Tarea 2

Dalia Camacho, Gabriela Vargas, Elizabeth Monroy

Parte B

```
library(Rcpp)
library(ggplot2)
library(grid)
library(gridExtra)
library(knitr)
sourceCpp( "~/Dropbox/ITAM_MCC/Semestre_1/Algoritmos/Tarea 2/Insercion.cpp")
sourceCpp( "~/Dropbox/ITAM_MCC/Semestre_1/Algoritmos/Tarea 2/Merge.cpp")
set.seed(48970)
```

Ejemplo de prueba

Se genera un arreglo de tamaño 10 con valores del uno al 100.

```
A <- sample(1:100,10)
print(A)
```

```
##  [1] 16 88 87 78 34 79 67 66 46 70
```

Ordenamos con el algoritmo de inserción

```
insercion(A)
```

```
##  [1] 16 34 46 66 67 70 78 79 87 88
```

Ordenamos con el algoritmo merge sort

```
merge_sort(A)
```

```
##  [1] 16 34 46 66 67 70 78 79 87 88
```

Datos para insercion

```
Ns <- c(seq(1,751, by=250), seq(1000, 10000, by=1000),
        seq(2*10^4, 10^5, by=2*10^4), seq(2*10^5, 10^6, by=2*10^5))

Muestra <- sample(10^6)

inserTime <- NULL

for(i in Ns){
     inserTime <- c(inserTime, system.time(insercion(Muestra[1:i]))[3])

}
```

Datos para merge sort

```
mergeTime <- NULL

for(i in Ns){
     mergeTime <- c(mergeTime, system.time(merge_sort(sample(i)))[3])
```

```
}
```

```
Tiempos <- rbind(c("Insercion", format(inserTime[c(5,14,19,24)], digits=2)),
                 c("Merge Sort", formatC(mergeTime[c(5,14,19,24)],digits = 2)))
colnames(Tiempos)<- c("Método", "1000", "10,000", "100,000", "1,000,000")

kable(Tiempos)
```

Método	1000	10,000	100,000	1,000,000
Insercion	0.001	0.020	1.925	209.408
Merge Sort	0	0.001	0.012	0.14

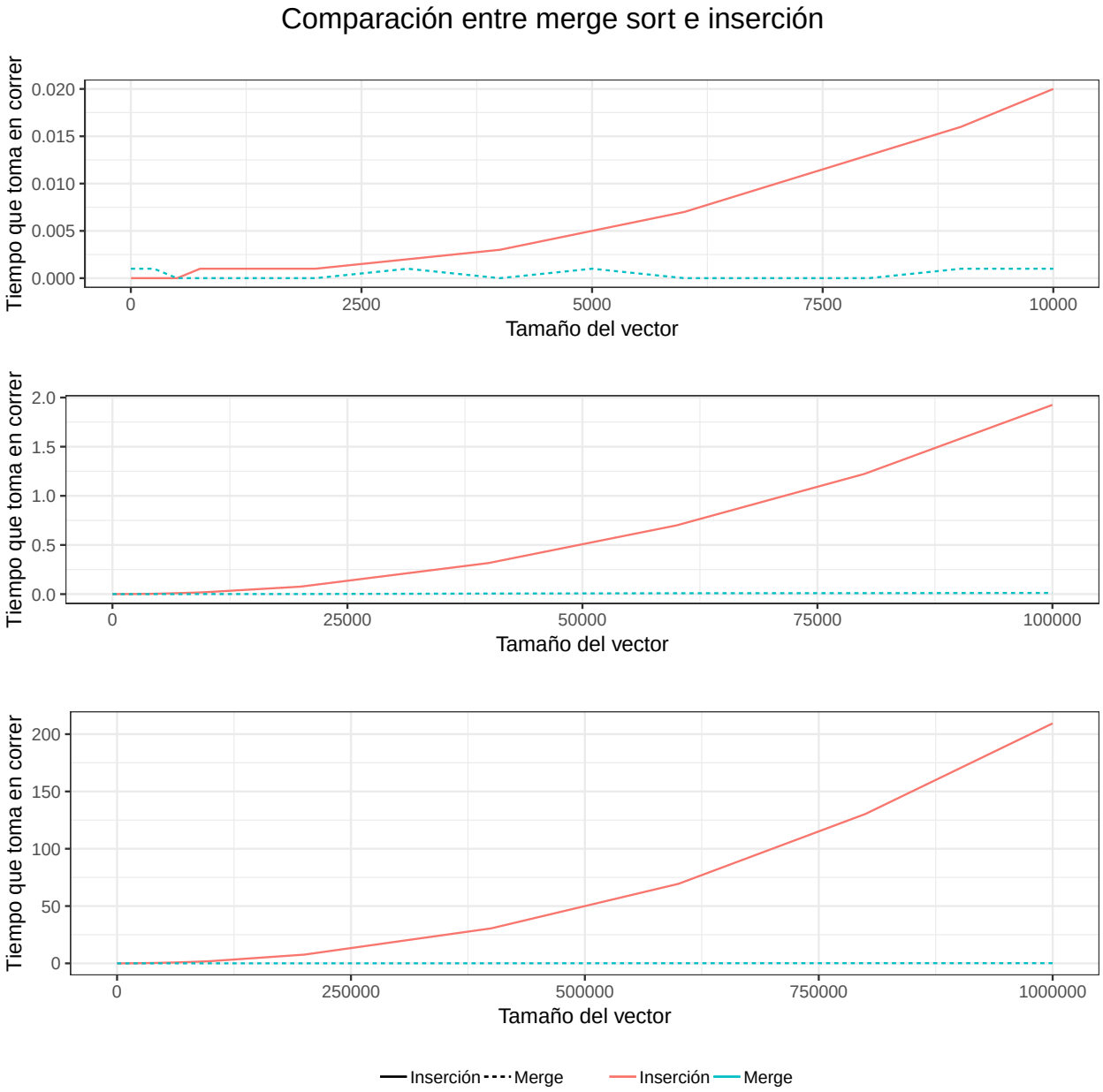
Generar gráficas

```
g1 <- ggplot()+theme_bw()+
  geom_line(aes(Ns[1:14], inserTime[1:14], color="Inserción", linetype="Inserción"))+
  geom_line(aes(Ns[1:14], mergeTime[1:14], color="Merge", linetype="Merge"))+
  ggtitle("")+
  xlab("Tamaño del vector")+
  ylab("Tiempo que toma en correr")+
  theme(legend.position = "none", legend.title = element_blank())

g2 <- ggplot()+theme_bw()+
  geom_line(aes(Ns[1:19], inserTime[1:19], color="Inserción", linetype="Inserción"))+
  geom_line(aes(Ns[1:19], mergeTime[1:19], color="Merge", linetype="Merge"))+
  ggtitle("")+
  xlab("Tamaño del vector")+
  ylab("Tiempo que toma en correr")+
  theme(legend.position = "none", legend.title = element_blank())

g3 <- ggplot()+theme_bw()+
  geom_line(aes(Ns[1:24], inserTime, color="Inserción", linetype="Inserción"))+
  geom_line(aes(Ns[1:24], mergeTime, color="Merge", linetype="Merge"))+
  ggtitle("")+
  xlab("Tamaño del vector")+
  ylab("Tiempo que toma en correr")+
  theme(legend.position = "bottom", legend.title = element_blank())

grid.arrange(g1,g2,g3, top= textGrob("Comparación entre merge sort e inserción", gp=gpar(fontsize=15)),
             heights=c(12,12,17))
```



