Matrix Multiplication and Time Complexity

Daryl Adopo

9/5/2021

```
library(microbenchmark)
library(tidyverse)
library(dasc2594)
library(patchwork)
```

Understanding Matrix Multiplication Algorithm

The Matrix Multiplication of two matrices of size n x n, **A** and **C**. Is an $n \times n$ matrix **C** where `C[i, j]` is the sum of the values of the ith row of **A** and the jth column of **B** multiplied together. The elementary algorithm for matrix multiplication can be implemented as three nested loops.

```
# use set.seed for reproduciblity
set.seed(2021)

# Iterative Approach
# A %*% B
matrix_multiply <- function(A, B){
  n = ncol(B)
  C <- matrix(0, n, n)
  for(i in 1:n){
    for(j in 1:n){
      for(k in 1:n){
        C[i, j] <- C[i, j] + A[i, k] * B[k, j]
      }
    }
  }
  return(C)
}


n <- 2
A <- matrix(sample(1:10, n*2, replace = TRUE), n, n)
B <- matrix(sample(1:10, n*2, replace = TRUE), n, n)

# Checking our algorithm
all.equal(A %*% B, matrix_multiply(A, B))
```

```
## [1] TRUE
```

```
bm <- microbenchmark(A %*% B, matrix_multiply(A, B))
```

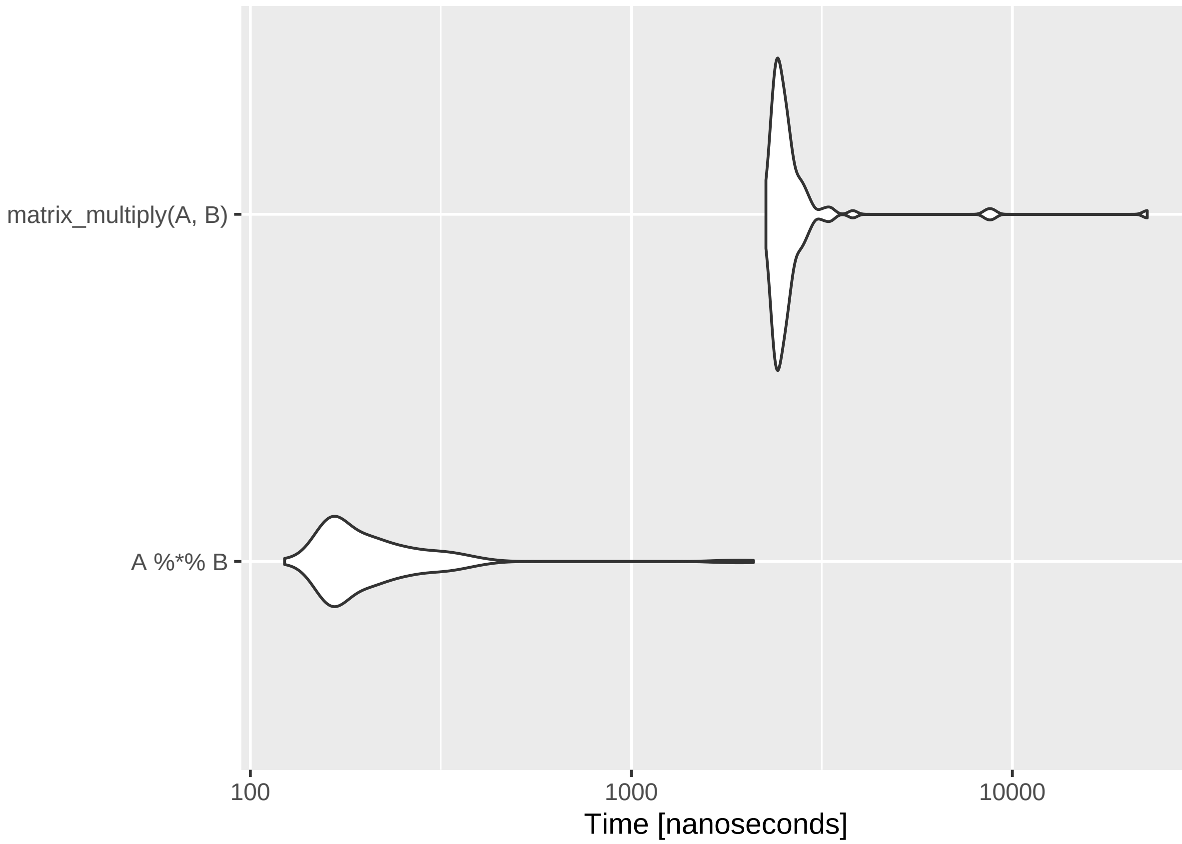
```
## Warning in microbenchmark(A %*% B, matrix_multiply(A, B)): less accurate
## nanosecond times to avoid potential integer overflows
```

```
bm
```

```
## Unit: nanoseconds
##                    expr  min   lq    mean median   uq   max neval
##                 A %*% B  123  164  241.90  184.5  246  2091   100
##  matrix_multiply(A, B) 2255 2378 2864.67 2501.0 2624 22591   100
```

```
# Plotting the results using ggplot function autoplot
autoplot(bm)
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```



Of course, performing the multiplication with the built-in operator `%*%` is way faster than our basic algorithm since it has been optimized through various techniques.

Time Complexity

The time complexity of the matrix multiplication algorythm is calculated by summing the number of multiplications in the three nested loops.

$$\begin{aligned}
M(n,n,n) &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} 1 && (1)\\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} n && (2)\\
&= \sum_{i=1}^{n} n^2 && (3)\\
&= n^3 && (4)
\end{aligned}$$

Therefore, will expect the mean processing time of our algorithm as n increases to look like a cubic distribution.

Let's plot n against the mean processing time for both the simple algorithm we created and using `%*%`

