Assignment 04 - HPC and SQL

September 10, 2021

Due Date

November 19, 2021 by midnight Pacific time.

The learning objectives are to conduct data scraping and perform text mining.

HPC

Problem 1: Make sure your code is nice

Rewrite the following R functions to make them faster. It is OK (and recommended) to take a look at Stackoverflow and Google

```
# Total row sums
fun1 <- function(mat) {
  n <- nrow(mat)
  ans <- double(n)
  for (i in 1:n) {
    ans[i] <- sum(mat[i, ])
  }
  ans
}

fun1alt <- function(mat) {
  # YOUR CODE HERE
}

# Cumulative sum by row
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) {
  # YOUR CODE HERE
}


# Use the data with this code
set.seed(2315)
dat <- matrix(rnorm(200 * 100), nrow = 200)

# Test for the first
microbenchmark::microbenchmark(
  fun1(dat),
  fun1alt(dat), unit = "relative", check = "equivalent"
)

# Test for the second
microbenchmark::microbenchmark(
  fun2(dat),
  fun2alt(dat), unit = "relative", check = "equivalent"
)
```

The last argument, check = “equivalent”, is included to make sure that the functions return the same result.

Problem 2: Make things run faster with parallel computing

The following function allows simulating PI

```r
sim_pi <- function(n = 1000, i = NULL) {
  p <- matrix(runif(n*2), ncol = 2)
  mean(rowSums(p^2) < 1) * 4
}

# Here is an example of the run
set.seed(156)
sim_pi(1000) # 3.132
```

In order to get accurate estimates, we can run this function multiple times, with the following code:

```r
# This runs the simulation a 4,000 times, each with 10,000 points
set.seed(1231)
system.time({
  ans <- unlist(lapply(1:4000, sim_pi, n = 10000))
  print(mean(ans))
})
```

Rewrite the previous code using `parLapply()` to make it run faster. Make sure you set the seed using `clusterSetRNGStream()`:

```r
# YOUR CODE HERE
system.time({
  # YOUR CODE HERE
  ans <- # YOUR CODE HERE
  print(mean(ans))
  # YOUR CODE HERE
})
```

SQL

Setup a temporary database by running the following chunk

```r
# install.packages(c("RSQLite", "DBI"))

library(RSQLite)
library(DBI)

# Initialize a temporary in memory database
con <- dbConnect(SQLite(), ":memory:")

# Download tables
film <- read.csv("https://raw.githubusercontent.com/ivanceras/sakila/master/csv-sakila-db/film.csv")
film_category <- read.csv("https://raw.githubusercontent.com/ivanceras/sakila/master/csv-sakila-db/film_category.csv")
category <- read.csv("https://raw.githubusercontent.com/ivanceras/sakila/master/csv-sakila-db/category.csv")

# Copy data.frames to database
dbWriteTable(con, "film", film)
dbWriteTable(con, "film_category", film_category)
dbWriteTable(con, "category", category)
```

When you write a new chunk, remember to replace the `r` with `sql, connection=con`. Some of these questions will reqruire you to use an inner join. Read more about them here https://www.w3schools.com/sql/sql_join_inner.asp

Question 1

How many many movies is there avaliable in each `rating` catagory.

Question 2

What is the average replacement cost and rental rate for each `rating` category.

Question 3

Use table `film_category` together with `film` to find the how many films there are witth each category ID

Question 4

Incorporate table `category` into the answer to the previous question to find the name of the most popular category.

PM566: Introduction to Health Data Science - PM 566 (Fall 2021)

University of Southern California

Department of Population and Public Health Sciences

George Vega Yon, Kim Siegmund, Abigail Horn

vegayon@usc.edu

View the source at GitHub.