HW_4

AUTHOR
Tyler Gallagher

HPC

```
fun1 <- function(mat) {
  n <- nrow(mat)
  ans <- double(n)
  for (i in 1:n) {
    ans[i] <- sum(mat[i, ])
  }
  ans
}
fun1alt <- function(mat) {
  rowSums(mat)
}
```

```
fun2 <- function(mat) {
  n <- nrow(mat)
  k <- ncol(mat)
  ans <- mat
  for (i in 1:n) {
    for (j in 2:k) {
      ans[i,j] <- mat[i, j] + ans[i, j - 1]
    }
  }
  ans
}
fun2alt <- function(mat) {
  apply(mat, 1, cumsum)
}
```

Question 1

```
library(microbenchmark)
set.seed(2315)
dat <- matrix(rnorm(200 * 100), nrow = 200)
result_original_fun1 <- fun1(dat)
result_modified_fun1 <- fun1alt(dat)
identical(result_original_fun1, result_modified_fun1)
```

```
[1] TRUE
```

```r
microbenchmark::microbenchmark(
  fun1(dat),
  fun1alt(dat),
  unit = "relative"
)
```

```
Warning in microbenchmark::microbenchmark(fun1(dat), fun1alt(dat), unit =
"relative"): less accurate nanosecond times to avoid potential integer
overflows

Unit: relative
         expr      min       lq     mean   median       uq       max neval
    fun1(dat) 31.48739 31.86905 16.56068 30.88213 30.60949 0.4591315   100
 fun1alt(dat)  1.00000  1.00000  1.00000  1.00000  1.00000 1.0000000   100
```

```r
result_original_fun2 <- fun2(dat)
result_modified_fun2 <- fun2alt(dat)
identical(result_original_fun2, result_modified_fun2)
```

```
[1] FALSE
```

```r
microbenchmark::microbenchmark(
  fun2(dat),
  fun2alt(dat),
  unit = "relative"
)
```

```
Unit: relative
         expr     min       lq     mean   median       uq       max neval
    fun2(dat) 4.53117 4.188943 3.258117 4.114559 4.064135 0.2415826   100
 fun2alt(dat) 1.00000 1.000000 1.000000 1.000000 1.000000 1.0000000   100
```

```r
sim_pi <- function(n = 1000, i = NULL) {
  p <- matrix(runif(n*2), ncol = 2)
  mean(rowSums(p^2) < 1) * 4
}
set.seed(1231)
system.time({
  ans <- unlist(lapply(1:4000, sim_pi, n = 10000))
  print(mean(ans))
})
```

```
[1] 3.14124

   user  system elapsed 
  0.666   0.153   0.822 
```

Question 2

```
library(parallel)

# Parallelized version of sim_pi
sim_pi_parallel <- function(n = 1000, i = NULL) {
  # Function to generate a single estimate of pi
  generate_pi_estimate <- function(dummy) {
    p <- matrix(runif(n * 2), ncol = 2)
    mean(rowSums(p^2) < 1) * 4
  }

  # Create a cluster for parallel processing
  cl <- makeCluster(detectCores())
  pi_estimates <- parLapply(cl, 1:i, generate_pi_estimate)
  stopCluster(cl)
  mean(pi_estimates)
}
system.time({
  result_parallel <- sim_pi_parallel(n = 1000, i = 100)
})
```

```
Warning in mean.default(pi_estimates): argument is not numeric or logical:
returning NA

   user  system elapsed 
  0.008   0.006   0.216 
```

Question 3

```
library(RSQLite)
library(DBI)
con <- dbConnect(SQLite(), ":memory:")
film <- read.csv("https://raw.githubusercontent.com/ivanceras/sakila/master/csv-sakila-db
film_category <- read.csv("https://raw.githubusercontent.com/ivanceras/sakila/master/csv-
category <- read.csv("https://raw.githubusercontent.com/ivanceras/sakila/master/csv-sakil
dbWriteTable(con, "film", film)
dbWriteTable(con, "film_category", film_category)
dbWriteTable(con, "category", category)
```

```
query <- "
SELECT rating, COUNT(*) AS movie_count
FROM film
GROUP BY rating
"
```

```
result <- dbGetQuery(con, query)
print(result)
```

```
  rating movie_count
1      G         180
2  NC-17         210
3     PG         194
4  PG-13         223
5      R         195
```

Question 4

```
query <- "
SELECT
  rating,
  AVG(replacement_cost) AS avg_replacement_cost,
  AVG(rental_rate) AS avg_rental_rate
FROM film
GROUP BY rating
"
result <- dbGetQuery(con, query)
print(result)
```

```
  rating avg_replacement_cost avg_rental_rate
1      G             20.12333        2.912222
2  NC-17             20.13762        2.970952
3     PG             18.95907        3.051856
4  PG-13             20.40256        3.034843
5      R             20.23103        2.938718
```

Question 5

```
query <- "
SELECT
  fc.category_id,
  COUNT(*) AS film_count
FROM
  film_category fc
  JOIN film f ON fc.film_id = f.film_id
GROUP BY
  fc.category_id
"
result <- dbGetQuery(con, query)
print(result)
```

```
   category_id film_count
1            1         64
2            2         66
3            3         60
4            4         57
5            5         58
6            6         68
7            7         62
8            8         69
9            9         73
10          10         61
11          11         56
12          12         51
13          13         63
14          14         61
15          15         74
16          16         57
```

Question 6

```
query <- "
SELECT
  c.category_id,
  c.name AS category_name,
  COUNT(*) AS film_count
FROM
  film_category fc
  JOIN film f ON fc.film_id = f.film_id
  JOIN category c ON fc.category_id = c.category_id
GROUP BY
  c.category_id, c.name
ORDER BY
  film_count DESC
LIMIT 1
"
result <- dbGetQuery(con, query)
print(result)
```

```
  category_id category_name film_count
1          15        Sports         74
```