Parte C

For each function $f(n)$ and time t in ther following table determine the largest size N of a problem that can be solced in time t, assuming that the algorithm to solve the problem takes $f(n)$ microseconds.

```
# Escalar microsegundos a segundos
micro  <- 1.e-6
# Definir tiempos para los que se determina la N
tiempo <- c(Segundo = 1, Minuto = 60, Dia = 60*60*24, Mes = 60*60*24*30, Año = 60*60*24*365)
tiempo_micro <- tiempo/micro
```

Para $log_2(n)$, $\sqrt{n}$ y n resolvemos analíticamente.

```
# log(n)
results <- c("log(n)", paste0("2^(", formatC(tiempo_micro, digits=2), ")"))
```

```
# sqrt(n)
results <- rbind(results,c("sqrt(n)", formatC(tiempo_micro^2, digits=2)))

# n
results <- rbind(results,c("n", formatC(tiempo_micro, digits=2)))
```

Para $n * log_2(n)$ utilizamos el método de Newton para encontrar la N correspondiente.

```
# n*log(n)
#Usando el método de Newton
fun_nlogn <- function(n,tiempo){n*log2(n)-tiempo}

grad_fun_nlogn <- function(n){log2(n)+1/(log(2)*n)}

res <- NULL
tol <- 0.5

for(i in tiempo_micro){
  n <- i/2
  while (abs(fun_nlogn(n, i))>tol) {
    n <- n - fun_nlogn(n, i)/grad_fun_nlogn(n)
  }
  res <- c(res, floor(n))
}

results <- rbind(results, c("n*log(n)", formatC(res,digits = 2, format = "e")))
```

Para n^2, n^3 y 2^n también resolvemos analíticamente.

```
# n^2
results <- rbind(results, c("n^2", formatC(tiempo_micro^(1/2), digits = 2)))

# n^3
results <- rbind(results, c("n^3", formatC(tiempo_micro^(1/3), digits = 2)))

# 2^n
results <- rbind(results, c("2^n", formatC(log2(tiempo_micro), digits = 0, format = "f")))
```

Para $n!$ encontramos la N utilizando búsqueda binaria.

```
# n!

# Búsqueda binaria
funfact <- function(n, tiempo){factorial(n)-tiempo}
res     <- "n!"
for(i in 1:length(tiempo_micro)){
  L <- 1
  U <- as.numeric(results[7,i+1])-1
  while (floor(L)!=floor(U)) {
    M <- (L+U)/2
    if(funfact(M, tiempo_micro[i])<0){
      L <- M
    }else{
      U <- M
    }
```

```
  }
  res <- c(res, floor(U))
}

results <- rbind(results, res)
rownames(results) <- NULL
```

```
kable(results)
```

	Segundo	Minuto	Dia	Mes	Año
log(n)	2^(1e+06)	2^(6e+07)	2^(8.6e+10)	2^(2.6e+12)	2^(3.2e+13)
sqrt(n)	1e+12	3.6e+15	7.5e+21	6.7e+24	9.9e+26
n	1e+06	6e+07	8.6e+10	2.6e+12	3.2e+13
n*log(n)	6.27e+04	2.80e+06	2.76e+09	7.19e+10	7.98e+11
n^2	1e+03	7.7e+03	2.9e+05	1.6e+06	5.6e+06
n^3	1e+02	3.9e+02	4.4e+03	1.4e+04	3.2e+04
2^n	20	26	36	41	45
n!	9	11	13	15	16