cost to consider the best practice.
- The best practice is the practice that minimizes the computation time and coding errors.

Avoid using `for`/`foreach` loop

```
for(m in i){
  y_1 <- t3[m*2-1,5]
  y_2 <- t3[m*2,5]
  if(y_1 > y_2)
    y[m*2-1]<- 1
  else
    y[m*2]<- 1
}
```

- Avoid using `for`/`foreach` loop except for the following cases:
  1. Use a parallel computing with `foreach` + `%dopar%`.
  2. The steps are sequentially dependent.
  3. The task is super easy.
- Otherwise, you can implement the same procedure using the functionalities of `dplyr` or `purrr`.
- The use of `for`/`foreach` loop **substantially** increase the computation time.

Avoid using global variables

```
loglikelihood_A1 <- function(n){
  exp_term <- exp(n * x)
  p <- rep(0, times = 2000)
  for(j in index){
    p[j] <- exp_term[j] / (exp_term[j] + exp_term[j+1])
    p[j+1] <- exp_term[j+1] / (exp_term[j] + exp_term[j+1])
  }
  sum <- 0
  for (l in 1:2000){
    sum <- sum + y[l] * log(p[l])
  }
```

```
  return(sum)
}
```

- Never use global variables.
- You will almost surely have a bug.
- You should always use `codetools::findGlobals()` to check the dependency of a function on global variables.