Modulo de Programación R-Cran

Acquistapace, Galarraga, Palma, Pedrosa, Portabella, Ruoti, Sánchez,

EJERCICIO 1

Secuencias generada con for

```
 start_timee <- Sys.time()

A <- seq(1,1000000, 2)
 head (A)
```

```
## [1]  1  3  5  7  9 11
```

```
##[1] 1 3 5 7 9 11

 tail (A)
```

```
## [1] 999989 999991 999993 999995 999997 999999
```

```
## [1] 999989 999991 999993 999995 999997 999999

 end_timee <- Sys.time()
end_timee - start_timee
```

```
## Time difference of 0.04645705 secs
```

##Secuencia generada con R

```
 start_timee <- Sys.time()
A <- seq(1,1000000, 2)
 head (A)
```

```
## [1]  1  3  5  7  9 11
```

```
## [1] 1 3 5 7 9 11

 tail (A)
```

```
## [1] 999989 999991 999993 999995 999997 999999
```

Preprint submitted to Facultad de Ingeniería, UnCuyo *June 24, 2021*

```
## [1] 999989 999991 999993 999995 999997 999999

  end_timee <- Sys.time()
  end_timee - start_timee
```

```
## Time difference of 0.03113008 secs
```

EJERCICIO 2

```
  for (i in 0:500000)
 {a<- i
  b<- i+1
  c<- a+b}
  print (c)
```

```
## [1] 1000001
```

Se necesitan 500000 iteraciones para obtener un número mayor a 1000000

EJERCICIO 3

Utilizando Sys.time()

```
    start_time <- Sys.time()
  # Tomo una muestra de 20000 numeros ente 1 y 100000
  x<-sample(1:100000,20000)
  # Creo una funcion para ordenar
  burbuja <- function(x){
  n<-length(x)
  for(j in 1:(n-1)){
  for(i in 1:(n-j)){
  if(x[i]>x[i+1]){
  temp<-x[i]
  x[i]<-x[i+1]
  x[i+1]<-temp
  }
  }
  }
  return(x)
  }
  res<-burbuja(x)
  end_time <- Sys.time()
  #x
  #res
  end_time - start_time
```

```
## Time difference of 59.1104 secs
```

## Ordanacion con el comando SORT de R-Cran

start_time <- Sys.time() x<-sample(1:100000,20000) #sort(x) end_time<- Sys.time() end_time - start_time

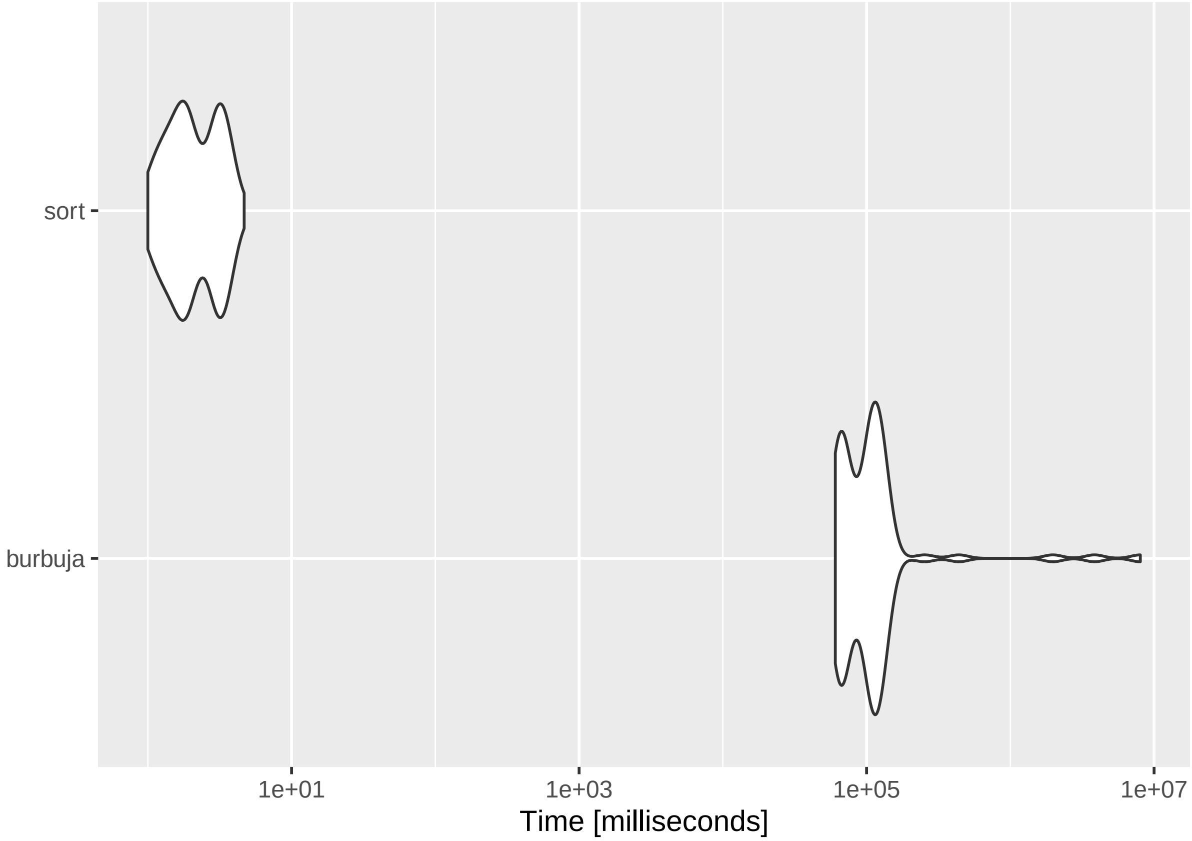
##Utilizando Microbenchmark

```
 library(microbenchmark)
 x<-sample(1:100000,20000)
 check <- function(values) {
 tol <- 1e-12
 error <- max(c(abs(values[[1]] - values[[2]])))
 error < tol
 }
 burbuja <- function(x){
 n<-length(x)
 for(j in 1:(n-1)){
 for(i in 1:(n-j)){
 if(x[i]>x[i+1]){
 temp<-x[i]
 x[i]<-x[i+1]
 flag=FALSE
 2
 x[i+1]<-temp
 }
 }
 }
 return(x)
 }

mbm<- microbenchmark("burbuja"={res<-burbuja(x)},"sort"={res2<-sort(x)})

mbm
```

```
## Unit: milliseconds
##     expr        min         lq         mean       median           uq
##  burbuja 60585.7351 67454.4996 2.362177e+05 105295.63305 118352.52930
##     sort     1.0002     1.5167 2.262656e+00      1.96645      3.05885
##           max neval
##  8021758.4172   100
##        4.6788   100
```

```
## Unit: milliseconds
## expr min lq mean median uq max neval cld
## lm 134.7953 141.9065 151.5039 145.6189 151.6854 192.1529 100 b
## pseudoinverse 174.2721 183.6686 192.6123 188.5093 192.5522 226.0337 100 c
## linear system 102.5293 109.0728 113.4229 112.9266 115.9160 147.2400 100 a
library(ggplot2)
autoplot(mbm)
```

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



Vemos que el metodo sort es mas eficiente en cuanto al consumo de memoria y ademas es mas rapido