Assignment - 4

Data Scienc with R

Deadline: 14 July 2023, 11:59 pm

Instructions: Read from previous Assignments

Problems:

1. Construct a matrix A which is 1000x1000. Find the norm of every column of the matrix using a loop and then also using function sapply. Which is faster?
2. Improve the efficiency of the following code that calculates AB^tx for matrices and , and vector .

```
n <- 1000
m <- 1000
A <- matrix(runif(n*m), nrow = n, ncol = m)
B <- matrix(rnorm(n*m), nrow = n, ncol = m)
x <- runif(m)
ABtx <- (A %*% t(B)) %*% x
```

3. Load file `ques3.Rdata`. This object contains a 200×200 matrix `mat`. Write `R` code that calculates

$$\text{ans} = \frac{[\det(A)]^{1/p} \cdot p! \cdot (2.7)^p}{p^p \cdot \text{trace}(A)}$$

where for a matrix A, the determinant is the product of its eigenvalues and the trace is the sum of its eigenvalues. The final answer should be stored in `ans` and the last line of the code should be

```
  ans <- ....
```

HINT: you can use function `eigen(mat, only.values = TRUE)` to calculate the eigenvalues of `mat`.

4. Improve the efficiency of the following code.

```
n <- 50
m <- 1e3
A <- matrix(runif(n*m), nrow = n, ncol = m)
# p_vec will store p_i eventually
p_vec <- numeric(length = m)
# running a loop for each column
```

```r
# to find the norm (the numerator)
for(k in 1:m)
{
p_vec[k] <- norm(A[ ,k], type = "2")
}
# divide by the sum
p_vec <- p_vec/sum(p_vec)
# choosing column
chosen <- sample(1:m, size = 1, prob = p_vec)
```

5. Consider the following function `autoreg` below. (It is your job to figure out what the function does.) Write and submit another `R` function `autoreg_fast` that is much faster than the above. (You CANNOT use `Rcpp` here.)

```r
  # n = integer
  # rho = number in (-1,1)
  autoreg <- function(n, rho)
  {
    out <- 0
    for(t in 2:n)
    {
      error <- rnorm(1)
      error <- rho*out[t-1] + error
      out <- c(out, error)
    }
    return(out)
  }
```

6. Write an `R` function called `sel_sums` that does the following tasks in the following order:

a. Takes an argument `mat` which is a square matrix. You may assume that the user will always input a square matrix. Let the size of `mat` be $p \times p$.

b. Randomly chooses a number, s, between 1 and p (inclusive), with equal probability.

c. Calls a C++ function, `sumsC`, that returns the sum of the first s columns of `mat`.

d. Returns the output obtained from the `sumsC` function.