```r
calc_time <- function(n, algo = TRUE) {
        ## create the matrices
        A <- matrix(rnorm(n^2), n, n)
        B <- matrix(rnorm(n^2), n, n)
        #calculate the time
        if(algo){
          bm <- microbenchmark(matrix_multiply(A, B))

        } else{
          bm <- microbenchmark(A %*% B)
          }
        # return the mean time
        return(mean(bm$time))

}
```

```r
n <- 1:10
# initialize the output vector
out <- length(n)

for (i in 1:length(n)) {
        out[i] <- calc_time(n[i])
}

dat <- data.frame(
        n = 1:10,
        mean_time = out)

plot1 <- dat %>%
        ggplot(aes(x = n, y = mean_time)) +
        geom_point() +
        stat_smooth(method = "lm", color = "blue", se = FALSE) +
        stat_smooth(method = "lm", color = "red", formula = y ~ poly(x, 2), se = FALSE) +
        stat_smooth(method = "lm", color = "orange", formula = y ~ poly(x, 3), se = FALSE) +
        ggtitle("Matrix Multiply")
```

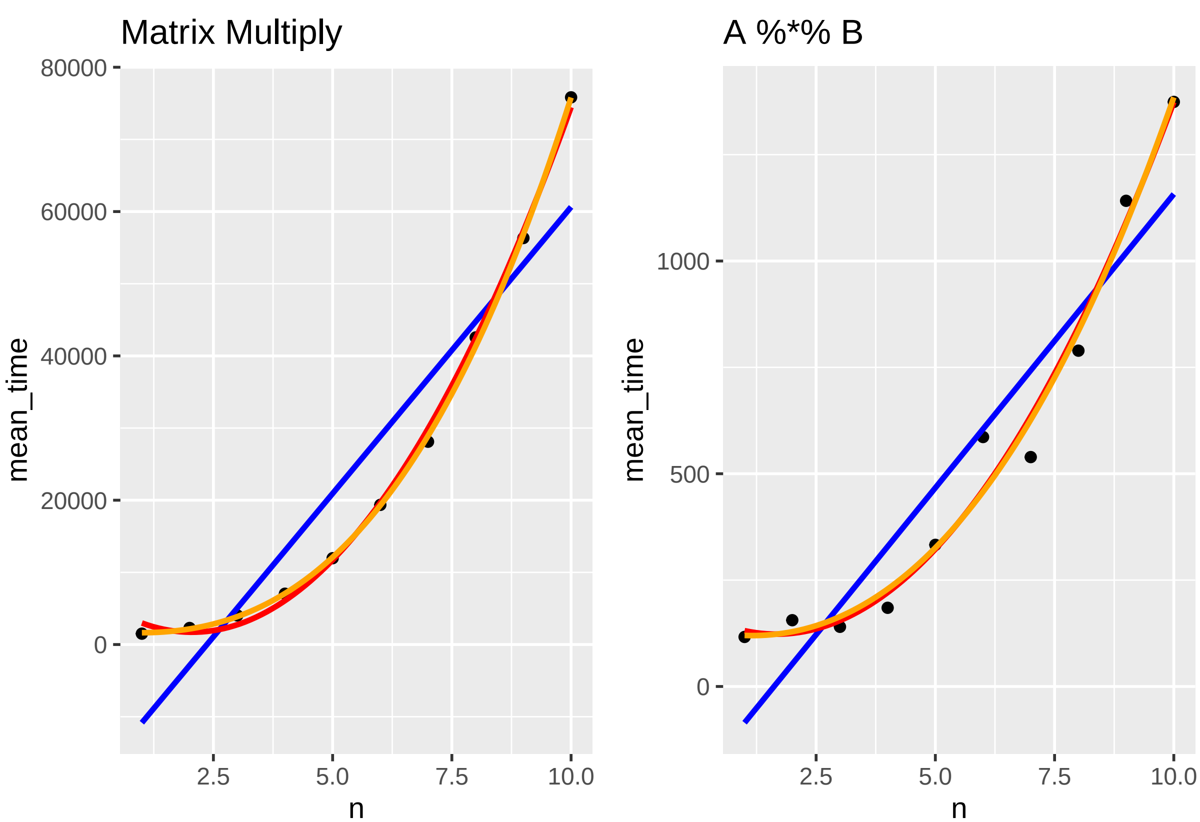
```
for (i in 1:length(n)) {
        out[i] <- calc_time(n[i], algo = FALSE)
}

dat <- data.frame(
        n = 1:10,
        mean_time = out)

plot2 <- dat %>%
        ggplot(aes(x = n, y = mean_time)) +
        geom_point() +
        stat_smooth(method = "lm", color = "blue", se = FALSE) +
        stat_smooth(method = "lm", color = "red", formula = y ~ poly(x, 2), se = FALSE) +
        stat_smooth(method = "lm", color = "orange", formula = y ~ poly(x, 3), se = FALSE) +
        ggtitle("A %*% B")

plot1 + plot2
```

```
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```



Although `A %*% B` is way faster than our `matrix_multiply` function, their processing time both follow a cubic function as n increases. Thus, we say that matrix multiplication of $n \times n$ matrices is $O(n^3)$